Oxygen XML Editor 19.0

User Guide

Contents

Chapter 1: Introduction.....1

Chapter 2: Getting Started.....2
What is Oxygen XML Editor.....2
Getting Familiar with the Layout.....3
Supported Document Types.....4
Resources to Help You Get Started Using Oxygen XML Editor.....4
Your First Document or Project.....6
Your First XML Document.....6
Your First DITA Topic.....9
Creating a New Project.....14
Getting Help.....15
Help Menu.....16
Frequently Used Shortcut Keys.....18

Chapter 3: Installation.....23
Installation Options.....23
Windows Installation.....24
Choosing an Installer.....24
System Requirements.....24
Install Using the Windows Installer.....25
Unattended Installation.....25
Mac OS X Installation.....25
Choosing an Installer.....25
System Requirements.....25
OS X Installation.....26
Linux Installation.....26
Choosing an Installer.....26
System Requirements.....26
Linux Installation.....27
Unattended Installation.....27
Windows Terminal Server Installation.....28
Choosing an Installer.....28
System Requirements.....28
Install Using the Windows Installer.....28
Configuring Windows Terminal Server.....29
Linux Server Installation.....29
Choosing an Installer.....29
System Requirements.....29
Linux Installation.....30
Unix / Linux Server Configuration.....30
Java Web Start (JWS) Installation.....30
Site-wide deployment.....32
Licensing.....32
Choosing a License Type.....32
Obtaining a License.....33
Register a Named-User or Subscription License.....33
Registering a Floating License.....34
Setting up a License Server.....36
Setting up an HTTP Floating License Server.....37

Setting up a TCP Floating License Server Using a 32-bit Windows Installer....41
Setting up a TCP Floating License Server Using All-Platforms Distribution....43
Transferring or Releasing a License....44
Upgrading....45
Checking for New Versions of Oxygen XML Editor....45
What is Preserved During an Upgrade?....45
Upgrading the Standalone Application....45
Installing and Updating Add-ons....45
Uninstalling....46
Uninstalling the Oxygen XML Editor Standalone....46
Unattended Uninstall....47
Oxygen XML Editor Installer Command Line Reference....47

Chapter 4: Configuration....50
Preferences....50
Global Preferences....52
Appearance Preferences....53
Application Layout Preferences....56
Add-ons Preferences....57
Document Type Association Preferences....57
Encoding Preferences....75
Editor Preferences....76
CSS Validator Preferences....113
XML Preferences....113
DITA Preferences....136
Data Sources Preferences....138
SVN Preferences....142
Diff Preferences....145
Archive Preferences....148
Plugins Preferences....149
External Tools Preferences....150
Menu Shortcut Keys Preferences....152
File Types Preferences....153
Open/Find Resource Preferences Page....154
Custom Editor Variables Preferences....156
Network Connection Settings Preferences....156
XML Structure Outline Preferences....159
Views Preferences....159
Messages Preferences....160
Configuring Options....161
Customizing Default Options....161
Storing Global and Project Level Options....162
Sharing Application Settings....163
Importing/Exporting/Resetting Global Options....164
Associating a File Extension with Oxygen XML Editor....164
Configuring the Layout of the Views and Editors....165
Configure Toolbars....167
Import/Export Transformation or Validation Scenarios....169
Editor Variables....169
Custom Editor Variables....174
Custom System Properties....174
Localizing the User Interface....177
Creating an Interface Localization File....177
Setting a Java Virtual Machine Parameter when Launching Oxygen XML Editor....178
Setting Parameters for the Application Launchers....178
Setting Parameters in the Command Line Scripts....179
Creating Custom Startup Parameters File....179

Chapter 5: Perspectives....181
Editor Perspective....181
XSLT Debugger Perspective....182
XQuery Debugger Perspective....183
Database Perspective....184

Chapter 6: Editing Modes....186
Text Editing Mode....186
Navigating the Document Content in Text Mode....186
Text Mode Views....188
Syntax Highlight Depending on Namespace Prefix....206
Presenting Validation Errors in Text Mode....206
Bidirectional Text Support in Text Mode....207
Grid Editing Mode....208
Layouts: Grid and Tree....208
Grid Mode Navigation....209
Bidirectional Text Support in Grid Mode....210
Author Editing Mode....211
Navigating the Document Content in Author Mode....211
Author Mode Views....213
Displaying the Markup....234
Displaying Referenced Content....234
Visual Hints for the Cursor Position....235
Presenting Validation Errors in Author Mode....236
Whitespace Handling in Author Mode....238
Bidirectional Text Support in Author Mode....238
Design Editing Mode (XML Schema Diagram Editor)....240
Navigation in the XML Schema Design Mode....241
XML Schema Outline View....242
XML Schema Attributes View....244
XML Schema Facets View....246
XML Schema Palette View....247

Chapter 7: Editing Documents....248
Working with Unicode....249
Opening and Saving Documents with Unsupported Characters....249
Inserting Symbols....250
Unicode Fallback Font Support....251
Creating and Working with Documents....252
Creating New Documents and Templates....252
Opening Documents....259
Saving Documents....260
Opening and Saving Remote Documents....261
Switching and Moving File Tabs....266
Closing Documents....267
Contextual Menu of the Current Editor Tab....267
Viewing File Properties....268
Searching Documents....268
Open/Find Resource View....268
Open/Find Resource Dialog Box....271
Searching in Content....273
Searching in File Paths....274
Searching in Reviews....275
Technical Aspects....275
Using Projects to Group Documents....275

Creating a New Project.... 275
Project View.... 276
Sharing a Project - Team Collaboration.... 284
Master Files Support.... 286
Editing XML Documents.... 290
Editing XML Documents in Text Mode.... 290
Editing XML Documents in Grid Mode.... 325
Editing XML Documents in Author Mode.... 329
Validating XML Documents.... 444
XML Quick Fixes.... 462
Associating a Schema to XML Documents.... 464
Finding and Replacing Text in the Current File.... 472
Finding and Replacing Text in Multiple Files.... 476
Search and Refactoring Actions for IDs and IDREFS.... 479
Working with Modular XML Files in the Master Files Context.... 481
XML Resource Hierarchy/Dependencies View.... 482
Working with XML Catalogs.... 484
Editing Large XML Documents with DTD Entities or XInclude.... 486
Viewing Status Information.... 488
Making a Persistent Copy of Results.... 488
Editor Highlights.... 489
Printing a Document.... 490
Refactoring XML Documents.... 491
Editing XSLT Stylesheets.... 510
Editing XSLT Stylesheets in the Master Files Context.... 510
Validating XSLT Stylesheets.... 510
XSLT Quick Fix Support.... 512
Content Completion in XSLT Stylesheets.... 514
Syntax Highlighting in XSLT.... 518
XSLT Outline View.... 519
XSLT/XQuery Input View.... 522
XSLT Resource Hierarchy/Dependencies View.... 525
XSLT Component Dependencies View.... 528
Highlight Component Occurrences.... 529
Finding XSLT References and Declarations.... 529
XSLT Stylesheet Component Documentation Support.... 530
XSLT Refactoring Actions.... 531
XSLT Quick Assist Support.... 533
XSLT Unit Test (XSpec).... 535
Generating Documentation for an XSLT Stylesheet.... 536
Compiling an XSL Stylesheet for Saxon.... 543
Editing Ant Build Files.... 544
Editing Ant Build Files in the Context of Master Files.... 544
Validating Ant Build Files.... 544
Ant Quick Fix Support.... 545
Content Completion in Ant Build Files.... 545
Syntax Highlighting in Ant Files.... 546
Ant Outline View.... 546
Ant Resource Hierarchy/Dependencies View.... 549
Ant Component Dependencies View.... 550
Highlight Component Occurrences.... 550
Find References and Declarations of Ant Components.... 550
Ant Refactoring Actions.... 551
Ant Quick Assist Support.... 552
Editing XML Schemas.... 552
Design Editing Mode (XML Schema Diagram Editor).... 553
Editing XML Schema in Text Editing Mode.... 583
Editing XML Schema in the Master Files Context.... 583

Validating XML Schema Documents....584
Quick Fixes for DTD, XSD, and Relax NG Errors....584
Content Completion in XML Schema....585
Syntax Highlighting in XML Schema....585
XML Schema Outline View....586
XML Schema Attributes View....588
XML Schema Palette View....589
XML Schema Facets View....590
XML Schema Resource Hierarchy / Dependencies View....591
Component Dependencies View for XML Schema....593
Highlight Component Occurrences....595
Searching and Refactoring Actions in XML Schemas....595
XML Schema Quick Assist Support....596
Generating Sample XML Files....597
Generating Documentation for an XML Schema....602
Converting Schema to Another Schema Language....610
Converting Database to XML Schema....612
Flatten an XML Schema....613
XML Schema Regular Expressions Builder....615
XML Schema 1.1....616
Setting the XML Schema Version....617
Editing XQuery Documents....618
XQuery Validation....618
Content Completion in XQuery....619
Syntax Highlighting in XQuery....619
Formatting and Indenting XQuery Documents....620
Folding in XQuery Documents....620
XQuery Outline View....621
XQuery Builder View....622
XSLT/XQuery Input View....624
Generating HTML Documentation for an XQuery Document....627
Transforming XML Documents Using XQuery....627
Editing WSDL Documents....631
Editing WSDL Documents in the Master Files Context....631
Validating WSDL Documents....632
Content Completion Assistance in WSDL Documents....632
WSDL Syntax Highlighting....633
WSDL Outline View....633
WSDL Resource Hierarchy/Dependencies View in WSDL Documents....636
Component Dependencies View in WSDL Documents....639
Highlight Component Occurrences in WSDL Documents....640
Searching and Refactoring Operations in WSDL Documents....640
Quick Assist Support in WSDL Documents....642
Generating Documentation for WSDL Documents....643
WSDL SOAP Analyzer....647
Editing CSS Stylesheets....649
Validating CSS Stylesheets....650
Content Completion in CSS Stylesheets....650
Syntax Highlighting in CSS Files....651
CSS Outline View....651
Folding in CSS Stylesheets....652
Formatting and Indenting CSS Stylesheets (Pretty Print)....652
Minifying CSS Stylesheets....652
Editing LESS CSS Stylesheets....653
Validating LESS Stylesheets....653
Content Completion in LESS Stylesheets....653
Syntax Highlighting in LESS Files....654
Compiling LESS Stylesheets to CSS....654

Editing Relax NG Schemas....654
Editing Relax NG Schema in the Master Files Context....655
Relax NG Schema Diagram Editor....655
Validating Relax NG Schema Documents....659
Content Completion in Relax NG Schemas....659
Syntax Highlighting in Relax NG Schemas....660
Quick Fixes for DTD, XSD, and Relax NG Errors....660
Relax NG Outline View....661
RNG Resource Hierarchy/Dependencies View....663
Component Dependencies View for RelaxNG Schemas....665
Searching and Refactoring Actions in RNG Schemas....666
RNG Quick Assist Support....667
Configuring a Custom Datatype Library for a RELAX NG Schema....668
Editing NVDL Schemas....669
NVDL Schema Diagram....669
Validating NVDL Schema Documents....671
Content Completion in NVDL Schemas....671
Syntax Highlighting in NVDL Schemas....672
NVDL Outline View....672
Component Dependencies View for NVDL Schemas....672
Searching and Refactoring Actions in NVDL Schemas....673
Editing JSON Documents....674
Editing JSON Documents in Text Mode....674
Editing JSON Documents in Grid Mode....676
Validating JSON Documents....677
Syntax Highlighting in JSON Documents....677
Folding in JSON....677
JSON Outline View....677
XML to JSON Converter....678
Editing StratML Documents....679
Editing XLIFF Documents....679
Editing JavaScript Documents....680
JavaScript Editing Actions....680
Validating JavaScript Files....681
Content Completion in JavaScript Documents....682
Syntax Highlighting in JavaScript Documents....682
JavaScript Outline View....683
Editing XProc Scripts....683
Editing Schematron Schemas....685
Editing Schematron Schema in the Master Files Context....686
Validating Schematron Documents....686
Presenting Schematron Validation Issues....686
Content Completion in Schematron Documents....687
Syntax Highlighting in Schematron....688
XML Schema or RELAX NG with Embedded Schematron Rules....688
Schematron Outline View....689
Schematron Resource Hierarchy/Dependencies View....690
Highlight Component Occurrences in Schematron Documents....692
Searching and Refactoring Operations in Schematron Documents....692
Quick Assist Support in Schematron Documents....694
Editing Schematron Quick Fixes....695
Schematron Quick Fixes (SQF)....695
Defining Schematron Quick Fixes....695
Validating Schematron Quick Fixes....700
Content Completion in SQF....701
Highlight Quick Fix Occurrences in SQF....701
Searching and Refactoring Operations in SQF....701
Embed Schematron Quick Fixes in Relax NG or XML Schema....702

Integrating SQF in a Framework....703
Editing SVG Files....704
Standalone SVG Viewer....704
Integrated SVG Viewer in the Results Panel....705
Editing XHTML Documents....706
Editing Markdown Documents....706
Markdown Editor....707
Creating New Markdown Documents....708
Actions Available in the Markdown Editor....709
Syntax Highlighting in the Markdown Editor....713
Automatic Validation in Markdown Documents....714
Working with Markdown Documents in DITA....714
Markdown Editor Syntax Rules and Specifications....715
Editing Non-XML Files....724
Spell Checking....725
Spell Check Dictionaries and Term Lists....727
Learned Words....730
Ignored Words (Elements)....731
Automatic Spell Check....731
Spell Check Multiple Files....732
AutoCorrect Misspelled Words....733
Add Dictionaries for the AutoCorrect Feature....734
Loading Large Documents....734
File Sizes Smaller than 300 MB....734
File Sizes Greater than 300 MB....735
Scratch Buffer....735
Handling Read-Only Files....735
Editing Documents with Long Lines....736
XML Digital Signatures....736
Digital Signatures Overview....736
Certificates....738
Canonicalizing Files....738
Signing Files....739
Verifying Signature....740
Example of How to Digitally Sign XML Files or Content....741
Compare Files or Directories....741
Compare Files....742
Compare Directories....753
Compare Directories Against a Base (3-Way)....757

Chapter 8: Document Types and Frameworks....763
Predefined Document Types (Frameworks)....763
DocBook 4 Document Type (Framework)....764
DocBook 5 Document Type (Framework)....778
DocBook 5.1 Assembly....792
DocBook 5.1 Topic....793
DocBook Targetset Document Type (Framework)....794
DITA Topics Document Type (Framework)....794
DITA Map Document Type (Framework)....804
XHTML Document Type (Framework)....833
TEI P5 Document Type (Framework)....837
TEI ODD Document Type (Framework)....843
jTEI Document Type (Framework)....848
JATS Document Type (Framework)....849
EPUB Document Type (Framework)....852
Other Supported Document Types....852

Chapter 9: Publishing....854
Transformation Scenarios.... 854
Built-in Transformation Scenarios.... 855
Creating New Transformation Scenarios....886
Editing a Transformation Scenario....925
Duplicating a Transformation Scenario....926
Configure Transformation Scenario(s) Dialog Box....926
Apply Batch Transformations.... 929
Sharing Transformation Scenarios.... 929
Transformation Scenarios View....930
Debugging PDF Transformations.... 932
Configuring Calabash with XEP....933
Integration of an External XProc Engine....933
XSLT Processors.... 933
XSL-FO Processors.... 936
WebHelp System Output....939
WebHelp Responsive System.... 940
WebHelp Classic System.... 988
WebHelp Classic Mobile System (Deprecated)....1023
Using the Oxygen XML WebHelp Plugin to Automate Output....1045

Chapter 10: Working with XPath Expressions.... 1057
XPath Toolbar....1057
XPath Builder View....1059
XPath Expression Results View.... 1061
XPath and XML Catalogs.... 1062
XPath Prefix Mapping.... 1062

Chapter 11: Working with Archives.... 1063
Browsing Archives....1063
Working with Archive Files.... 1065
Creating an Archive.... 1066
Editing and Saving Files Inside an Archive....1067
Migrating Archives to DITA or TEI....1067

Chapter 12: Databases and CMS.... 1068
Working with Databases....1068
Data Source Explorer View....1068
Table Explorer View.... 1069
Database Connection Support....1072
WebDAV Connections....1111
SQL Execution Support....1113
XQuery and Databases....1115
Content Management System (CMS) Integration....1120
Integration with Documentum (CMS) (deprecated)....1120
Integration with Microsoft SharePoint.... 1126

Chapter 13: Importing Data.... 1132
Import from Text Files....1132
Import from MS Excel Files....1134
Import Data from MS Excel 2007 or Newer....1136
Import Database Data as an XML Document....1136
Import from HTML Files....1138

Import Content Dynamically....1139

Chapter 14: XSLT / XQuery Debugging.... 1142
Debugger Layout.... 1143
Control Toolbar....1144
Debugging Information Views.... 1146
Multiple Output Documents in XSLT 2.0 and XSLT 3.0....1155
Working with the XSLT / XQuery Debugger.... 1155
Steps in a Typical Debugging Process.... 1156
Using Breakpoints....1156
Identify the XSLT / XQuery Expression that Generated Particular Output.... 1157
Debugging Java Extensions.... 1159
Supported Processors for XSLT / XQuery Debugging....1160
Performance Profiling of XSLT Stylesheets and XQuery Documents.... 1160
XSLT/XQuery Performance Profiling Overview....1160
Working with XSLT/XQuery Profiler....1160

Chapter 15: Author Mode Customization.... 1164
Author Mode Customization Guide.... 1164
Simple Customization Tutorial....1164
Advanced Framework Customization.... 1169
Example Files for a Custom Framework....1251
CSS Support in Author Mode....1256
Configuring and Managing Multiple CSS Styles....1256
Handling CSS Imports.... 1259
`oxygen` Media Type.... 1260
CSS At-Rules....1261
Standard W3C CSS Supported Features....1262
Oxygen XML Editor CSS Extensions....1274
Debugging CSS Stylesheets....1311
Creating and Running Automated Tests.... 1312
API Frequently Asked Questions (API FAQ)....1313
Add Custom Actions to the Contextual Menu?....1313
Add Custom Callouts....1314
Add Custom Highlights to Content.... 1317
Auto-Generate an ID When a Document is Opened or Created....1318
Change the Default Track Changes (Review) Author Name....1318
Change the DOCTYPE of an Opened XML Document.... 1319
Control XML Serialization in the Oxygen XML Author Component....1319
Customize the Default Icons for Toolbars/Menus....1320
Customize the Outline View in Text Mode?....1320
Difference Between a Framework (Document Type) and a Plugin Extension.... 1323
Disable Context-Sensitive Menu Items for Custom Author Actions.... 1324
Dynamically Add Form Controls Using a Styles Filter.... 1325
Dynamically Modify the Content Inserted by the Author....1326
Dynamically Open File in Oxygen XML Editor Distributed via JavaWebStart.... 1326
Extend the Java Functionality of an Existing Framework (Document Type).... 1327
Impose Custom Options for Authors....1328
Insert an Element with all the Required Content....1328
Modify the XML Content on Open....1329
Modify the XML Content on Save.... 1330
Multiple Rendering Modes for the Same Document in Author Mode....1331
Obtain a DOM Element from `AuthorNode` or `AuthorElement`....1331
Obtain the Currently Selected Element Using the Author API....1331
Run XSLT or XQuery Transformations....1332
Save a New Document with a Predefined File Name Pattern.... 1332
Split Paragraph on Enter (Instead of Showing Content Completion List)....1333

Use Custom Rendering Styles for Entity References, Comments, or PIs....1333

Chapter 16: Using the Oxygen XML SDK....1336
Extending Oxygen XML Editor with Plugins....1336
General Configuration of an Oxygen XML Editor Plugin....1336
Installing an Oxygen XML Editor Plugin....1337
Types of Plugin Extensions Available with the SDK....1338
Oxygen XML Editor Plugin How to....1348
Creating and Running Automated Tests....1352
Debugging a Plugin Using the Eclipse Workbench....1353
Debugging an Oxygen SDK Extension Using the Eclipse Workbench....1354
Disabling a Plugin....1354
Oxygen XML Author Component....1355
Licensing....1355
Installation Requirements....1356
Customization....1356
Deployment....1358
Sample SharePoint Integration of the Oxygen XML Author Component....1363
Frequently Asked Questions....1368
Feature Matrix....1371
Oxygen XML Web Author Component Component....1372
Deploying Oxygen XML Web Author Component....1372
Oxygen XML Web Author Component Customization Overview....1380
Oxygen XML Web Author Component How to....1388
Oxygen XML Author Options Supported by Web Author....1395
Feature Matrix....1397

Chapter 17: Tools....1399
XML Refactoring....1399
Predefined Refactoring Operations....1402
Custom Refactoring Operations....1406
Storing and Sharing Refactoring Operations....1414
Localizing XML Refactoring Operations....1415
Generate Sample XML Files....1416
Schema Tab (Generate Sample XML Files Tool)....1416
Options Tab (Generate Sample XML Files Tool)....1417
Advanced Tab (Generate Sample XML Files Tool)....1419
Running the Generate Sample XML Files Tool from the Command Line....1420
Generate/Convert Schema....1421
Convert DB Structure to XML Schema....1423
Flatten Schema....1423
XML to JSON....1425
Compile XSL Stylesheet for Saxon....1426
Format and Indent Files....1427
Generate Documentation....1428
XML Schema Documentation....1428
XSLT Stylesheet Documentation....1431
XQuery Documentation....1434
WSDL Documentation....1435
Canonicalize....1437
Sign....1438
Verify Signature....1440
WSDL SOAP Analyzer....1440
Testing Remote WSDL Files....1442
UDDI Registry Browser....1442
XML Schema Regular Expressions Builder....1443
Large File Viewer....1445

Hex Viewer....1447
SVG Viewer....1447
Tree Editor....1449
Compare Files....1449
Toolbar and Contextual Menu Actions of the Compare Files Tool....1455
Compare Files Tool Menus....1457
Compare Directories....1460
Toolbar and Contextual Menu Actions of the Compare Directories Tool....1462
Compare Directories Tool Menus....1463
Compare Images....1464
Compare Directories Against a Base (3-Way)....1464
Syncro SVN Client....1469
Main Window....1469
Getting Started....1479
Syncro SVN Client Views....1529
Revision Graph of a SVN Resource....1555
Oxygen XML Editor SVN Preferences....1559
Entering Local Paths and URLs....1559
Technical Issues....1560
External Tools....1562

Chapter 18: Common Problems....1564
Performance Problems and Solutions....1564
Display Problems on Linux or Solaris....1564
Out of Memory on External Processes....1564
Too many nested apply-templates calls Error When Running a Transformation....1565
Performance Issues with Large Documents....1565
Misc Problems and Solutions....1565
'Address Family Not Supported by Protocol Family; Connect' Error....1565
Alignment Issues of the Main Menu on Linux Systems Based on Gnome 3.x....1566
Archive Distribution Fails to Run on Mac OS 10.12 (Sierra)....1566
Cannot Associate Oxygen XML Editor With a File Type on My Windows Computer....1567
Cannot Connect to SVN Repository from Repositories View....1567
Cannot Open XML Files in Internet Explorer....1567
Compatibility Issue Between Java and Certain Graphics Card Drivers....1568
Crash at Startup on Windows with an Error About the `nvoglv32.dll` File....1568
Damaged File Associations on OS X....1568
Details to Submit in a Request for Technical Support Using the Online Form....1569
DITA Map Transformation Fails (Cannot Connect to External Location)....1570
DITA PDF Transformation Fails....1570
DITA to CHM Transformation Fails....1570
Drag and Drop Without Initial Selection Does Not Work....1571
Gray Window on Linux With the Compiz / Beryl Window Manager....1571
Image Appears Stretched Out in the PDF Output....1571
Increasing the Memory for the Ant Process....1572
JPEG CMYK Color Space Issues....1572
Keyboard Shortcuts Do Not Work....1573
Keyboard Language Resets to Default on Windows....1573
MSXML 4.0 Transformation Issues....1573
Navigation to the web page was canceled when viewing CHM on a Network Drive....1573
Out Of Memory Error When Opening Large Documents....1574
Oxygen XML Editor Crashed on My Mac OS X Computer....1574
Oxygen XML Editor Takes Several Minutes to Start....1574
PDF Processing Fails When Publishing DITA Content....1574
Scroll Function of my Notebook Trackpad is Not Working....1576
Segmentation Fault Error on Mac OS X....1576
Set Specific JVM Version on Mac OS X....1576
Signature Verification Failed Error on a Resource from Documentum....1576

Special Characters are Replaced with a Square in Editor....1577
Syntax Highlight Not Available in Eclipse Plugin.... 1577
TocJS Transformation Does not Generate All Files for a Tree-Like TOC....1577
Topic References Outside the Main DITA Map Folder....1578
Wrong Highlights of Matched Words in a Search in User Manual....1578
XML Document Takes a Long Time to Open.... 1578
XSLT Debugger Is Very Slow....1579

Chapter 19: DITA Authoring and Publishing.... 1580
Working with DITA Maps....1581
DITA Maps Manager....1582
Creating a Map....1590
Managing DITA Maps....1592
Chunking DITA Topics.... 1609
DITA Map Validation and Completeness Check....1609
DITA Map Author Mode Actions....1614
Working with DITA Topics.... 1617
Creating a New DITA Topic....1618
Fast Create Multiple DITA Topics....1620
Editing DITA Topics.... 1622
Converting DITA Topics to Another Type.... 1624
Adding Images in DITA Topics.... 1625
Adding Video, Audio, and Embedded HTML Resources in DITA Topics....1626
Image Maps in DITA....1628
Adding Tables in DITA Topics....1633
Adding MathML Equations in DITA Topics....1642
DITA Author Mode Actions.... 1643
Working with Markdown Documents in DITA.... 1651
Working with Keys....1652
Reusing DITA Content....1653
Reusing DITA Topics in Multiple Maps....1654
Working with Content References....1655
Working with the Conref Push Mechanism....1665
Working with Reusable Components.... 1667
Working with Variable Text in DITA....1668
Working with DITA 1.3 Key Scopes....1669
Working with DITA 1.3 Branch Filtering.... 1670
DITA Reusable Components View....1671
Linking in DITA....1673
Hierarchical Linking in DITA Maps.... 1674
Linking in DITA Topics....1674
Linking with Relationship Tables in DITA....1678
Publishing DITA Output....1679
Transforming DITA Content.... 1680
DITA Profiling / Conditional Text.... 1698
Creating and Editing Profiling Attributes in DITA.... 1698
Apply Profiling Attributes in DITA....1700
Creating and Editing Profiling Condition Sets in DITA....1701
Apply Profiling Condition Sets in DITA....1703
Showing and Filtering Profiled Content in DITA....1705
Customizing Colors and Styles for Rendering Profiling in Author Mode....1707
Profiling with a Subject Scheme Map....1708
Filtering Attribute Values with a DITAVAL File.... 1709
Styling the Rendering of Profiled Content Using a DITAVAL File....1711
Publishing Profiled DITA Content.... 1711
DITA Open Toolkit Support....1712
Creating a DITA OT Customization Plugin....1712
Installing a Plugin in the DITA Open Toolkit....1714

Use an External DITA Open Toolkit in Oxygen XML Editor.....1715
Third-Party DITA Open Toolkit Plugins.....1715
DITA Specialization Support.....1715
Integration of a DITA Specialization.....1715
Editing DITA Map Specializations.....1716
Editing DITA Topic Specializations.....1716
Master Files Support in DITA.....1716
Metadata.....1718
Migrating MS Office Documents to DITA.....1718
DITA 1.3 Support.....1719

Chapter 20: Glossary.....1722

Index.....1730

Copyright.....

Introduction

1

Welcome to the User Manual of Oxygen XML Editor 19.0.

Oxygen XML Editor is a cross-platform application designed to accommodate all of your XML editing, authoring, developing, and publishing needs. It is the best XML editor available for document development using structured mark-up languages such as XML, XSD, Relax NG, XSL, DTD. It is a comprehensive solution for authors who want to edit XML documents visually, with or without extensive knowledge about XML and XML-related technologies. The WYSIWYG-like editor is driven by CSS stylesheets associated with the XML documents and offers many innovative, user-friendly authoring features that make XML authoring easy and powerful.

It offers developers and authors a powerful **Integrated Development Environment** and the intuitive **Graphical User Interface** of Oxygen XML Editor is easy to use and provides robust functionality for content editing, project management, and validation of structured mark-up sources. Coupled with *XSLT* and *FOP* transformation technologies, Oxygen XML Editor offers support for generating output to multiple target formats, including: *PDF*, *PS*, *TXT*, *HTML*, *JavaHelp*, *WebHelp*, and *XML*.

This user guide is focused on describing features, functionality, the application interface, and to help you quickly get started. It also includes a vast amount of advanced technical information and instructional topics that are designed to teach you how to use Oxygen XML Editor to accomplish your tasks. It is assumed that you are familiar with the use of your operating system and the concepts related to XML technologies and structured mark-up.

Getting Started

2

Topics:

- *What is Oxygen XML Editor*
- *Getting Familiar with the Layout*
- *Supported Document Types*
- *Resources to Help You Get Started Using Oxygen XML Editor*
- *Your First Document or Project*
- *Getting Help*
- *Frequently Used Shortcut Keys*

This section provides a variety of resources to help you get the most out of the application. Typically, the first step of getting started with Oxygen XML Editor would be to install the software. For detailed information about that process, see the *Installation chapter*.

After installation, when you launch Oxygen XML Editor for the first time, you are greeted with a **Welcome** dialog box. It presents upcoming events, useful video demonstrations, helpful resources, the tip of the day, and also gives you easy access to recently used files and projects and to create new ones.

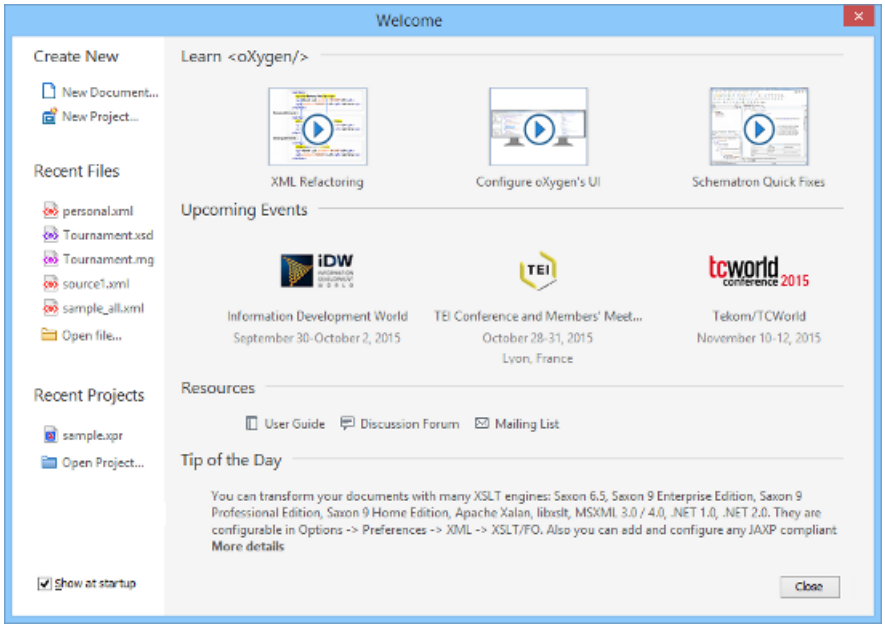


Figure 1: Welcome Dialog Box

If you do not want it to be displayed every time you launch Oxygen XML Editor, deselect the **Show at startup** option in the bottom-left corner of the dialog box. To display it any time, go to **Help** > **Welcome**.

What is Oxygen XML Editor

Oxygen XML Editor is the best XML editor available and is a complete XML development and authoring solution. It is designed to accommodate a large number of users, ranging from beginners to XML experts. It is the only XML tool that supports all of the XML schema languages and provides a large variety of powerful tools for editing and publishing XML documents.

You can use Oxygen XML Editor to work with most XML-based standards and technologies. It is a cross-platform application available on all the major operating systems (Windows, Mac OS X, Linux, Solaris) and can be used either as a standalone application or as an Eclipse plugin.

For a list of many of the features and technologies that are included in Oxygen XML Editor, see the *Oxygen Website*.

Getting Familiar with the Layout

Oxygen XML Editor includes several *perspectives* and *editing modes* to help you accomplish a wide range of tasks. Each *perspective* and editing mode also includes a large variety of helper view, menu actions, toolbars, and contextual menu functions.

Regardless of the *perspective* or *editing mode* that you are working with, the default layout is comprised of the following areas:

Menus

Menu driven access to all the features and functions available in Oxygen XML Editor. Most of the menus are common for all types of documents, but Oxygen XML Editor also includes some context-sensitive and *framework*-specific menus and actions that are only available for a specific context or type of document.

Toolbars

Easy access to common and frequently used functions. Each icon is a button that acts as a shortcut to a related function. Some of the toolbars are common for all *perspectives*, editing modes, and types of documents, while others are specific to the particular *perspective* or mode. Some toolbars are also *framework*-specific, depending on the type of document that is being edited. All the *toolbars can be configured* to suit your specific needs.

Helper Views

Oxygen XML Editor includes a large variety of *dockable* views to assist you with editing, viewing, searching, validating, transforming, and organizing your documents. Many of the views also contain useful contextual menu actions, toolbar buttons, or menus. The most commonly used views for each *perspective* and editing mode are displayed by default and you can choose to display others to suit your specific needs. The *layout of the views can also be configured* according to your preferences. \

Editor Pane

The main editing area in the center of the application. Each *editing mode* provides a main editor pane where you spend most of your time reading, editing, applying markup, and validating your documents. The editor pane in each *editing mode* also includes a variety of contextual menu actions and other features to help streamline your editing tasks.

Perspectives

Oxygen XML Editor includes *several different perspectives* that you can use to work with your documents. The **Editor** *perspective* is the most commonly used *perspective* used for displaying and editing the content of your XML documents, and it is the default *perspective* when you start Oxygen XML Editor for the first time. Oxygen XML Editor also includes a **Database** *perspective* that allows you to manage databases and their connections and a few debugging *perspectives* that allow you to detect problems in XSLT or XQuery transformations.

Status Bar

The status bar at the bottom of the application contains some useful information when your working with documents. It includes the following information, in the order it is displayed from left to right:

- The path of the current document.
- The *Unicode value* for the character directly to the right of the current cursor position.
- The status of the current document. The status of **Modified** is displayed for documents that have not yet been saved. Otherwise, this section is left blank.
- In ***Text*** *editing mode*, the current line and character position is displayed.
- If the ***Check for notifications*** *option* is selected, this section will show you when new messages have been received. The types of messages include the addition of new videos on the Oxygen XML Editor website, the announcement of upcoming webinars and conferences where the Oxygen XML Editor team will participate, and more.

- The memory consumption, including the memory used by the application and the maximum amount that is allocated to the application.
- If the ***Show memory status*** *option* is selected, a **Free unused memory** icon is displayed in the bottom-right corner and you can use this icon to free up unused memory.

Supported Document Types

You can use the main editing pane in Oxygen XML Editor to edit a large variety of document types.

The supported document types include the following:

- - XML documents
- - XSLT stylesheets
- - XML Schema
- - DTD (Document Type Definition) schemas
- - RELAX NG full syntax schemas
- - RELAX NG compact syntax schemas
- - NVDL (Namespace-based Validation Dispatching Language) schemas
- - XSL:FO documents
- - XQuery documents
- - WSDL documents
- - Schematron documents
- - JavaScript documents
- - Python documents
- - CSS documents
- - LESS documents
- - XProc scripts
- - SQL documents
- - JSON documents
- - Ant build scripts
- - Markdown documents

Resources to Help You Get Started Using Oxygen XML Editor

Configuring Oxygen XML Editor

There are numerous ways that you can configure Oxygen XML Editor to accommodate your specific needs.

- See the *Configuring Oxygen section* for details on the various ways that you can configure the application and its features.

Video Tutorials

The Oxygen XML Editor website includes numerous video demonstrations and webinars that present many of the features that are available in Oxygen XML Editor and show you how to complete specific tasks or how to use the various features.

- Go to the *Oxygen XML Editor Videos Page* to see the list of video tutorials and webinars.

Oxygen XML Editor Documentation

The Oxygen XML Editor documentation includes a plethora of sections and topics to provide you with a variety of information, ranging from basic authoring tasks to advanced developer techniques. You can, of course, search through the documentation using standard search mechanisms, but you can also place the cursor in any particular position in the interface and use the **F1** key to open a dialog box that presents a section in the documentation that is appropriate for the context of the current cursor position. Aside from the other topics in this *Getting Started* section, the following are links to other sections of the documentation that might be helpful for your specific needs:

- ***Text Editing Mode Section*** - Provides information about the **Text** editor.
- ***Author Editing Mode Section*** - Provides information about the visual WYSIWYG-like **Author** editing mode.
- ***Editing Documents Section*** - Includes information about editing numerous different types of documents.
- ***DITA Authoring and Publishing Section*** - Provides information about using DITA to edit and structure your content.
- ***WebHelp System Output Section*** - Provides information about the WebHelp system that can be used for publishing content.
- ***Importing Data Section*** - Provides information about importing data from text files, MS Excel files, database data, and HTML files.

Sample Documents

Your installation of Oxygen XML Editor includes a large variety of sample documents and projects that you can use as templates to get started and to experiment with the various features and technologies. They are located in the **samples** folder that is located in the installation directory of Oxygen XML Editor. You will find files and folders for various types of documents, including the following:

- **`sample.xpr` file** - A sample project file that will allow you to experiment with how projects can be structured and used. When you open this project file, you will be able to see all the sample files and folders in the ***Project*** *view*.
- **`personal` files** - A collection of interrelated sample files that will allow you to experiment with the structure and relationship between XML files, stylesheets, and schemas.
- **Various document type folders** - The various folders contain sample files for numerous document types, such as CSS, DITA, DocBook, ePub, TEI, XHTML, and many others.

Other Resources

The following list includes links to various other resources that will help you get started using the features of Oxygen XML Editor:

- See the *Oxygen XML Editor Blog Site* for a large variety of current and archived blogs in regards to numerous features, requests, and instructional topics.
- Take advantage of the *Oxygen XML Editor Forum* to see various announcements and learn more about specific issues that other users have experienced.
- If you are using DITA, see the incredibly helpful *DITA Style Guide Best Practices for Authors*.
- To learn about the WebHelp features in Oxygen XML Editor, see the *Publishing DITA and DocBook to WebHelp* section of the website.
- For more information about various additional tools that are integrated into Oxygen XML Editor, see the *Tools section*.
- See the *External Resource Page* for links to various other helpful resources, such as discussion lists, external tutorials, and more.
- See the *Oxygen SDK* section for details about the SDK that allows you to extend and develop Oxygen XML Editor *frameworks* and *plugins*, and to integrate Eclipse plugins.
- For a list of new features that were implemented in the latest version of Oxygen XML Editor, see the *What's New Section of the Website*
- You can select the ***Tip of the Day*** action in the ***Help*** *menu* to display a dialog box that includes a variety of tips for using Oxygen XML Editor.
- You can select ***Show Dynamic Help view*** from the ***Help*** *menu* to dynamically opens a topic that is relevant to the focused editor, view, or dialog box.

Your First Document or Project

This section includes several topics that will help you get started with your first document or project.

Your First XML Document

To create your first XML document, select **File** > **New** or click the **New** button on the toolbar. The ***New document wizard*** is displayed:

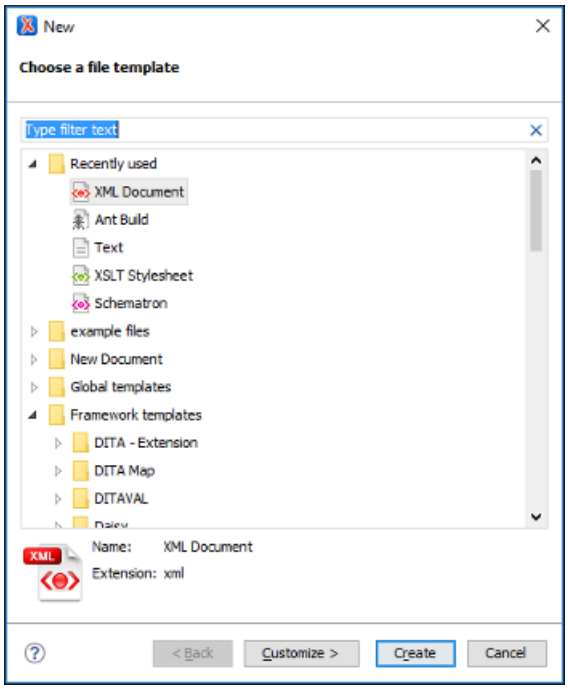


Figure 2: New Document Wizard

You can either create a new XML document from scratch by choosing one of the available types in the wizard. You can also create one from a template by choosing a template from the **Global templates** or **Framework templates** folders. If you are looking for a common document type, such as DITA or DocBook, you can find templates for these document types in the **Framework templates** folder. If your company has created its own templates, you can also find them there. After you use this dialog box to create a few documents, those document types will appear in the **Recently used** folder, which allows you to easily create other new documents of those types.

For some document types, you may find a lot of different templates. For example, there are numerous templates for DocBook documents, and DITA topic types and maps. Choose the template that best meets your needs.

Writing Your First Document

Depending on the type of document you choose, the Oxygen XML Editor interface changes to support editing that document type. This may include new menus, toolbar buttons, and items in the contextual menus.

Also, depending on the type of document you choose, Oxygen XML Editor may open your document in ***Text*** or ***Author*** mode. **Text** mode shows the raw XML source file, while **Author** mode shows a graphical view of the document.

The availability of **Author** mode for your document type depends on the type you choose and if there is a CSS stylesheet available to create the **Author** mode. Oxygen XML Editor includes default **Author** mode views for most of the document types it supports. If your company has created its own document types, **Author** mode stylesheets may have also been created for that type. However, if you create a plain XML file, or one based on a schema that is not included in the Oxygen XML Editor built-in support, you need to edit it in **Text** mode or *create your own **Author** mode style sheet* for it.

You can switch back and forth between **Author** mode and **Text** mode at any time by clicking the buttons at the bottom left of the editor window. You do not lose any formatting when switching from **Author** to **Text** mode. **Text** and **Author** modes are just different views for the same XML document. There is also a ***Grid** mode* available, which is useful for certain kinds of documents, particularly those that are structured like databases. You can also use it to *sort things such as list items and table rows*.

If you use **Author** mode, you might find that it is similar to word processors that you are used to. Likewise, the **Text** mode is similar to many typical text editors. If you are new to XML, the biggest difference is that XML documents have a particular structure that you have to follow. Oxygen XML Editor assists you with a continuous validation of the XML markup.

Structuring Your First Document

Each XML document type has a particular structure that you have to follow as you write and edit the document. Some document types give you a lot of choices, while others give you very few. In either case, you need to make sure that your document follows the particular structure for the document type you are creating. This means:

- At any given location in the document, there are only certain XML elements allowed. Oxygen XML Editor helps you determine which elements are allowed. In **Author** mode, when you press **Enter**, Oxygen XML Editor assumes that you want to enter a new element and shows you a list of elements that can be created in this location. Keep typing until the element you want is highlighted and press **Enter** to insert the element. If you want to view the overall structure of a document and see what is allowed (and where), you can use the ***Model** view* (**Window** > **Show View** > **Model**).
- When you create certain elements, you may find that your text gets a jagged red underline and you get a warning that your content is invalid. This is usually because the element you have just created requires certain other elements inside of it. Your document will be invalid until you create those elements. Oxygen XML Editor does its best to help you with this. If there is only one possible element that can go inside the element you just created, Oxygen XML Editor creates it for you. However, if there is more than one possibility, you have to create the appropriate elements yourself. In many cases, Oxygen XML Editor presents *XML Quick Fixes* that help you resolve errors by offering proposals to quickly fix problems such as missing required attributes or invalid elements.

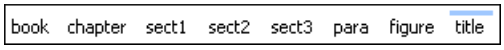
Editing Your First Document

Once you have completed the first draft of your document, you may need to edit it. As with any editor, Oxygen XML Editor provides the normal cut, copy, and paste options as well as drag and drop editing. However, when you are editing an XML document, you have to make sure that your edits respect the structure of the XML document type. In fact, you are often editing the structure as well as the content of your document.

Oxygen XML Editor provides many tools to help you edit your structure and to keep your structure valid while editing text.

The Document Breadcrumbs

Across the top of the editor window, there is a set of breadcrumbs that shows you exactly were the insertion point is in the structure of the document. You can click any element in the breadcrumbs to select that entire element in the document.



Showing Tags

To see exactly where you are in the structure of the document, you can show the tags graphically in the **Author** view. There are several levels of tag visibility that you can choose using the ***Display tags mode** drop-down menu* on the toolbar (the button may look a little different than this, as it changes to reflect the level of tags currently displayed).

Outline View

The ***Outline** view* shows you the structure of your document in outline format. You can use it to select elements, or to move elements around in the document.

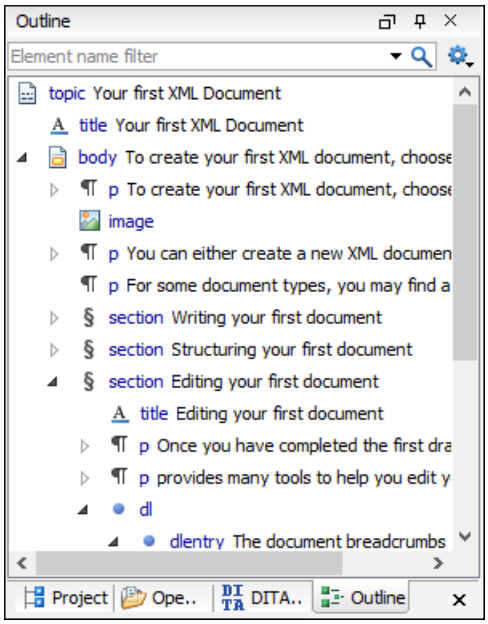


Figure 3: Outline View

You can configure the **Outline** view to determine what is shown, such as element names, attributes, and comments. Certain choices may work better for particular document types. You can also filter the **Outline** view to show only elements with a certain name.

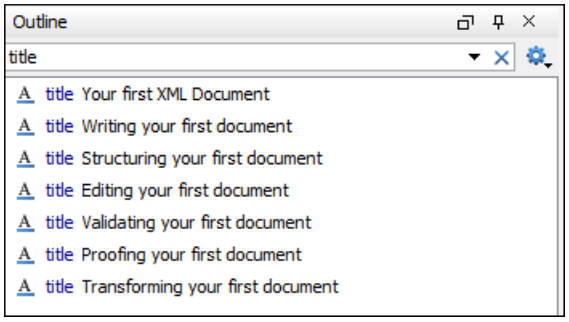


Figure 4: Outline View Filtered to only Show Element Names

Cut and Paste, Drag and Drop

You can cut and paste or drag and drop text, just as you would in any other editor. However, when you do this in **Author** view, it is important to remember that you are actually moving blocks of XML. When you cut and paste or drag and drop a block of XML, the result has to be valid both where the content is inserted, and where it is removed from.

A big part of doing this correctly is to make sure that you pick up the right block of text in the first place. Using the breadcrumbs or **Outline** view, or showing tags and using them to select content, can help ensure that you are selecting the right chunk of XML.

If you do try to paste or drop a chunk of XML somewhere that is not valid, Oxygen XML Editor warns you and tries to suggest actions that make it valid (such as by removing surrounding elements from the chunk you are moving, by creating a new element at the destination, or by inserting it in a nearby location).

If you are using **Author** mode, you can also switch to **Text** mode to see exactly which bits of XML you are selecting and moving.

Refactoring actions

You can perform many common structure edits, such as renaming an element or wrapping text in an element, using the actions in the **Refactoring** menu of the contextual menu (or the **Document** > **Markup** menu). More advanced refactoring operations are also available using the ***XML Refactoring*** *tool* that is available in the **Tools** menu.

Validating Your First Document

Validation is the process of making sure that an XML document abides by the rules of its schema. If Oxygen XML Editor knows how to find the schema, it validates the document for you as you type. Oxygen XML Editor finds the schema automatically for most of the document types created from templates. However, in some cases you may have to *tell it how to find the schema*.

When Oxygen XML Editor validates as you type, there is a small bar at the right edge of the editor that shows you if the document is invalid and where errors are found. If the indicator at the top of that bar is green, your document is valid. If the document is invalid, the indicator turns red and a red flag shows you where the errors are found. Click that flag to jump to the error. Remember that sometimes your document is invalid simply because the structure you are creating is not yet complete.

In addition to problems with the validity of the XML document itself, Oxygen XML Editor also reports warnings for a number of conditions, such as if your document contains a cross reference that cannot be resolved, or if Oxygen XML Editor cannot find the schema specified by the document. The location of these warnings is marked in yellow on the validation bar. If the document contains warnings, but no errors, the validity indicator turns yellow.

You can also validate your document at any time by selecting the **Validate** action from the **Validation** toolbar drop-down menu or the **Document** > **Validate** menu.. When you validate in this manner, if errors are found, the validation result opens in a new pane at the bottom of the editor that shows each validation error on a separate line. Clicking the error takes you to the location in your document where the error was detected.

Note: Be aware that the problem is sometimes in a different location from where the validator detects the error. To get more information about a validation error, right-click a validation error message, and select **Show Message**.

Proofing Your First Document

Oxygen XML Editor includes an *automatic (as-you-type) spell checking feature*, as well as a manual spell checking action. To check the spelling of your document manually, use the **Check Spelling** action on the toolbar or from the **Edit** menu.

Transforming Your First Document

An XML document must be transformed to be published. Transformations are specific to the particular type of document you have created. For example, a DITA transformation cannot be used on a DocBook file, or vice versa. A single document type may have many multiple transformations that produce different kinds of outputs. For some document types, such a DITA, many different content files may be combined together by a transformation. You need to locate and launch a transformation that is appropriate for your document type and the kind of output you want to generate.

Oxygen XML Editor uses *transformation scenarios* to control the transformation process. Depending on the document type you have created, there may be several transformation scenarios already configured for your use. This may include the default transformation scenarios supplied by Oxygen XML Editor or ones created by your organization.

To see the list of transformations available for your document, select the **Apply Transformation Scenario(s)** action from the toolbar or the **Document** > **Transformation** menu. A list of available transformation scenarios are displayed. Choose one or more scenarios to apply, and click **Apply associated**. Exactly how your transformed content appears depends on how the transformation scenario is configured.

Your First DITA Topic

To create your first DITA topic, select **File** > **New** or click the **New** button on the toolbar. The ***New Document Wizard*** is displayed:

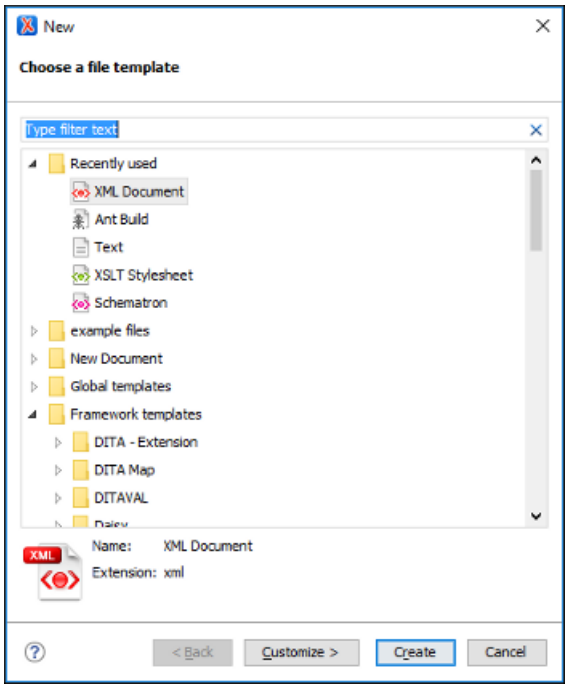


Figure 5: New Document Wizard

Go to **Framework templates** > **DITA** > **topic** and select the type of topic that you want to create.

Note: If your organization has created DITA customizations, the appropriate template files may be in another location, and various types of topics may be provided for your use. Check with the person who manages your DITA system to see if you should be using templates from another directory.

Your DITA topic is an XML document, thus all *the editing features that Oxygen XML Editor provides for editing XML documents* also apply to DITA topics. Oxygen XML Editor also provides extensive additional support for *editing DITA topics*, *their associated DITA maps*, and for *creating DITA output*.

Understanding DITA Topics

It is important to understand the role that a *DITA topic plays in a DITA project*. A DITA topic is not associated with a single published document. It is a separate entity that can potentially be included in many different books, help systems, or websites. Therefore, when you write a DITA topic you are not writing a book, a help system, or a website. You are writing an individual piece of content. This affects how you approach the writing task and how Oxygen XML Editor works to support you as you write.

Most of your topics are actually related to other topics, and those relationships can affect how you write and handle things such as links and content reuse. Oxygen XML Editor helps you manage those relationships. Depending on how your topics are related, you can use the tools provided in Oxygen XML Editor, along with the features of DITA, in a variety of ways.

Role of Maps

The basic method that DITA uses to express the relationship between topics is through a *DITA map*. Other relationships between topics, such as cross references, generally need to be made between topics in the same map. DITA uses maps to determine which topics are part of any output that you create. While customized DITA solutions can use other mechanisms, generally DITA is not used as a way to publish individual topics. Output is created from a map and includes all the topics referenced by the map.

A publication is not always represented by a single map. For instance, if you are writing a book, you might use a map to create each chapter and then organize the chapters in another map to create the book. If you are writing help topics, you might use a map to combine several DITA topics to create a single help topic and then create another map to organize your help topics in a help system. This allows you to reuse the content of a map in multiple projects.

Creating a Map

To add topics to a map, you must first *create the map*. A map is an XML document, similar to a topic. To create a map, select **File** > **New** or click the **New** button on the toolbar, go to **Framework templates** > **DITA Map** > **map** and select the type of map you want to create. Oxygen XML Editor asks if you want to open your map in the editor or in the ***DITA Maps Manager***. Usually, opening it in the **DITA Maps Manager** is the best choice, but you can also open the map in the editor from the **DITA Maps Manager**. The **DITA Maps Manager** presents a view of the *DITA map* that is similar to a table of contents.

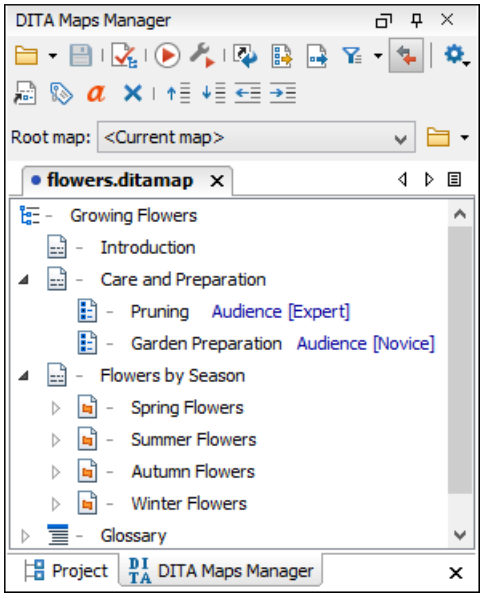


Figure 6: DITA Maps Manager View

Adding Topics to a Map

To *add a topic to a map*, add a topic reference to the map using a `topicref` element. The easiest way to do this is to open the topic in the editor, then right-click the *DITA map* in the ***DITA Maps Manager*** *view* and choose **Reference to the currently edited file** from the **Append Child**, **Insert Before**, or **Insert After** submenu. This opens the ***Insert Reference*** *dialog box* with all of the required fields already filled in for you. You can fill in additional information in the various tabs in this dialog box or add it to the map later. When you select **Insert and close**, a reference to your topic is added to the map.

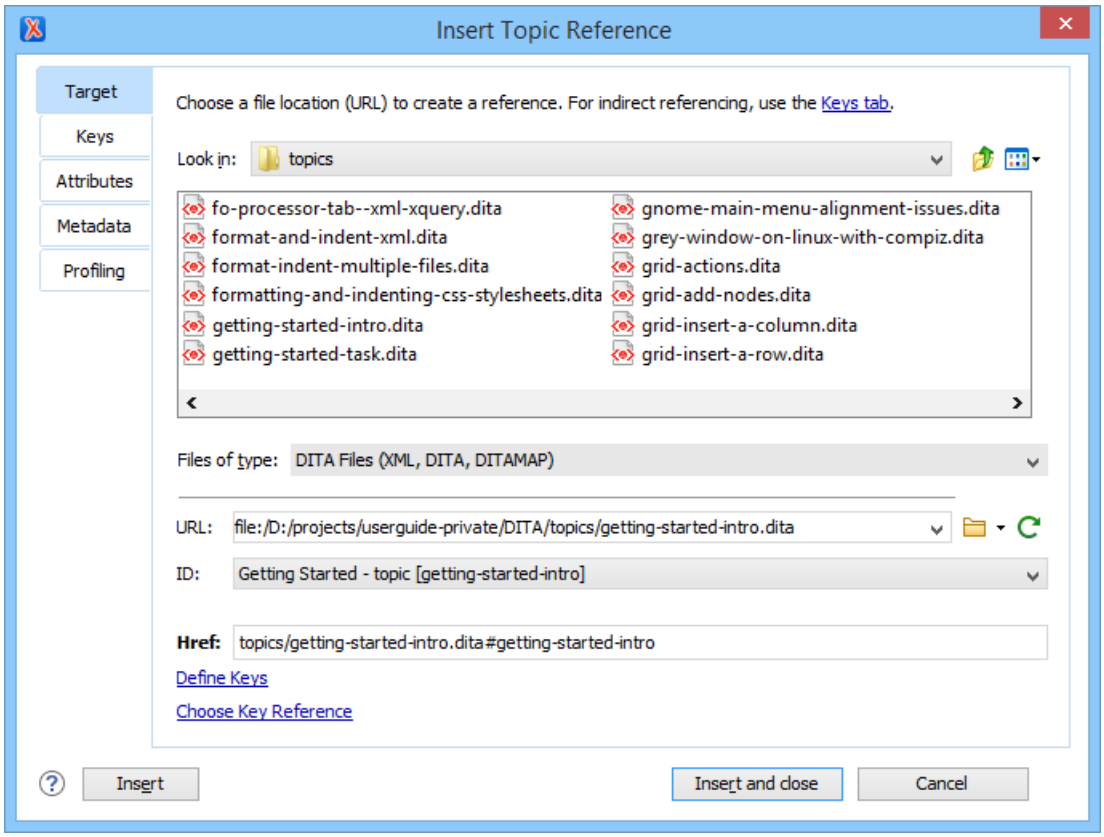


Figure 7: Insert Reference Dialog Box

If you want to see what the resulting map looks like in XML, save your map and then double-click the *DITA map* in the ***DITA Maps Manager*** *view*. The XML version of the map opens in the editor.

As you add topics to your map, you may want to make a topic the child or sibling of another topic. This is usually done at the map level. To create a child topic reference, right-click the parent topic in the **DITA Maps Manager** view and choose **Append Child**. To create a sibling topic reference, right-click a topic in the **DITA Maps Manager** view and choose **Insert After** or **Insert After**. From any of these submenus you can then choose one of the following options:

- **New** - Opens the ***New*** *file wizard* for creating a new topic.
- **Reference** - Opens the ***Insert Reference*** *dialog box* that allows you to create a reference to an existing topic.
- **Reference to the currently edited file** - Opens the ***Insert Reference*** *dialog box* that helps you to easily create a reference to the file that is currently opened in the editor.

You can also change the order and nesting of topics in the **DITA Maps Manager** view by doing either of the following:

- Select the topic to move while holding down the **Alt** key and use the arrow keys to move it around.
- Use the mouse to drag and drop the topic to the desired location.

The way your parent and child topics are organized in any particular output depends on both the configuration of those topics in the map and the rules of the output transformation that is applied to them. Do not assume that your topics must have the same organization for all output types. The map defines the organization of the topics, not the topics themselves. It is possible to create a variety of maps, each with different organization and configuration options to produce a variety of outputs.

Child Maps

If you have a large set of information, such as a long book or extensive help system, a single map can become long and difficult to manage. To make it easier to manage, you can *break up the content into smaller submaps*. A submap might represent a chapter of a book, a section of a user manual, or a page on a website.

To build a publication out of these smaller maps, you must add them to a map that represents the overall publication. To add a child map to the current map, right-click the parent *DITA map* and choose **Append child** > **Map reference**.

Validating a Map

Just as it is with your individual topics, it is important to *validate your maps*. Oxygen XML Editor provides a validation function for *DITA maps* that does more than simply validating that the XML is well formed. It also does the following:

- Validates all of the relationships defined in the maps.
- Validates all of the files that are included in the map.
- Validates all of the links that are expressed in the files.

Validating the map that describes your entire publication validates all the files that make up the publication and all of the relationships between them. To validate a map, click the **Validate and Check for Completeness** button in the ***DITA Maps Manager*** *view*.

Publishing Your Topics

As noted previously, in DITA standards you usually do not publish output from an individual topic. Instead, you *create published output* by running a DITA transformation on a map. This collects all the topics that are referenced in the map, organizes them, and produces output in a particular format. By default, Oxygen XML Editor uses the transformations provided by the **DITA Open Toolkit** for publishing to various output formats (such as PDF, WebHelp or EPUB). Your organization may have created various custom transformations or modified the built-in **DITA Open Toolkit** transformations. In either case, Oxygen XML Editor manages them by using transformation scenarios.

To publish output for a map, select the transformation scenario you want to run and set any of the parameters it requires. To select a transformation, click the **Configure Transformation Scenario(s)** button in the ***DITA Maps Manager*** *view*. This opens the ***Configure Transformation Scenario(s)*** *dialog box*.

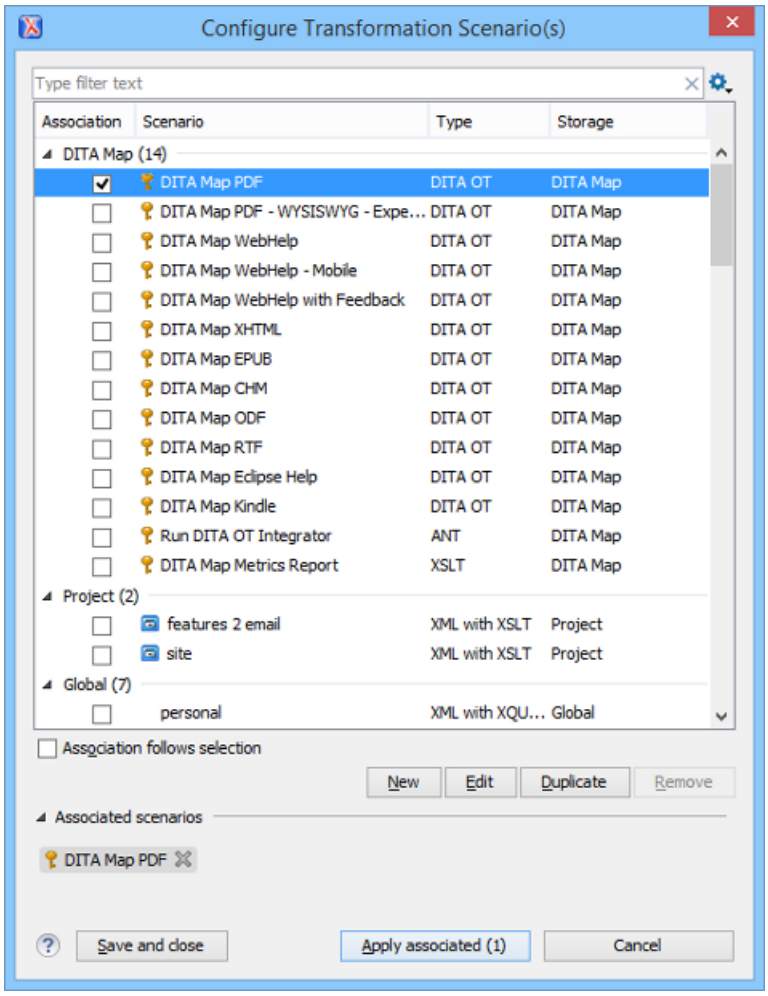


Figure 8: Configure Transformation Scenarios Dialog Box

Choose the transformation scenarios you want to apply and click **Apply associated**. Depending on the configuration of the transformation scenario, when the transformation is finished, your output may automatically be opened in the appropriate application. To change or view the configuration or storage options for a transformation scenario, select the transformation and click **Edit**.

Related Information:

DITA Authoring and Publishing on page 1580
Editing XML Documents in Author Mode on page 329

Creating a New Project

Oxygen XML Editor allows you to organize your XML-related files into projects. This helps you manage and organize your files and also allows you to perform batch operations (such as validation and transformation) over multiple files. You can also *share your project settings and transformation/validation scenarios* with other users. Use the ***Project** view* to manage projects, and the files and folders contained within.

Creating a New Project

To create a new project, select **New Project** from the **Project** menu, the **New** menu in the contextual menu, or the drop-down menu at the top-left of the **Project** view. This opens a dialog box that allows you to assign a name to the new project and adds it to the structure of the project in the ***Project** view*.

Adding Items to the Project

To add items to the project, select any of the following actions that are available when invoking the contextual menu in the **Project** view:

New > File

Opens a **New** file dialog box that helps you create a new file and adds it to the project structure.

New > Folder

Opens a **New Folder** dialog box that allows you to specify a name for a new folder and adds it to the structure of the project.

The project itself is considered a logical folder. You can add a logical folder, or content to a logical folder, by using one of the following actions that are available in the contextual menu, when invoked from the *project root*:

New > Logical Folder

Creates a logical folder in the tree structure (the icon is a magenta folder on Mac OS X -).

New > Logical Folders from Web

Replicates the structure of a remote folder accessible over FTP/SFTP/WebDAV, as a structure of logical folders. The newly created logical folders contain the file structure of the folder it points to.

Add Folder

Adds a link to a physical folder, whose name and content mirror a real folder that exists in the local file system (the icon of this action is different on Mac OS X).

Add Files

Adds links to files on the local file system.

Add Edited File

Adds a link to the currently edited file in the project.

Using Linked Folders (Shortcuts)

Another easy way to organize your XML working files is to place them in a directory and then to create a corresponding linked folder in you project. If you add new files to that folder, you can simply use the **Refresh (F5)** action from the toolbar or contextual menu and the ***Project*** *view* will display the existing files and subdirectories. If your files are scattered amongst several folders, but represent the same class of files, you might find it useful to combine them in a logical folder.

You can create linked folders (shortcuts) by dragging and dropping folders from the Windows Explorer (Mac OS X Finder) to the project tree, or by selecting **Add Folder** in the contextual menu from the *project root*. Linked folders are displayed in the ***Project*** *view* with bold text. To create a file inside a linked folder, select the **New** > **File** action from the contextual menu. The linked files presented in the ***Project*** *view* are marked with a special icon.

Note: Files may have multiple instances within the folder system, but cannot appear twice within the same folder.

For more information on managing projects and their content, see *Project View* on page 188.

For more details about how you can share projects with other users, see *Sharing a Project - Team Collaboration* on page 284.

Related Information:

Using Projects to Group Documents on page 275

Getting Help

If you run into specific problems while using Oxygen XML Editor you can take advantage of a variety of support related resources. Those resources include the following:

- *The Oxygen XML Editor Support Section of the Website*
- *The Oxygen XML Editor Forum*
- *The Oxygen XML Editor Video Tutorials*

- *The Common Problems and Solutions Section of the User Manual*
- *The Online Technical Support Form*

The application also includes various specific help-related resources in the **Help** menu.

Help Menu

The Oxygen XML Editor **Help** menu provides various resources to assist you with your tasks.

This menu includes the following actions or options:

Welcome

This option opens the **Welcome** screen that includes some resources to assist you with using Oxygen XML Editor.

Help (F1)

Use this action (or the **F1** key) to open a dialog box that presents a section in the User Manual that is appropriate for the context of the current cursor position. If the **Use online help** option is selected, this action will open the User Manual in an online mode.

Use online help

If this option is selected, the **Help (F1)** action will open the Oxygen XML Editor User Manual in an online mode.

Show Dynamic Help view

Use this action to open a view that loads the latest online WebHelp version of the Oxygen XML Editor User Manual, and dynamically opens a topic that is relevant to the focused editor, view, or dialog box. It requires Java 1.8 and an online connection. In Windows, if a Java 1.8 version is not detected, you will be advised to upgrade, while in Linux and Mac OS X with Java 1.7 and lower, Oxygen XML Editor will attempt to load an offline version of the documentation. In all three operating systems, with Java 1.8, if an online connection is not detected, you will receive an error message advising you to check your proxy settings.

You can also open the **Dynamic Help** view by selecting it from the **Window** > **Show View** menu.

Install new add-ons

Opens a dialog box that allows you to install new *add-ons* to extend the functionality of Oxygen XML Editor.

Check for add-ons updates

Opens a dialog box that allows you to check for updates on installed *add-ons*.

Manage add-ons

Opens a dialog box that allows you to manage installed *add-ons*.

Check for a New Version

Use this action to view information about the latest version of Oxygen XML Editor.

Browse Oxygen Website

Opens the Oxygen XML Editor website in your default internet browser.

Register

If you encounter problems with your Oxygen XML Editor license, you can use this option to open a dialog box that provides options for obtaining or using a license key.

Lock/Unlock floating license

If you are using a *Floating License*, you can lock it so that it does not get *released to the pool* unless you or the system administrator unlocks it.

Report problem

You can use this option to open a dialog box that allows you to write the description of a problem that was encountered while using the application. You can also select additional information to be sent to the technical support team in the five tabs:

- **General info** - You can edit your contact details in case you want to be contacted for further details or to be notified of a resolution.
- **Class Loader URLs** - You can choose whether or not to include the listed *Class Loader URLs* with your report.

- **System properties** - You can choose whether or not to include the listed system property details with your report.

  Tip: You are able to change the URL where the reported problem is sent by using the *com.oxygenxml.report.problems.url* system property. The report is sent in XML format through the `report` parameter of the POST HTTP method.
- **Plugins** - You can choose whether or not to include details about your installed *plugins* with your report.
- **Frameworks** - You can choose whether or not to include details about your installed *frameworks* with your report.

Support Center

Use this option to open the *Oxygen XML Editor Support Section of the Website*.

Support Tools > Clipboard Inspector

Opens a dialog box that displays extensive details of all the transferable objects from the clipboard. This is helpful if you experience problems while copying content from other applications and pasting it into Oxygen XML Editor. You can use the **Copy** button to copy all of this data and then paste it into an email to be sent to the Oxygen support team.

Support Tools > Randomize XML text content

Use this action when you need to send samples to the Oxygen support team and you want to keep the text content confidential. It opens a dialog box that allows you to select the resources for which the text content will be randomized. You can then save the resources and send them to the Oxygen support team without fear of compromising sensitive or private data. For more information, see *Randomize XML Text Content* on page 17.

Warning: Before using this action, it is highly recommended that you copy the XML resources to be processed, save them in a separate folder, and then process this operation on the copies instead of the original files. Otherwise, you may lose your original content.

Tip of the Day

Opens a dialog box that offers tips for using Oxygen XML Editor.

About

Use this option to open a dialog box that contains information about Oxygen XML Editor and the installed version. This dialog box includes the following tabs:

- **Copyright** - This tab contains general information about the product and the version of the product you are using, along with contact details and the *SGN* number. Details regarding the memory usage are also presented at the bottom of the dialog box.
- **Libraries** - This tab presents the list of third party libraries that Oxygen XML Editor uses. To view the End User Licence Agreement of each library, double-click it.
- **Frameworks** - This tab contains a list with the *framework* that are bundled with Oxygen XML Editor.
- **System Properties** - This tab contains a list with system properties and their values. The contextual menu allows you to select and copy the properties.

Related Information:

Details to Submit in a Request for Technical Support Using the Online Form on page 1569

Randomize XML Text Content

Oxygen XML Editor includes an action that randomizes the text content of an XML document. This action is available in the **Help** > **Support Tools** menu. It is helpful if you need to send XML samples to the Oxygen support team and you want to keep the text content confidential. It opens a dialog box that allows you to select the resources for which the text content will be randomized. You can then save the resources and send them to the Oxygen support team without fear of compromising sensitive or private data.

Warning: Before using this action, it is highly recommended that you copy the XML resources to be processed, save them in a separate folder, and then perform this operation on the copies instead of the original files. Otherwise, you may lose your original content.

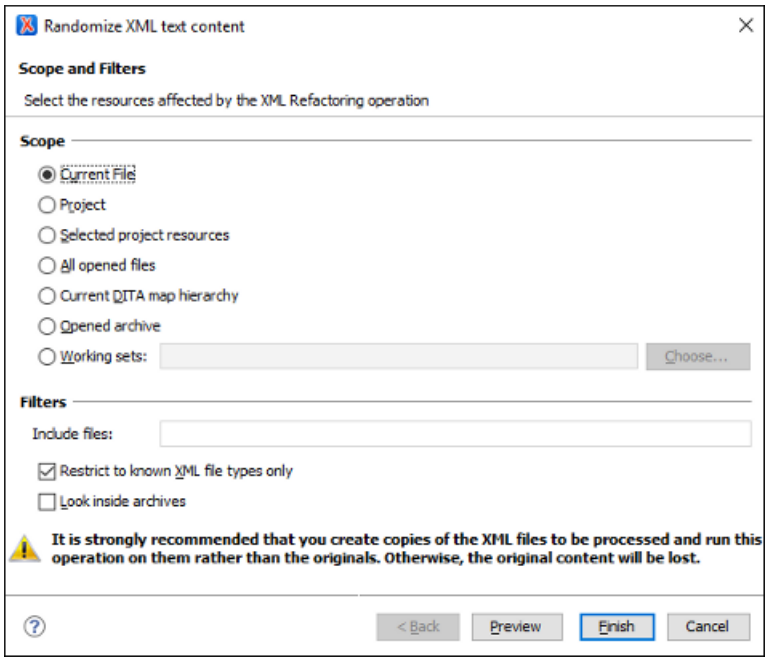


Figure 9: Randomize XML Text Content Dialog Box

The **Randomize XML Text Content** dialog box includes the following options:

Scope

Allows you to select the set of files whose text content will be randomized by the operation. You can select from predefined resource sets (such as the current file, your whole project, the current *DITA map* hierarchy for DITA projects, etc.) or you can define your own set of resources by creating a *working set*.

Filters

This section includes the following options:

- **Include files** - Allows you to filter the selected resources by using a file pattern. For example, to restrict the operation to only analyze build files you could use `build*.xml` for the file pattern.
- **Restrict only to known XML file types** - When selected, only resources with a known XML file type will be affected by the operation.
- **Look inside archives** - When selected, the resources inside archives will also be affected.

Frequently Used Shortcut Keys

Oxygen XML Editor includes numerous shortcut keys that are assigned to actions to help you edit content. All the shortcuts that are assigned to actions are displayed in the table in the ***Menu Shortcut Keys*** *preference page*.

For information about how to assign or configure shortcut keys, see *How to Assign a Shortcut Key or Edit an Existing Shortcut* on page 153.

Table 1: Frequently Used Shortcut Keys in Oxygen XML Editor

Action	Windows/Linux Shortcut Keys	Mac/OS X Shortcut Keys	Description of Default Assigned Action
Attribute Editor	**Alt + Enter**	**Alt + Enter**	• Opens the in-place attribute editor
Beginning	**Ctrl + Home**	**Command + Home**	• Navigates to the beginning of the document

Action	Windows/Linux Shortcut Keys	Mac/OS X Shortcut Keys	Description of Default Assigned Action
Check Spelling	**F7**	**F7**	• Opens the spell checking dialog box
Check Well-Formedness	**Ctrl + Shift + W**	**Command + Shift + W**	• Check well-formedness of current document
Configure Transformation	**Ctrl + Shift + C**	**Command + Shift + C**	• Opens the Configure Transformation Scenario dialog box
Content Completion / New Line	**Enter**	**Enter**	• **Author** mode - Opens the content completion window • **Text** mode - Moves cursor to the next line
Content Completion (Text Mode)	**Ctrl + Space**	**Command + Space**	• **Text** mode - Opens the content completion window
Create Bookmark #	**Ctrl + Shift + 1-9**	**Command + Shift + 1-9**	• Create bookmarks numbered 1 through 9
Create Next Bookmark	**F9**	**F9**	• Create bookmark numbered whatever is next in sequence
Delete Next Word	**Ctrl + Delete**	**Command + Delete**	• Deletes the next word or whitespace
Delete Previous Word	**Ctrl + Backspace**	**Command + Backspace**	• Deletes the previous word or whitespace
Delete Tags	**Alt + Shift + X**	**Command + Alt + X**	• Deletes the start and end tag of the current element.
End	**Ctrl + End**	**Command + End**	• Navigates to the end of the document
Exit	**Ctrl + Q**	**Command + Q**	• Exit the application
Find	**Ctrl + F**	**Command + F**	• Opens Find/ Replace dialog box
Find Next	**F3**	**Command + G**	• Finds next occurrence of the last searched term
Find Previous	**Shift + F3**	**Command + Shift + G**	• Finds previous occurrence of the last searched term
Go To Bookmark	**Ctrl + 1-9**	**Command + 1-9**	• Go to specific bookmark

Action	Windows/Linux Shortcut Keys	Mac/OS X Shortcut Keys	Description of Default Assigned Action
Help	**F1**	**F1**	• Opens help documentation
Insert Para / Format Indent	**Ctrl + Shift + P**	**Command + Shift + P**	• **Author** mode - Inserts a paragraph at cursor position • **Text** mode - Formats and indents current document
Move Tab Left	**Ctrl + Alt + Comma**	**Ctrl + Alt + Comma**	• Moves the current file tab one position to the left
Move Tab Right	**Ctrl + Alt + Period**	**Ctrl + Alt + Period**	• Moves the current file tab one position to the right
Move Node Down (Author)	**Alt + DownArrow**	**Alt + DownArrow**	• Moves the selected XML node down in **Author** mode
Move Node Down (Text)	**Ctrl + Alt + DownArrow**	**Command + Alt + DownArrow**	• Moves the selected XML node down in **Text** mode
Move Node Up (Author)	**Alt + UpArrow**	**Alt + UpArrow**	• Moves the selected XML node up in **Author** mode.
Move Node Up (Text)	**Ctrl + Alt + UpArrow**	**Command + Alt + UpArrow**	• Moves the selected XML node up in **Text** mode
New File	**Ctrl + N**	**Command + N**	• Opens wizard for creating new documents
Next Word	**Ctrl + RightArrow**	**Command + RightArrow**	• Navigates to next word
Open/Find Resource	**Ctrl + Shift + R**	**Command + Shift + R**	• Opens the Open/Find Resource dialog box
Previous Word	**Ctrl + LeftArrow**	**Command + LeftArrow**	• Navigates to previous word
Print Preview	**Ctrl + P**	**Command + P**	• Opens the print preview (page setup) dialog box
Quick Assist	**Alt + 1**	**Command + Alt + 1**	• Opens Quick Assist menu if actions are available in the current context (usually indicated with a bulb 💡 icon in the left stripe)
Quick Find	**Alt + Shift + F**	**Alt + Shift + F**	• Opens the Quick Find mechanism at the bottom of the editor

Action	Windows/Linux Shortcut Keys	Mac/OS X Shortcut Keys	Description of Default Assigned Action
Redo	**Ctrl + Y** (Windows) - **Ctrl + Shift + Z** (Linux)	**Command + Shift + Z**	• Redo last editing action
Refresh	**F5**	**F5**	• Refresh
Remove Bookmarks	**Ctrl + F7**	**Command + F7**	• Removes all bookmarks
Reopen Last Closed Editor	**Ctrl + Alt + T**	**Command + Alt + T**	• Reopens the editor tab that was closed most recently
Reset Zoom	**Ctrl + NumPad0**	**Command + NumPad0**	• Resets zoom (default font size)
Save	**Ctrl + S**	**Command + S**	• Saves current document
Save All	**Ctrl + Shift + S**	**Command + Shift + S**	• Saves all opened files
Scroll Down	**Ctrl + DownArrow**	**Command + DownArrow**	• Scrolls the editor down
Scroll Up	**Ctrl + UpArrow**	**Command + Up Arrow**	• Scrolls the editor up
Shift Left	**Shift + Tab**	**Shift + Tab**	• **Author** mode - Moves cursor to the previous XML node • **Text** mode - Shifts content to the left
Shift Right	**Tab**	**Tab**	• **Author** mode - Moves cursor to the next XML node • **Text** mode - Shifts content to the right
Split Element	**Alt + Shift + D**	**Ctrl + Alt + D**	• Splits the element the cursor position
Surround With	**Ctrl + E**	**Command + E**	• Surrounds selected content with specified tag
Switch Tabs	**Ctrl + Tab**	**Command + Tab**	• Switches between opened tabs
Transform	**Ctrl + Shift + T**	**Command + Shift + T**	• Opens a dialog box for selecting a transformation scenario
Underline / Open URL	**Ctrl + U**	**Command + U**	• Underlines select content (in main editor) • Opens URL (when focus is outside the main editor)
Undo	**Ctrl + Z**	**Command + Z**	• Undo last editing action

Action	Windows/Linux Shortcut Keys	Mac/OS X Shortcut Keys	Description of Default Assigned Action
Validate	**Ctrl + Shift + V**	**Command + Shift + V**	• Validates current document
Zoom In	**Ctrl + NumPad+**	**Command + NumPad+**	• Zooms in (increase font size)
Zoom Out	**Ctrl + NumPad-**	**Command + NumPad-**	• Zooms out (decrease font size)

Installation

3

Topics:

- *Installation Options for Oxygen XML Editor*
- *Install Oxygen XML Editor on Windows*
- *Install Oxygen XML Editor on Mac OS X*
- *Install Oxygen XML Editor on Linux*
- *Installing Oxygen XML Editor on Windows Server*
- *Installing Oxygen XML Editor on a Linux / UNIX Server*
- *Installing Oxygen XML Editor using the Java Web Start (JWS) Installer*
- *Group Deployment*
- *Obtaining and Registering a License Key for Oxygen XML Editor*
- *Setting Up a Floating License Server*
- *Transferring a License Key*
- *Upgrading Oxygen XML Editor*
- *Installing and Updating Add-ons in Oxygen XML Editor*
- *Uninstalling Oxygen XML Editor*
- *Oxygen XML Editor Installer Command Line Reference*

This chapter includes information about installing and licensing Oxygen XML Editor on various platforms. Oxygen XML Editor is available on Windows, Linux, and Mac OS X and there are a variety of methods and options for installing and running Oxygen XML Editor on your system or server. This section also includes information about registering, transferring, or releasing licenses, upgrading, installing *add-ons*, and uninstalling.

Installation Options for Oxygen XML Editor

Choosing how Oxygen XML Editor runs

You can install Oxygen XML Editor to run in a number of ways:

- As a desktop application (running standalone or as an Eclipse plugin) on *Windows*, *Linux*, or *Mac*.
- As a desktop application (running standalone or as an Eclipse plugin) on a *Unix or Linux server* or on *Windows Terminal Server*.
- From within a browser though the *Java Web Start* technology.

Choosing an installer

You have a choice of installers;

- The native installer for your platform. On Windows and Linux, the native installer can run also in unattended mode.

The installation packages were checked before publication with an antivirus program to make sure they are not infected with viruses, trojan horses, or other malicious software.

Choosing a license option

You must *obtain and register a license* key to run Oxygen XML Editor.

You can choose from two kinds of license:

- A named-person license, which can be used by a single person on multiple computers.
- A floating license, which can be used by different people at different times. Only one person can use a floating license at a time.

Upgrading, transferring, and uninstalling.

You can also *upgrade* Oxygen XML Editor, *transfer a license*, or *uninstall* Oxygen XML Editor.

Getting help with installation

If you need help at any point during these procedures, please send us an email at support@oxygenxml.com.

Install Oxygen XML Editor on Windows

Choosing an Installer

You can install Oxygen XML Editor on Windows using one of the following methods:

- *Windows installer*
- *Windows installer in unattended mode*

System Requirements

System requirements for a Windows install:

Operating systems

Windows Vista, Windows 7, Windows 8, Windows 10, Windows Server 2008, Windows Server 2012

CPU

- Minimum - Intel Pentium III™/AMD Athlon™ class processor, 1 GHz
- Recommended - Dual Core class processor

Memory

- Minimum - 2 GB of RAM
- Recommended - 4 GB of RAM

Storage

- Minimum - 400 MB free disk space
- Recommended - 1 GB free disk space

Java

Oxygen XML Editor requires Java. If you use the native Windows installer, Oxygen XML Editor will be installed with its own copy of Java. If you use the all platforms installer, your system must have a compatible Java virtual machine installed.

Oxygen XML Editor only supports official and stable Java Virtual Machines with the version number 1.6.0 or later (the recommended version is 1.7) from Oracle available at *http://www.oracle.com/technetwork/java/javase/downloads/index.html*. Oxygen XML Editor may work with JVM implementations from other vendors, but there is no guarantee that those implementations will work with future Oxygen XML Editor updates and releases.

Oxygen XML Editor uses the following rules to determine which installed version of Java to use:

1. If you install using the native Windows installer, which installs a version of Java as part of the Oxygen XML Editor installation, the version in the `jre` subdirectory of the installation directory is used.
2. Otherwise, if the Windows environment variable `JAVA_HOME` is set, Oxygen XML Editor uses the Java version pointed to by this variable.
3. Otherwise, the version of Java pointed to by your PATH environment variable is used.

If you run Oxygen XML Editor using the batch file, `oxygen.bat`, you can edit the batch file to specify a particular version to use.

Install Using the Windows Installer

To install Oxygen XML Editor using the Windows installer, follow these steps:

1. Make sure that your system meets the *system requirements*.
2. Download the Windows installer.
3. Validate the integrity of the downloaded file by *checking it against the MD5 sum* published on the download page.
4. Run the installer and follow the instructions in the installation program.
5. Start Oxygen XML Editor using one of the following methods:
   - Using one of the shortcuts created by the installer.
   - By running `oxygen.bat`, which is located in the install folder.
6. To license your copy of Oxygen XML Editor go to **Help** > **Register** and enter your *license information*.

Unattended Installation

You can run the installation in unattended mode by running the installer from the command line with the -q parameter. By default, running the installer in unattended mode installs Oxygen XML Editor with the default options and does not overwrite existing files. You can change many options for the unattended installer using the *installer command line parameters*.

Install Oxygen XML Editor on Mac OS X

Choosing an Installer

You can install Oxygen XML Editor on Mac OS X using one of the following methods:

- Install using the Mac OS X installation package, oxygen.dmg.

System Requirements

System requirements for a Mac OS X install:

Operating system

OS X version 10.8 64-bit or later

CPU

- Minimum - Intel-based Mac, 1 GHz
- Recommended - Dual Core class processor

Memory

- Minimum - 2 GB of RAM
- Recommended - 4 GB of RAM

Storage

- Minimum - 400 MB free disk space
- Recommended - 1 GB free disk space

Java

Oxygen XML Editor requires Java to run. OS X includes Java by default or it will install it on the first attempt to run a Java application.

Oxygen XML Editor only supports official and stable Java Virtual Machines with the version number 1.6.0 or later (the recommended version is 1.6.0 from Apple). Oxygen XML Editor may work with JVM

implementations from other vendors, but there is no guarantee that other implementations will work with future Oxygen XML Editor updates and releases.

Oxygen XML Editor uses the following rules to determine which installed version of Java to use:

1. If you start Oxygen XML Editor with the application launcher (.app) file then:
   a. If you use the zip distribution for OS X Oxygen XML Editor uses the Apple Java SE 6 available on your Mac computer
   b. If you use the tar.gz distribution that contains a bundled JRE then Oxygen XML Editor will use that bundled JRE
2. If you start Oxygen XML Editor using a startup .sh script then:
   a. If a bundled JRE is available then it will be used
   b. Otherwise, if the `JAVA_HOME` environment variable is set then the Java distribution indicated by it will be used
   c. Otherwise, the version of Java pointed to by your `PATH` environment variable will be used

If you run Oxygen XML Editor using the `oxygen.sh` script, you can change the version of Java used by editing to script file. Go to the Java command at the end of the script file and specify the full path to the Java executable of the desired JVM version. For example:

```
/System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home/bin/java "-
Xdock:name= ...
```

OS X Installation

To install Oxygen XML Editor on OS X, follow these steps:

1. Download the OS X installation package (oxygen.dmg).
2. Validate the integrity of the downloaded file by *checking it against the MD5 sum* published on the download page.
3. Double-click the oxygen.dmg disk image file to mount it.
4. Drag/Copy the *`Oxygen XML Editor`* folder to your `/Applications` folder (or another location if you wish).

   Do not copy the files/folders from within the *`Oxygen XML Editor`* folder (always copy the folder itself), otherwise you will omit invisible files/folders and the application may no longer start.

   Oxygen XML Editor is now installed.
5. Start Oxygen XML Editor, using one of the following methods:
   - Double-click `Oxygen XML Editor.app`.
   - Run `sh oxygenMac.sh` in the command line interface.

   Notice: You can start multiple instances on the same computer by running the following command for each new instance:

   ```
   open -n Oxygen.app
   ```
6. To license your copy of Oxygen XML Editor, go to **Help** > **Register** to enter your *license key*.

Install Oxygen XML Editor on Linux

Choosing an Installer

You can install Oxygen XML Editor on Linux using any of the following methods:

- Install using the Linux installer.
- Install using the Linux installer in unattended mode.

System Requirements

System requirements for a Linux install:

Operating system

Any Unix/Linux distribution with an available Java SE Runtime Environment version 1.6.0 or later from Oracle

CPU

- Minimum - Intel Pentium III™/AMD Athlon™ class processor, 1 GHz
- Recommended - Dual Core class processor

Memory

- Minimum - 2 GB of RAM
- Recommended - 4 GB of RAM

Storage

- Minimum - 400 MB free disk space
- Recommended - 1 GB free disk space

Java

Oxygen XML Editor requires Java. Oxygen XML Editor only supports official and stable Java Virtual Machines with the version number 1.6.0 or later (the recommended version is 1.6.0) from Oracle available at *http://www.oracle.com/technetwork/java/javase/downloads/index.html*. Oxygen XML Editor may work with JVM implementations from other vendors, but there is no guarantee that other implementations will work with future Oxygen XML Editor updates and releases. Oxygen XML Editor does not work with the GNU libgcj Java Virtual Machine.

Oxygen XML Editor uses the following rules to determine which installed version of Java to use:

1. If you used the Linux installer, which installs a version of Java as part of the Oxygen XML Editor installation, the version in the `jre` subdirectory of the installation directory is used.
2. Otherwise, if the Linux environment variable `JAVA_HOME` is set, Oxygen XML Editor uses the Java version pointed to by this variable.
3. Otherwise the version of Java pointed to by your PATH environment variable is used.

You can also change the version of the Java Virtual Machine that runs Oxygen XML Editor by editing the script file, `oxygen.sh`. Go to the Java command at the end of the script file and specify the full path to the Java executable of the desired JVM version. For example:

```
/usr/bin/jre1.6.0_45/bin/java -Xmx256m ...
```

Linux Installation

Linux installation procedure.

To install Oxygen XML Editor on Linux, follow these steps:

1. Download the Linux installer.
2. Optionally, you can validate the integrity of the downloaded file by *checking it against the MD5 sum* published on the download page.
3. Run the installer that you downloaded.

   For example, open a shell, `cd` to the installation directory, and at the prompt type `sh ./oxygen-32bit.sh` or `sh ./oxygen-64bit.sh`, depending on which installer you downloaded.
4. Start Oxygen XML Editor using one of the following methods:
   - Use the `oxygen` shortcut created by the installer.
   - From a command line, type `sh oxygen.sh`. This file is located in the installation folder.
5. To license your copy of Oxygen XML Editor go to **Help** > **Register** and enter your *license key*.

Unattended Installation

You can run the installation in unattended mode by running the installer from the command line with the -q parameter. By default, running the installer in unattended mode installs Oxygen XML Editor with the default

options and does not overwrite existing files. You can change many options for the unattended installer using the *installer command line parameters*.

Installing Oxygen XML Editor on Windows Server

Choosing an Installer

You can install Oxygen XML Editor on Windows using one of the following methods:

- *Windows installer*
- *Windows installer in unattended mode*

System Requirements

System requirements for a Windows Server install:

Operating systems

Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2

CPU

- Minimum - Intel Pentium III™/AMD Athlon™ class processor, 1 GHz
- Recommended - Dual Core class processor

Memory

- Minimum values per user - 512 MB of RAM
- Recommended values per user - 2 GB of RAM

Storage

- Minimum - 400 MB free disk space
- Recommended - 1 GB free disk space

Java

Oxygen XML Editor requires Java. If you use the native Windows installer, Oxygen XML Editor will be installed with its own copy of Java. If you use the all platforms installer, your system must have a compatible Java virtual machine installed.

Oxygen XML Editor only supports official and stable Java Virtual Machines with the version number 1.6.0 or later (the recommended version is 1.7) from Oracle available at *http://www.oracle.com/technetwork/java/javase/downloads/index.html*. Oxygen XML Editor may work with JVM implementations from other vendors, but there is no guarantee that those implementations will work with future Oxygen XML Editor updates and releases.

Oxygen XML Editor uses the following rules to determine which installed version of Java to use:

1. If you install using the native Windows installer, which installs a version of Java as part of the Oxygen XML Editor installation, the version in the `jre` subdirectory of the installation directory is used.
2. Otherwise, if the Windows environment variable `JAVA_HOME` is set, Oxygen XML Editor uses the Java version pointed to by this variable.
3. Otherwise, the version of Java pointed to by your PATH environment variable is used.

If you run Oxygen XML Editor using the batch file, `oxygen.bat`, you can edit the batch file to specify a particular version to use.

Install Using the Windows Installer

To install Oxygen XML Editor using the Windows installer, follow these steps:

1. Make sure that your system meets the *system requirements*.
2. Download the Windows installer.

3. Validate the integrity of the downloaded file by *checking it against the MD5 sum* published on the download page.
4. Run the installer and follow the instructions in the installation program.
5. Start Oxygen XML Editor using one of the following methods:
   - Using one of the shortcuts created by the installer.
   - By running `oxygen.bat`, which is located in the install folder.
6. To license your copy of Oxygen XML Editor go to **Help** > **Register** and enter your *license information*.

Configuring Windows Terminal Server

Windows Terminal Server configuration procedure.

1. Install Oxygen XML Editor on the server and make its shortcuts available to all users.
2. If you need to run multiple instances of Oxygen XML Editor, make sure you add the *-Dcom.oxygenxml.MultipleInstances=true* parameter in the `.bat` startup script.
3. Make sure you allocate sufficient memory to Oxygen XML Editor by adding the `-Xmx` parameter either in *the `.bat` startup script*, or in *the `.vmoptions` configuration file* (if you start it from an executable launcher).

Installing Oxygen XML Editor on a Linux / UNIX Server

Choosing an Installer

You can install Oxygen XML Editor on Linux using any of the following methods:

- Install using the Linux installer.
- Install using the Linux installer in unattended mode.

System Requirements

System requirements for a Linux install:

Operating system

Any Unix/Linux distribution with an available Java SE Runtime Environment version 1.6.0 or later from Oracle

CPU

- Minimum - Intel Pentium III™/AMD Athlon™ class processor, 1 GHz
- Recommended - Dual Core class processor

Memory

- Minimum - 2 GB of RAM
- Recommended - 4 GB of RAM

Storage

- Minimum - 400 MB free disk space
- Recommended - 1 GB free disk space

Java

Oxygen XML Editor requires Java. Oxygen XML Editor only supports official and stable Java Virtual Machines with the version number 1.6.0 or later (the recommended version is 1.6.0) from Oracle available at *http://www.oracle.com/technetwork/java/javase/downloads/index.html*. Oxygen XML Editor may work with JVM implementations from other vendors, but there is no guarantee that other implementations will work with future Oxygen XML Editor updates and releases. Oxygen XML Editor does not work with the GNU libgcj Java Virtual Machine.

Oxygen XML Editor uses the following rules to determine which installed version of Java to use:

1. If you used the Linux installer, which installs a version of Java as part of the Oxygen XML Editor installation, the version in the `jre` subdirectory of the installation directory is used.

2. Otherwise, if the Linux environment variable `JAVA_HOME` is set, Oxygen XML Editor uses the Java version pointed to by this variable.
3. Otherwise the version of Java pointed to by your PATH environment variable is used.

You can also change the version of the Java Virtual Machine that runs Oxygen XML Editor by editing the script file, `oxygen.sh`. Go to the Java command at the end of the script file and specify the full path to the Java executable of the desired JVM version. For example:

```
/usr/bin/jre1.6.0_45/bin/java -Xmx256m ...
```

Linux Installation

Linux installation procedure.

To install Oxygen XML Editor on Linux, follow these steps:

1. Download the Linux installer.
2. Optionally, you can validate the integrity of the downloaded file by *checking it against the MD5 sum* published on the download page.
3. Run the installer that you downloaded.

   For example, open a shell, `cd` to the installation directory, and at the prompt type `sh ./oxygen-32bit.sh` or `sh ./oxygen-64bit.sh`, depending on which installer you downloaded.
4. Start Oxygen XML Editor using one of the following methods:
   - Use the `oxygen` shortcut created by the installer.
   - From a command line, type `sh oxygen.sh`. This file is located in the installation folder.
5. To license your copy of Oxygen XML Editor go to **Help** > **Register** and enter your *license key*.

Unix / Linux Server Configuration

To install Oxygen XML Editor on a Unix / Linux server:

1. Install Oxygen XML Editor on the server and make sure the oxygen.sh script is executable and the installation directory is in the PATH of the users that need to use the application.
2. Make sure you allocate sufficient memory to Oxygen XML Editor by setting an appropriate value for the `-Xmx` parameter in the `.sh` startup script.

   The default value of the `-Xmx` parameter is 512 MB. To avoid *performance issues with large documents*, you may need to adjust it.
3. Make sure the X server processes located on the workstations allow connections from the server host. For this, use the xhost command.
4. Start telnet (or ssh) on the server host.
5. Start an *xterm* process with the **display** parameter set on the current workstation. For example: `xterm -display workstationip:0.0`.
6. Start Oxygen XML Editor by typing `sh oxygen.sh` from the command line. This file is located in the installation folder.

Installing Oxygen XML Editor using the Java Web Start (JWS) Installer

Oxygen XML Editor provides the tools to create your own JWS distribution that can be installed on a custom web server. The advantages of a JWS distribution include:

- Oxygen XML Editor is run locally, not inside a web browser, overcoming many of the browser compatibility problems common to applets.
- JWS ensures that the most current version of the application will be deployed, as well as the correct version of JRE.
- Applications launched with Java Web Start are cached locally. Thus, an already downloaded application is launched on par with a traditionally installed application.

- You can preconfigure Oxygen XML Editor and the rest of your team will use the same preferences and *frameworks*.

Important: If you want to create your own JWS distribution package, please *contact Syncro Soft* for permission through a *Value Added Reseller Agreement*.

Note: A code signing certificate is needed to sign the JWS distribution. The following procedure assumes that you already have such a certificate (for example, Thawte™, or Verisign™).

The following schematics depicts the Oxygen XML Editor Java Web Start deployment procedure:

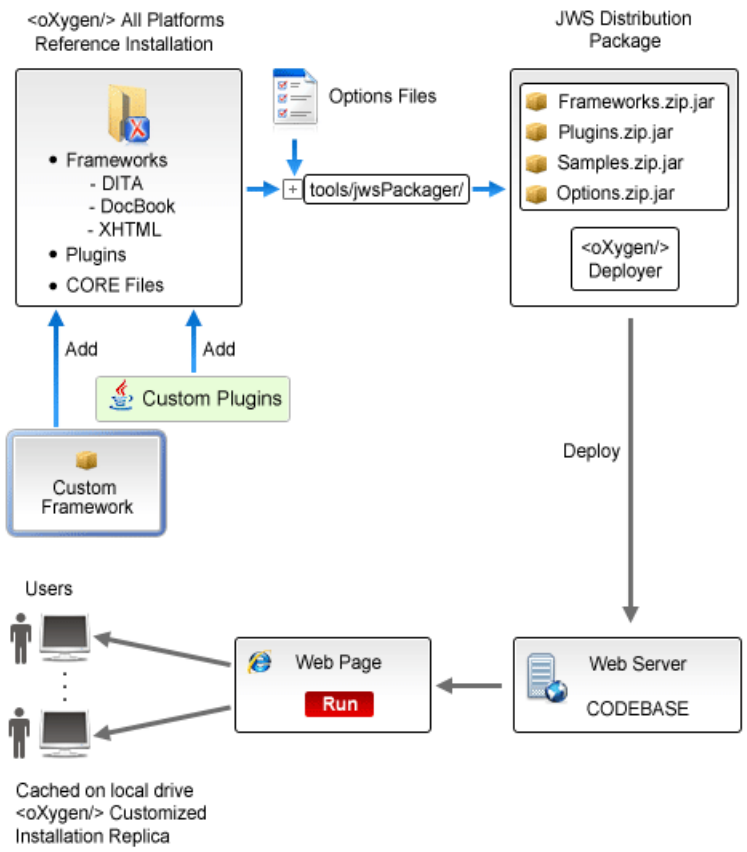


Figure 10: Java Web Start Deployment Procedure

To deploy an Oxygen XML Editor installation on a server.

1. Go to *https://www.oxygenxml.com/InstData/Editor/All/oxygen.tar.gz* and download the All Platforms Installation package to a local drive.
2. Expand the archive to a temporary location.
   The `oxygen` folder is created.
3. Optionally, you can customize your own *framework*.
4. Edit the `oxygen\tools\jwsPackager\packager.properties` configuration file. Adjust the following properties appropriately for your server:
   - **codebase** - Represents the location of the future JWS distribution.
   - **keystore** - The *keystore* location path.
   - **storepass** - The password for *keystore* integrity.
   - **storetype** - The type of the certificate file, such as PKCS12 or JKS.
   - **alias** - The *keystore* alias.
   - **optionsDir** - Points to the options directory that may be distributed with the JWS installer. If the directory contains an XML document named `options.xml` or `default.xml` containing exported options, these options will be used. Otherwise, the structure of the options folder has to match the structure of a stand alone application options folder.

Note: This property is optional. It is provided only if *custom options* need to be delivered to the end users.

The values of **keystore**, **storepass**, and **alias** properties are all provided by the code signing certificate. For more information, see the documentation for your *jarsigner* tool.

5. If you want to modify the default settings, edit the JNLP `oxygen\tools\jwsPackager\dist\javawebstart\oxygen\oxygen.jnlp` template file. You can specify the list of files opened at startup by modifying the `<argument>` list. To pass system properties directly to Oxygen XML Editor when it is started, add the `oxy` prefix to them (for example: `<property name="oxyPropertyName" value="testValue"/>`). The system property is passed to Oxygen XML Editor with the prefix stripped.
6. Open a command-line console and run `ant` in the `oxygen\tools\jwsPackager` folder.
   The **ant** process creates the `oxygen\tools\jwsPackager\dist\InstData\oxygenJWS.zip` archive that contains the actual remote JWS installer.
7. Copy the expanded content of the archive to the folder specified in the **codebase** property, previously set in the **`packager.properties`** file.
8. Using your favorite web browser, go to the address specified in the **codebase** property or to its parent folder and start the remote installer.

   Important: When running the Java Web Start distribution on OS X, due to changes in this *security release*, clicking the link to the JNLP file does not start the application. The selected JNLP is downloaded locally. Right-click it and choose to open the resource.

Group Deployment

If you are deploying Oxygen XML Editor for a group, there are a number of things you can do to customize Oxygen XML Editor for your users and to make the deployment more efficient.

Creating custom default options

You can *create a custom set of default options* for Oxygen XML Editor. These will become the default options for each of your users, replacing the normal default settings. Users can still set options to suit themselves in their own copies of Oxygen XML Editor, but if they choose to reset their options to defaults, the custom defaults that you set will be used.

Creating default project files

Oxygen XML Editor project files are used to configure a project. You can create and deploy default project files for your projects so that your users will have a preconfigured project file to begin work with.

Shared project files

Rather than each user having their own project file, you can create and deploy shared project files so that all users share the same project configuration and settings and automatically inherit all project changes.

Using the unattended installer

You can speed up the installation process by using the *unattended installer for Windows* or *Linux installs*.

Using floating licenses

If you have a number of people using Oxygen XML Editor on a part-time basis or in different time zones, you can use a *floating license* so that multiple people can share a license.

Obtaining and Registering a License Key for Oxygen XML Editor

Oxygen XML Editor is not free software. To activate and use Oxygen XML Editor, you need a license.

For demonstration and evaluation purposes, a time limited license is available upon request at *https://www.oxygenxml.com/register.html*. This license is supplied at no cost for a period of 30 days from the date of issue. During this period, the software is fully functional, enabling you to test all its functionality. To continue using the software after the trial period, you must purchase a permanent license.

Choosing a License Type

You can use one of the following license types with Oxygen XML Editor:

1. A **Named-User License** may be used by a single *Named User* on one or more computers. Named-user licenses are not transferable to a new *Named User*. If you order multiple named-user licenses, you will receive a single license key good for a specified number of *named users*. It is your responsibility to keep track of the *named users* that each license is assigned to.
2. A **Floating License** may be used by any user on any machine. However, the total number of copies of Oxygen XML Editor in use at one time must not be more than the number of floating licenses available. A user who runs two different distributions of Oxygen XML Editor (for example, Standalone and Eclipse Plugin) at the same time on the same computer, consumes a single floating license.
3. A **Subscription** license that allows you to use the application for a specific period of time (either 6 months or 1 year). This type of license is user-based and is covered by a Support and Maintenance Pack, which means that during the subscription period you will get free upgrades to all major and minor releases and priority technical support.
4. A special **Academic Group License** (*Classroom*, *Department*, or *Site* license) may be used by students and teachers in academic institutions. These licenses provide a cost effective way of getting access to Oxygen XML Editor for learning purposes.

For definitions and legal details of the license types, consult the End User License Agreement available at *https://www.oxygenxml.com/eula.html*.

Obtaining a License

You can obtain a license for Oxygen XML Editor in one of the following ways:

- You can purchase one or more licenses from the Oxygen XML Editor website at *https://www.oxygenxml.com/buy.html*. A license key will be sent to you by email.
- If your company or organization has already purchased licenses, please contact your license administrator to obtain a license key.
- If you purchased a subscription and you received a registration code, you can use it to obtain a license key from *https://www.oxygenxml.com/registerCode.html*. A license key will be sent to you by email.
- If you want to evaluate the product, you can obtain a trial license key for 30 days from the Oxygen XML Editor website at *https://www.oxygenxml.com/register.html*.

Register a Named-User or Subscription License

To register a *Named-User License* or *Subscription License* on a machine owned by the *Named User*, follow these steps:

1. Purchase a license from the *Oxygen XML Editor website*. You will receive an email that contains your license key.
2. Save a backup copy of your email message that contains the new license key.
3. Start Oxygen XML Editor.
   If this is a new install of Oxygen XML Editor, the registration dialog box is displayed. If the registration dialog box is not displayed, go to **Help** > **Register**.

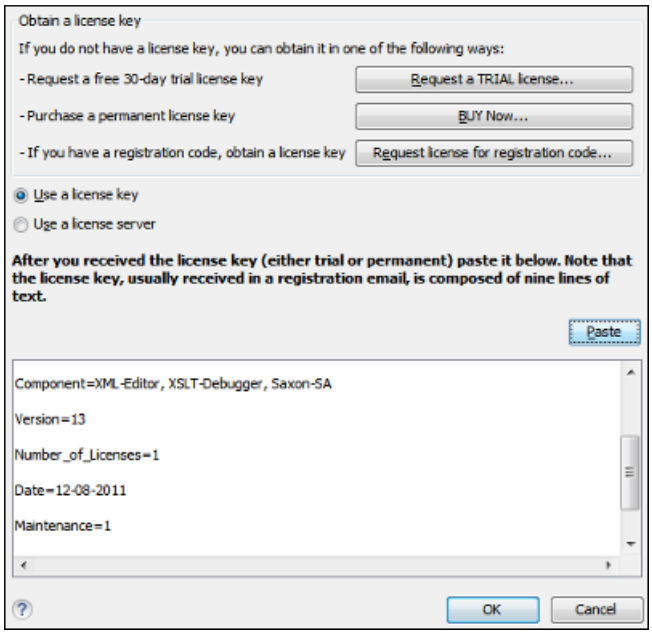


Figure 11: License Registration Dialog Box

4. Select **Use a license key** as licensing method.
5. Paste your license key into the registration dialog box. The license key is composed of nine lines of text between two text markers.
6. Press **OK**.

Related Information:

Oxygen XML Editor End-User License Agreement

Registering a Floating License

How you register to use a floating license will depend on how floating licenses are managed in your organization.

- If the machines that share the pool of floating licenses are on multiple network segments, someone in your company will need to *set up a license server*. Consult that person to determine if they have set up a license server as a *TCP* or *HTTP* server as the registration process is different for each.
- If all the machines sharing a pool of floating licenses are on the same network segment, you will register your licence the same way you *register a Named-User Licence*. Oxygen XML Editor will use your connection to a local area network, without additional notice, to automatically connect to other running instances of Oxygen XML Editor. These connections may transmit your IP address to the local network.

  Note: [For System Administrators] Multiple running instances of Oxygen XML Editor communicate with each other using UDP broadcast on the 59153 port, to the 239.255.255.255 group.

  Warning: This mechanism was deprecated starting with version 17.0 and it is scheduled for removal in a future version. It is recommended to switch to the license server licensing mechanism.

Request a Floating License from a TCP License Server

Use this procedure if your company uses an Oxygen XML Editor TCP license server and the license server has already been set up by your server administrator:

1. Contact your server administrator to get network address and login details for the license server.
2. Start Oxygen XML Editor.
3. Go to **Help** > **Register** .
   The license registration dialog box is displayed.
4. Choose **Use a license server** as licensing method.
5. Select **TCP server** as server type.
6. In the *Host* field, enter the host name or IP address of the license server.

7. In the *Port* field, enter the port number used to communicate with the license server.
8. Click the **OK** button.

If a floating license is available, it is registered in Oxygen XML Editor. To display the license details, open the **About** dialog box from the **Help** menu. If a floating license is not available, you will get a message listing the users currently using floating licenses.

Related Information:

Setting up a TCP Floating License Server Using a 32-bit Windows Installer on page 41
Setting up the TCP floating license server as a Windows process.

Setting up a TCP Floating License Server Using All-Platforms Distribution on page 43
This installation method can be used for running the TCP license server on any platform where a Java virtual machine can run (OS X, Linux/Unix, Windows).

Request a Floating License from an HTTP License Server

Use this procedure if your company uses an Oxygen XML Editor HTTP license server and the license server has already been set up by your server administrator:

1. Contact your server administrator to get network address and login details for the license server.
2. Start Oxygen XML Editor.
3. Go to **Help** > **Register**.
   The license registration dialog box is displayed.
4. Choose **Use a license server** as licensing method.
5. Select **HTTP/HTTPS Server** as server type.
6. In the *URL* field, enter the address of the license server.
   The URL address has the following format: `http://hostName:port/oXygenLicenseServlet/license-servlet`.
7. Complete the *User* and *Password* fields.
8. Click the **OK** button.

If a floating license is available, it is registered in Oxygen XML Editor. To display the license details, open the **About** dialog box from the **Help** menu. If a floating license is not available, you will get a message listing the users currently using floating licenses.

Related Information:

Setting up an HTTP Floating License Server on page 37

Release a Floating License

The floating license you are using will be released and returned to the pool if any of the following occur:

- The connection with the license server is lost.
- You exit the application running on your machine, and no other copies of Oxygen XML Editor running on your machine are using your floating license.
- You register a *Named User* license with your copy of Oxygen XML Editor, and no other copies of Oxygen XML Editor running on your machine are using your floating license.
- Your computer idles for more than 2 hours.
- Your system administrator manually revokes the license.

Tip: To prevent your floating license from being released, you can use the **Lock floating license** action available in the **Help** menu. You can use the same action to unlock the license and your system administrator also has the ability unlock your license.

To release a floating license on demand, follow these steps:

1. Go to **Help** > **Register**.
   The license registration dialog box is displayed.
2. The license key field should be empty (this is normal). If it is not empty, delete any text in the field.
3. Make sure the **Use a license key** option is selected.

4. Click **OK**.
   A dialog box is displayed asking if you want to reset your license key.
5. Select between:
   - **Use the last one** - Falls back to your previous license key. Use this option if you want to release a floating license and revert to a *Named User* license.
   - **Reset** - Removes your license key from your user account on the current computer.

   The **Reset** button erases all the licensing information. To complete the reset operation, close and restart Oxygen XML Editor.

Register a Floating License for Multiple Users

If you are an administrator registering floating licenses for multiple users, you can avoid having to open Oxygen XML Editor on each machine and configuring the registration details by using the following procedure:

1. Reset the registration details:
   a. Select **Register** from the **Help** menu.
   b. Click **OK** without entering any information in this dialog box.
   c. Click **Reset** and restart the application.
2. Register the license using one of the *floating license registration procedures*.
3. Copy the `license.xml` file from the Oxygen XML Editor *preferences directory* to the installation folder on each installation to be registered.

Setting Up a Floating License Server

Installing a License Server to Manage Floating Licenses

If you are using floating licenses for Oxygen XML Editor, you must set up an Oxygen XML Editor floating license server. A floating license server can be installed as one of the following:

- An *HTTP server*. This is the recommended method.
- A *TCP server* (deprecated).

Note: Oxygen XML Editor version 17 or higher requires a license server version 17 or higher. License servers version 17 or higher can be used with any version of a floating license key.

Activating Floating License Keys

To help you comply with the Oxygen XML Editor EULA (terms of licensing), all floating licenses require activation. This means that the license key will be locked to a particular license server deployment and no multiple uses of the same license key are possible.

During the activation process, a code that uniquely identifies your license server deployment is sent to the Oxygen XML Editor servers, which in turn will sign the license key.

Split or Combine License Keys to Work with Your License Servers

A license server can only manage one license key (which can cover any number of floating licenses). If you have multiple license keys for the same Oxygen XML Editor version and you want to have all of them managed by the same server, or if you have a multiple-user floating license and you want to split it between two or more license servers, please contact *support@oxygenxml.com* and ask for a new license key.

Setting up an HTTP Floating License Server

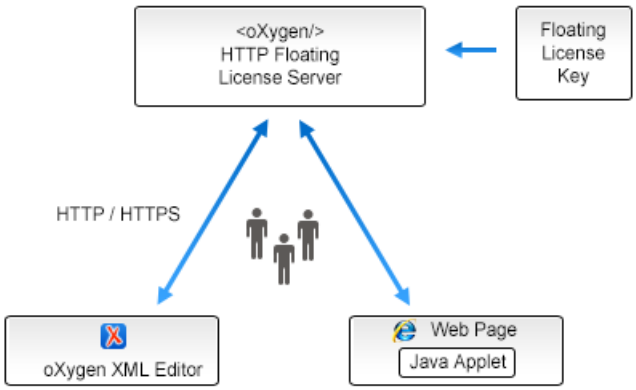


Figure 12: Floating License Server (HTTP Server)

The Oxygen XML Editor license server is available in several distributions, tailored for covering a variety of deployment configurations:

- ***Windows installer*** - Easy-to-use Windows installation wizard. Requires elevated permissions to run it.
- ***All-platform distribution*** - Script-based deployment that does not require elevated permissions to run it. Provides scripts for Windows, Mac, and Linux.
- ***Web Archive (WAR) distribution*** - Provides more flexibility in your deployment configuration, but it requires an existing HTTP server (such as Apache Tomcat).

Installation Steps for the HTTP License Server Installer Distribution for Windows

1. Download the HTTP license server installer from the *Oxygen XML Editor website*.
2. Run the installer and follow the on-screen instructions.
3. You need to configure two sets of credentials:
   a. **Administrator credentials** - used for accessing the Oxygen XML Editor license server administrative interface. Optionally you can choose to change the standard 8080 port.
   b. **Standard user credentials** - used by an Oxygen XML Editor application to connect to the license server.
4. Optionally you can choose to install the server as a Windows service. In this case, you can choose the name of the Windows service.

Installation Steps for the HTTP License Server All-Platform Distribution

1. Download the HTTP license server all-platform archive from the *Oxygen XML Editor website*.
2. Unpack the archive.
3. Run the license server scripts suitable for your operating system (`licenseServer.bat` for Windows or `licenseServer.sh` for Linux and Mac).

   Note: To specify a different port (other than the default 8080), you can pass the new port number as an argument to the scripts (for example, `licenseServer.bat 8082`).
4. On the first run, you will be prompted to set two sets of credentials:
   a. **Administrator credentials** - used for accessing the Oxygen XML Editor license server administrative interface.
   b. **Standard user credentials** - used by an Oxygen XML Editor application to connect to the license server.

Installation Steps for the HTTP License Server WAR Distribution

1. Make sure that Apache Tomcat 5.5 or higher is running on the machine you have selected to be the license server. To get it, go to *http://tomcat.apache.org*.

2. Download the HTTP license server **Web ARchive** (**.war**) from the *Oxygen XML Editor website*.
3. Configure two Tomcat users:
   a. One user with the role *user*, used by an Oxygen XML Editor application to connect to the license server. In the subsequent example, this user name is **John**.
   b. Another user with the roles *admin* and *manager-gui*, used for accessing the Oxygen XML Editor license server administrative interface and the Tomcat management interface. In the subsequent example, this user name is **Mary**.

   A typical way to achieve this is to edit the `tomcat-users.xml` file from your Tomcat installation (if using a Tomcat **zip**/**tar.gz** distribution, by default this configuration file is found in the `/TomcatInstallFolder/conf/` directory). After adding the two users, the configuration file might look like this:

```
<tomcat-users xmlns="http://tomcat.apache.org/xml"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
              version="1.0">
  <!-- ... other user and role definitions ... -->
  <role rolename="user"/>
  <role rolename="admin"/>
  <role rolename="manager-gui"/>
  <user username="John" password="user_pass" roles="user"/>
  <user username="Mary" password="admin_pass" roles="admin,manager-gui"/>
</tomcat-users>
```

4. Go to the Tomcat Web Application Manager page and log-in with the user you configured with the *manager-gui* role (**Mary** in the example above). In the **WAR file to deploy** section, choose the WAR file and click the **Deploy** button. The `oXygenLicenseServlet` application is now up and running, but the license key is not yet registered.
5. Go to the Oxygen license server administration page by clicking the `oXygenLicenseServlet` link in the manager page. You will need to authenticate with the user configured with the `admin` role (Mary in our example).
6. Activate the floating license key. This process involves binding your license key to your license server deployment. Once the process is completed you cannot activate the floating license with another license server. Follow these steps to activate the license:
   a. Access the HTTP license server by following the link provided by the Tomcat Web Application Manager page. If prompted for authentication, use the credentials configured for the *admin* or *manager* users.

      Result: A page is displayed that prompts for a license key.
   b. Paste your floating license key into the form and press **Submit**. The browser used in the activation process needs to have Internet access.

      Result: You will be redirected to an online form hosted on the Oxygen XML Editor website. This form is pre-filled with an activation code that uniquely identifies your license server deployment, and your license key.

      Note: If, for some reason, your browser does not take you to this activation form, refer to the *Manual Activation Procedure*.
   c. Press **Activate**.

      If the activation process is successfully completed, your license server is running. Follow the on-screen instructions to configure the Oxygen XML Editor client applications.
7. By default, the license server logs its activity in the `TomcatInstallDir/logs/oxygenLicenseServlet.log` file. To change the log file location, edit the `log4j.appender.R2.File` property from the `TomcatInstallDir/webapps/oXygenLicenseServlet/WEB-INF/lib/log4j.properties` configuration file.

Manual License Activation Procedure

1. Access the HTTP license server by following the link provided by the Tomcat Web Application Manager page. You will be taken to the license registration page.
2. Copy the license server activation code.
3. Go to the activation page at `http://www.oxygenxml.com/activation/`.
4. Paste the license server activation code and floating license key in the displayed form, then click **Activate**.
5. The activated license key is displayed on-screen. Copy the activated license key and paste it in the license registration page of the HTTP server.

Automatic Subscription Renewal

If the HTTP license server is configured with a subscription license, then the license server will automatically check to see if a new subscription license was purchased and will automatically download and install it for you.

Important: This automatic checking procedure implies a connection to a web service located at `oxygenxml.com`. You can deactivate this automatic behavior by deselecting the **Automatically check for subscription renewal** option from the main management page of the license server.

If the automatic renewal process fails, you can try either of the following possible solutions:

- If your server uses a proxy to connect to the Internet, go to the main management page of the license server and configure the proxy settings by clicking the **Proxy settings** link in the **Management tasks** section.
- *Manually replace the floating license key*.

Floating License Server Management and Statistics Pages

A system administrator can manage and access information about the floating license server at: `http://hostName:port/oXygenLicenseServlet`.

This page provides access to several statistics reports and management tasks. It also shows the current status of the server and provides additional instructions for using the license server with Oxygen XML Editor.

This page includes the following links for accessing statistics or managing tasks:

- **Current Allocated Licenses** - Opens the *Allocated License Report page*.
- **Usage Statistics** - Opens the *Floating License Usage Statistics page*.
- **View License Key** - Use this link to view details about the license key.
- **Replace License Key** - Use this link if you need to *replace a license key*.
- **Configuration** - Opens a page where you can configure notification settings and specify whether or not users are allowed to lock licenses. This page can be used for setting up the mail server used for sending rejection emails.

Allocated License Report Page

This report page provides a system administrator the ability to revoke or unlock current running instances of licenses and includes the following information:

- **License load** - A graphical indicator that shows how many licenses are available.
- **Floating license server status** - General information about the license server status, such as start time, license counts, rejected and acknowledged requests, average usage time, license refresh and timeout intervals, location of the license key, and the server version.
- **Current running instances** - Lists all currently acknowledged users, including user name, date and time when the license was granted, IP and MAC address of the computer where Oxygen XML Editor runs, and lock status.
  - **Revoke** - A system administrator can click on the **Revoke** icon next to a user name to release that particular license and return it to the pool.
  - **Unlock** - If a user has locked their floating license, the system administrator can also unlock it from this page.

Note: This report is also available in XML format at: `http://hostName:port/oXygenLicenseServlet/license-servlet/report-xml`.

Floating License Usage Statistics Page

This report page provides some usage statistics for the floating licenses. It is helpful for determining the number of licenses that are needed and monitoring times when licenses are consumed. It includes the following information:

- **Maximum number of concurrent licenses** - Shows the maximum number of floating licenses that can be consumed at any given time.
- **Concurrent license consumption per day** - A chart that shows the peak number of licenses that were consumed and the total number of users that were rejected, on a daily basis. This chart can be used to detect the amount of concurrent licenses that are needed to avoid having rejected users.

Tip: You can click on any bar to see the license consumption per hour for that particular day.

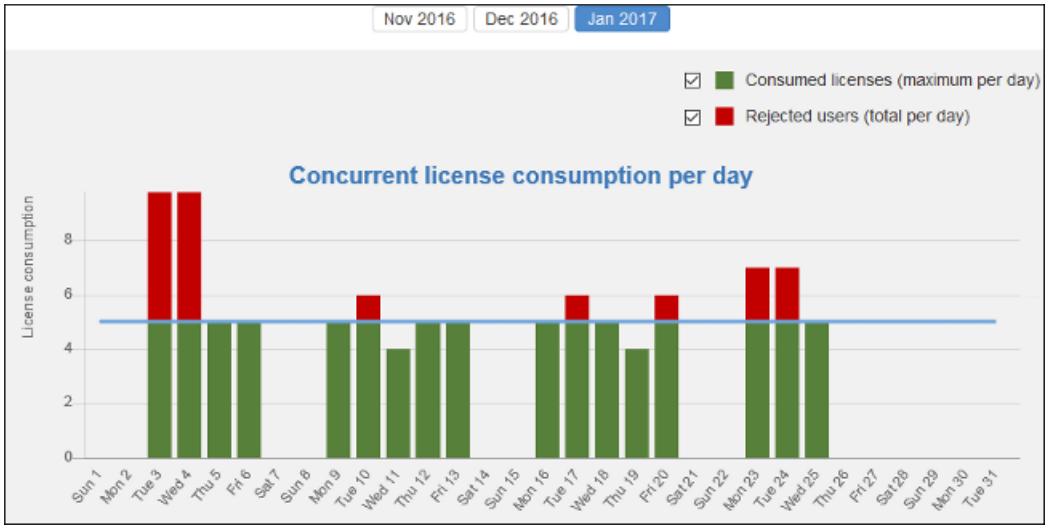


Figure 13: Concurrent License Consumption per Day Chart

- **Concurrent license consumption per hour** - A chart that shows the peak number of licenses that were consumed per hour throughout that particular month. This is useful for identifying the time of day when the most licenses were consumed.

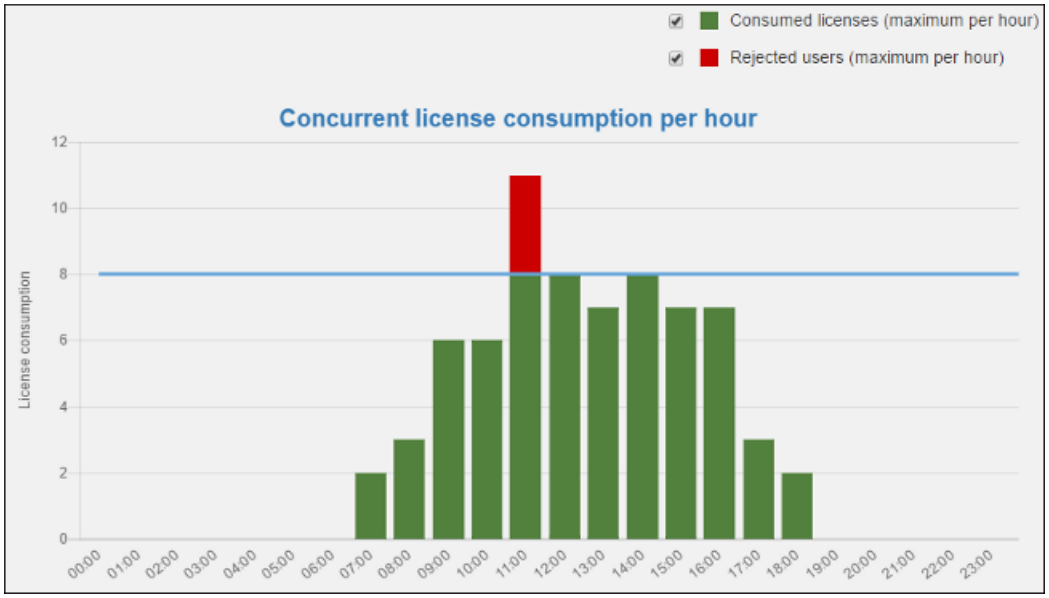


Figure 14: Concurrent License Consumption per Hour Chart

Replacing a Floating License Key in an HTTP Floating License Server

The following procedure assumes that your Oxygen XML Editor HTTP floating license server contains a previously *activated license key* and provides instructions for replacing it with another one. The goal of the procedure is to allow you to activate and configure the new license key without any downtime.

This is useful if, for instance, you want to upgrade your existing license to the latest version or if you receive a new license key *that accommodates a different number of users*.

To replace a floating license key that is activated on your HTTP floating license server with a new one, follow these steps:

1. Access the license server by following the link provided by the Tomcat Web Application Manager page.
2. Click the **Replace license key** link. This will open a page that contains details about the license currently in use.

3. Click the **Yes** button to begin the replacement procedure.
4. Paste the new floating license key in the displayed form, then click **Submit**. The browser used in the process needs to have Internet access.

   You will be redirected to an online form hosted on the Oxygen XML Editor website. This form is pre-filled with an activation code that uniquely identifies your license server deployment and your license key.

   Note: If for some reason your browser does not take you to this activation form, refer to the *Manual Activation Procedure*.
5. Press **Activate**.

   If the activation process is completed successfully, your license server is now running using the new license key. You can click **View license key** to inspect the key currently used by the license server.

Upgrading Your HTTP Floating License Server

The goal of the following procedure is to help you minimize the downtime when you upgrade the Oxygen XML Editor HTTP floating license server to its latest version.

Follow this procedure:

1. Access the license server by following the link provided by the Tomcat Web Application Manager page. If prompted for authentication, use the **admin** or **manager** credentials.
2. Click the **View license key** link and copy the displayed license key to a file for later use.
3. Go to the Tomcat Web Application Manager page, log in with the user you configured with the **manager** role, and *Undeploy* the floating license server.
4. Go to Oxygen XML Editor website and *download the HTTP license server*.
5. Deploy the downloaded license server.
6. Access the license server by following the link provided by the Tomcat Web Application Manager page. If prompted for authentication, use the credentials configured for the **admin** or **manager** users.
7. Paste the license key into the form and register it.

Setting up a TCP Floating License Server Using a 32-bit Windows Installer

Setting up the TCP floating license server as a Windows process.

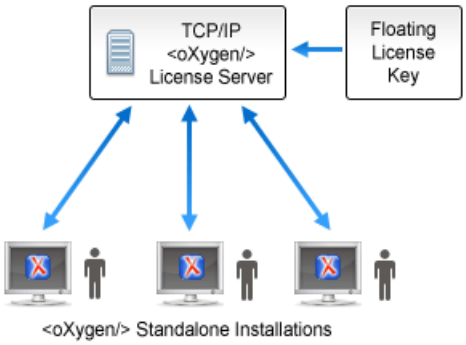


Figure 15: TCP Floating License Server (Process in Windows)

Installation Steps

1. Download the license server installation kit for Windows from the *Oxygen XML Editor website*.
2. Run the downloaded installer and follow the on-screen instructions.

   By default, the installer installs the license server as a Windows service. Optionally, you have the ability to start the Windows service automatically at Windows startup or create shortcuts on the **Start** menu for starting and stopping the Windows service manually. If you want to manually install, start, stop, or uninstall the server as a Windows service, run the following scripts from a command line as an Administrator:

- `installWindowsService.bat [serviceName]` - Installs the server as a Windows service with the name *serviceName*. The parameters for the license key folder and the server port can be set in the `oXygenLicenseServer.vmoptions` file.
- `startWindowsService.bat [serviceName]` - Starts the Windows service.
- `stopWindowsService.bat [serviceName]` - Stops the Windows service.
- `uninstallWindowsService.bat [serviceName]` - Uninstalls the Windows service.

Note: If you do not provide the `serviceName` argument, the default name *oXygenLicenseServer* is used.

If the license server is installed as a Windows service, the output and error messages are automatically redirected to the following log files that are created in the install folder:

- `outLicenseServer.log` - Standard output stream of the server.
- `errLicenseServer.log` - Standard error stream of the server.

3. Manually add the oXygenLicenseServer.exe file in the Windows Firewall list of exceptions. Go to **Control Panel** > **System and Security** > **Windows Firewall** > **Allow a program or feature through Windows Firewall** > **Allow another program** and browse for oXygenLicenseServer.exe from the Oxygen XML Editor License Server installation folder.
4. Floating licenses require activation prior to use. More details are available either on-screen (if the license server is started in a command line interface) or in the `outLicenseServer.log` log file.

   Note: A license server can only manage one license key (which can cover any number of floating licenses). If you have multiple license keys for the same Oxygen XML Editor version and you want to have all of them managed by the same server, or if you have a multiple-user floating license and you want to split it between two or more license servers, please contact *support@oxygenxml.com* and ask for a new license key.

Replacing a Floating License Key in a TCP Floating License Server

The following procedure assumes that your Oxygen XML Editor TCP floating license server contains a previously *activated license key* and provides instructions for replacing the activated license key with another one. The goal of the procedure is to minimize the license server downtime during the activation step of the new license key.

This is useful if, for instance, you want to upgrade your existing license to the latest version or if you receive a new license key *that accommodates a different number of users*.

To replace a floating license key that is activated on your floating license server with a new one, follow these steps:

1. Stop the service that runs the floating license server.
2. Locate the folder that holds the previous activated license key (by default, it is named `license` and it is located in the installation directory of the license server).
3. Remove the `license.txt` file and try to restart the server. Since the file that stores the license key is missing, *the server will fail to start*.
4. Find the license activation procedure in the on-screen instructions (if the license server is started in a **command line interface**) or in the `outLicenseServer.log` log file.
5. After you copy the activated license key in the `license.txt` file, restart the license server.

Upgrading Your TCP Floating License Server

The goal of the following procedure is to help you minimize the downtime generated when you upgrade the Oxygen XML Editor floating license server to its newest version.

Follow this procedure:

1. Go to the Oxygen XML Editor website and download the *latest floating license server*.
2. Run the installation kit.
3. Leave the default **Update the existing installation** option selected. This will ensure that some options set in the previous version (namely the installation folder, port number, and the floating license key in use) of the license server will be preserved.
4. Follow the on-screen instructions to complete the installation process.

Common Problems

This section includes some common problems that may appear when setting up a TCP floating license server.

Windows Service Reports 'Incorrect Function When Started'

The "Incorrect Function" error message when starting the Windows service usually appears because the Windows service launcher cannot locate a Java virtual machine on your system.

Make sure that you have installed a 32-bit Java SE from Oracle (or Sun) on the system: *http://www.oracle.com/technetwork/java/javase/downloads/index.html*.

Windows Service Reports 'Error 1067: Process Terminated Unexpectedly'

This error message appears if the Windows service launcher quits immediately after being started.

This problem usually happens because the license key has not been correctly deployed (`license.txt` file in the license folder). For more information, see the *Setting up a Floating License Server section*.

Setting up a TCP Floating License Server Using All-Platforms Distribution

This installation method can be used for running the TCP license server on any platform where a Java virtual machine can run (OS X, Linux/Unix, Windows).

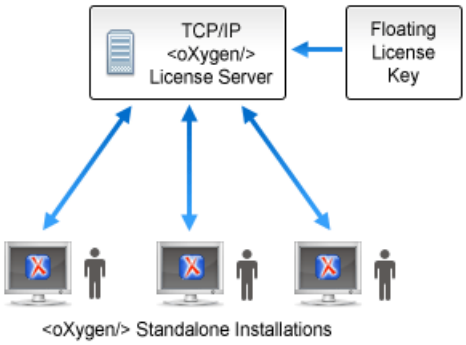


Figure 16: TCP Floating License Server (All-Platforms Distribution)

Installation Steps

1. Ensure that a Java runtime version 6 or later is installed on the server machine.
2. Download the license server installation kit for your platform from the *Oxygen XML Editor website*.
3. Unzip the installation kit into a new folder.
4. Start the server using the startup script from a command line console.

   The startup script is called `licenseServer.sh` for OS X and Unix/Linux or `licenseServer.bat` for Windows. The following parameters are accepted:

   - `licenseDir` - The path of the directory where the license files will be placed. The default value is `license`.
   - `port` - The TCP port number used to communicate with Oxygen XML Editor instances. The default value is **12346**.

   Example: The following is an example of the command line for starting the license server on Unix/Linux and OS X:

   ```
   sh licenseServer.sh myLicenseDir 54321
   ```

5. Floating licenses require activation prior to use. Follow the on-screen instruction to complete the license activation process.

Replacing a Floating License Key in a TCP Floating License Server

The following procedure assumes that your Oxygen XML Editor TCP floating license server contains a previously *activated license key* and provides instructions for replacing the activated license key with another one. The goal

of the procedure is to minimize the HTTP license server downtime during the activation step of the new license key.

This is useful if, for instance, you want to upgrade your existing license to the latest version or if you receive a new license key *that accommodates a different number of users*.

To replace a floating license key that is activated on your floating license server with a new one, follow these steps:

1. Stop the process that runs the floating license server.
2. Locate the folder that holds the previous activated license key (by default, it is named `license` and it is located in the installation directory of the license server).
3. Remove the `license.txt` file and try to restart the server. Since the file that stores the license key is missing, the server will fail to start.
4. Find the license activation procedure in the on-screen instructions.
5. After you copy the activated license key in the `license.txt` file, restart the license server.

Upgrading Your TCP Floating License Server

The goal of the following procedure is to help you minimize the downtime generated when you upgrade the Oxygen XML Editor TCP floating license server to its newest version.

Follow this procedure:

1. Stop the current license server process.
2. Locate and open the floating server startup script. It should look like this:

```
sh licenseServer.sh pathToLicenseDir 54321
```

3. Make a note of the path to the license directory (in our example is `pathToLicenseDir`) and the port number (in our example is 54321).
4. Go to the license directory and copy the license key file (`license.txt`) for later use.
5. Go to the Oxygen XML Editor website and download the *all-platforms floating license server installation kit*.
6. Unzip the archive and overwrite the content of your current floating license server installation.
7. Copy the license key file (`license.txt`) saved in step 4 to `license` directory of the floating license server installation.
8. Edit the floating server startup script and configure with the info you made note of in step 3.
9. Start the floating license server process.

Transferring a License Key

If you want to transfer your Oxygen XML Editor license key to another computer (for example, if you are disposing of your old computer or transferring it to another person), you must first unregister your license. You can then *register your license* on the new computer in the normal way.

1. Go to **Help** > **Register**.
   The license registration dialog box is displayed.
2. The license key field should be empty (this is normal). If it is not empty, delete any text in the field.
3. Make sure the **Use a license key** option is selected.
4. Click **OK**.
   A dialog box is displayed asking if you want to reset your license key.
5. Select between:
   - **Use the last one** - Falls back to your previous license key, if applicable.
   - **Reset** - Removes your license key from your user account on the current computer.

   The **Reset** button erases all the licensing information. To complete the reset operation, close and restart Oxygen XML Editor.

Upgrading Oxygen XML Editor

From time to time, upgrade and patch versions of Oxygen XML Editor are released to provide enhancements that fix problems, and add new features.

Checking for New Versions of Oxygen XML Editor

Oxygen XML Editor checks for new versions automatically at start up. To disable this check, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Global**, and deselect **Automatic Version Checking**.

To check for new versions manually, go to **Help** > **Check for New Versions**.

What is Preserved During an Upgrade?

When you install a new version of Oxygen XML Editor, some data is preserved and some is overwritten. If there is a previous version of Oxygen XML Editor already installed on your computer, it can coexist with the new one, which means you do not have to uninstall it.

If you install over a previously installed version:

- All the files from its install directory will be removed, including any modification in *framework* files, XSLT stylesheets, *XML Catalogs*, and templates.
- All global user preferences are preserved in the new version.
- All project preferences will be preserved in their project files.
- Any custom *frameworks* that were stored outside the installation directory (as configured in ***Document type associations > Locations***) will be preserved and will be found by the new installation.

If you install in a new directory.

- All the files from the old install directory will be preserved, including any modification in *framework* files, XSLT stylesheets, *XML Catalogs*, and templates. However, these modifications will not be automatically imported into the new installation.
- All global user preferences are preserved in the new version.
- All project preferences will be preserved in their project files.
- Any custom *frameworks* that were stored outside the installation directory (as configured in ***Document type associations > Locations***) will be preserved and will be found by the new installation.

Upgrading the Standalone Application

1. Upgrading to a new version might require a new license key. To check if your license key is compatible with the new version, select **Help** > **Check for New Version**. Note that the application needs an Internet connection to check the license compatibility.
2. Download and install the new version according to the instructions for your platform and the type of installer you selected.
3. If you installed from an archive (as opposed to an executable installer) you may have to update any shortcuts you have created or modify the system PATH to point to the new installation folder.
4. Start Oxygen XML Editor.
5. If you require an new license for your upgrade, install it now according to the procedure for your platform and the type of installer you selected.

Installing and Updating Add-ons in Oxygen XML Editor

Oxygen XML Editor provides an *add-on* mechanism that can automatically discover and install *frameworks* and *plugins* from a remote location.

Note: *Frameworks* that you install through the add-ons system are read-only.

Installing Add-ons

To install a new add-on, follow these steps:

1. Go to **Help** > **Install new add-ons**.
2. In the displayed dialog box, fill-in the **Show add-ons from** with the *update site* that hosts add-ons. If you want to see all Oxygen XML Editor default add-ons, choose the **ALL AVAILABLE SITES** option from the **Show add-ons from** drop down. The add-ons list contains the name, status, update version, Oxygen XML Editor version, and the type of the add-on (either **framework**, or **plugin**). A short description of each add-on is presented under the add-ons list.

   Note: To see all the add-ons from the remote update site, deselect **Show only compatible add-ons** and **Show only the latest version of the add-ons**. Incompatible add-ons are shown only to acknowledge their presence on the remote update site. You cannot install an incompatible add-on.
3. By default, only the latest versions of the add-ons that are compatible with the current version of Oxygen XML Editor are displayed.
4. Choose the add-ons you want to install, press the **Next** button, then follow the on-screen instructions.

   Note: Accepting the license agreement of the add-on is a mandatory step in the installation process.

   Note: All add-ons are installed in the `extensions` directory inside the Oxygen XML Editor *preferences directory*.

Managing installed add-ons

To manage the installed add-ons, follow these steps:

1. Go to **Help** > **Manage add-ons**
2. The displayed dialog box presents a list of the available updates (compatible with the current version of Oxygen XML Editor) and with the already installed updates. Under the updates list, Oxygen XML Editor presents a short description of each update.
3. Select the checkbox for a specific add-on, then press **Update** to update it (or **Uninstall** to remove it). If there is a newer version of the add-on available, Oxygen XML Editor will download the package and install it. Follow the on-screen instructions to complete the installation process.

   Note: Accepting the license agreement of the add-on is a mandatory step in the installation process.

Checking for add-on updates

To check if there are available updates for the installed add-ons, go to **Help** > **Check for add-ons updates**. This action only displays updates that are compatible with the current Oxygen XML Editor version.

To watch a video demonstration about the add-ons support in Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/AddonsSupport.html*.

Related Information:

Packing and Deploying Plugins or Frameworks as Add-ons on page 1204

Uninstalling Oxygen XML Editor

Uninstalling the Oxygen XML Editor Standalone

CAUTION: The following procedure will remove Oxygen XML Editor from your system. **All data stored in the installation directory will be removed, including any customizations or any other data you have stored within that directory. Make a back up of any data you want to keep before uninstalling Oxygen XML Editor.**

1. Backup any data you want to keep from the Oxygen XML Editor installation folder.
2. Remove the application.
   - On Windows use the appropriate uninstaller shortcut provided with your OS.
   - On OS X and Unix manually delete the installation folder and all its contents.

3. If you want to remove the user preferences:
   - **On Windows Vista/7/8/10**, remove the directory: `%APPDATA%\Roaming\com.oxygenxml` (usually `%APPDATA%` has the value: `[user-home-dir]\Application Data`). Note that this directory is hidden.
   - **On Windows XP**, remove the directory: `%APPDATA%\com.oxygenxml` (usually `%APPDATA%` has the value: `[user-home-dir]\Application Data`).
   - **On Linux**, remove the directory: `.com.oxygenxml` from the user home directory.
   - **On OS X**, remove the directory: `Library/Preferences/com.oxygenxml` of the user home folder .

Unattended Uninstall

The unattended uninstall procedure is available only on Windows and Linux.

Run the uninstaller executable from command line with the -q parameter.

The uninstaller executable is called `uninstall.exe` on Windows and `uninstall` on Linux and is located in the application's install folder.

Oxygen XML Editor Installer Command Line Reference

The Oxygen XML Editor installers for Windows and Linux creates a file called `response.varfile`, which records the choices that the user made when running the installer interactively. You can use a `response.varfile` to set the options for an unintended install. Here is an example of a `response.varfile`:

```
#install4j response file for Oxygen XML Editor 18.0
#Fri Jul 18 21:52:15 EDT 2014
sys.adminRights$Boolean=true
sys.programGroupDisabled$Boolean=false
createDesktopLinkAction$Boolean=true
autoVersionChecking=true
sys.fileAssociation.launchers$StringArray="19","19","19","19","19","19",
   "19","19","19","19","19","19","19","19","19","19","19","19","19","19",
   "19","19","19","19","19","19"
sys.programGroupAllUsers$Boolean=true
reportProblem=true
downloadResources=true
sys.languageId=en
sys.installationDir=C\:\\Program Files (x86)\\Oxygen XML Editor 18
createQuicklaunchIconAction$Boolean=true
sys.programGroupName=Oxygen XML Editor 18.0
executeLauncherAction$Boolean=true
sys.fileAssociation.extensions$StringArray="xml","dita","ditamap","ditaval",
   "xsl","xslt","xspec","xsd","rng","rnc","sch","dtd","mod","ent","nvdl","fo",
   "wsdl","xquery","xq","xqy","xqm","xql","xpl","css","json","xpr"
```

The following table describes some of the settings that can be used in the `response.varfile`:

Table 2: *response.varfile* Options Parameters

Parameter Name	Description	Values
autoVersionChecking	Automatic version checking.	*true* / *false*. Default setting is *true*.
reportProblem	Allows you to report a problem encountered while using Oxygen XML Editor.	*true* / *false*. Default setting is *true*.
downloadResources	Allows Oxygen XML Editor to download resources (links to video demonstrations, webinars and upcoming events) from *https://www.oxygenxml.com* to populate the application welcome screen.	*true* / *false*. Default setting is *true*.

The Oxygen XML Editor installation uses the install4j installer. A description of the `response.varfile` format can be found on the *install4j site*.

Command line parameters

The Oxygen XML Editor installer supports the following command line parameters:

Option	Meaning

-q	Run the installer in unattended mode. The installer will not prompt the user for input during the install. Default settings will be used for all options unless a `response.varfile` is specified using the *-varfile* option or individual settings are specified using
	- on Windows:
	`oxygen.exe -q`
	- on Linux:
	`oxygen.sh -q`

-overwrite	In unattended mode, the installer does not overwrite files with the same name if a previous version of the Oxygen XML Editor is installed in the same folder. The -overwrite parameter added after the -q parameter forces the overwriting of these files.
	- on Windows:
	`oxygen.exe -q -overwrite`
	- on Linux:
	`oxygen.sh -q -overwrite`

-console	To display a console for the unattended installation, add a -console parameter to the command line.
	- on Windows:
	`start /wait oxygen.exe -q -console`
	Note: The use of *start /wait* on Windows is required to make the installer run in the foreground. It you run it without *start /wait*, it will run in the background.
	- on Linux:
	`oxygen.sh -q -console`

-varfile	Points to the location of a `response.varfile` to be used during an unattended installation. For example:
	- on Windows:
	`oxygen.exe -q -varfile response.varfile`
	- on Linux:
	`oxygen.sh -q -varfile response.varfile`

-V	Is used to define a variable to be used by an unattended installation. For example: **- on Windows:** `oxygen.exe -q -VusageDataCollector=false`

Option	Meaning
	- on Linux: `oxygen.sh -q -VusageDataCollector=false`

The Oxygen XML Editor installation uses the install4j installer. A full list of the command line parameters supported by the install4j installer can be found on the *install4j site*.

Configuring Oxygen XML Editor

4

Topics:

- *Preferences*
- *Configuring Options*
- *Associating a File Extension with Oxygen XML Editor*
- *Configuring the Layout of the Views and Editors*
- *Configure Toolbars*
- *Import/Export Transformation or Validation Scenarios*
- *Editor Variables*
- *Custom System Properties*
- *Localizing the User Interface*
- *Setting a Java Virtual Machine Parameter when Launching Oxygen XML Editor*

This chapter presents all the user preferences and options that allow you to configure various features and aspects of the application itself. It also includes information about storing and sharing options, importing and exporting options or scenarios, customizing system properties, setting startup parameters, and the *editor variables* that are available for customizing user-defined commands..

Preferences

You can configure Oxygen XML Editor options using the **Preferences** dialog box.

To open the preferences dialog box, go to **Options** > **Preferences**.

You can select the preference page you are interested in from the tree on the left of the **Preferences** dialog box. You can filter the tree by using the filter text box and the following buttons are available to the right of the text box:

- **Expand All** - Expands the structure of the tree to show all preference pages.
- **Collapse All** - Collapses the structure of the tree to show only the 1st level preference pages.
- **Project-Level Options Only** - If toggled on, it filters the tree to only show the preference pages that are *saved at project level*.

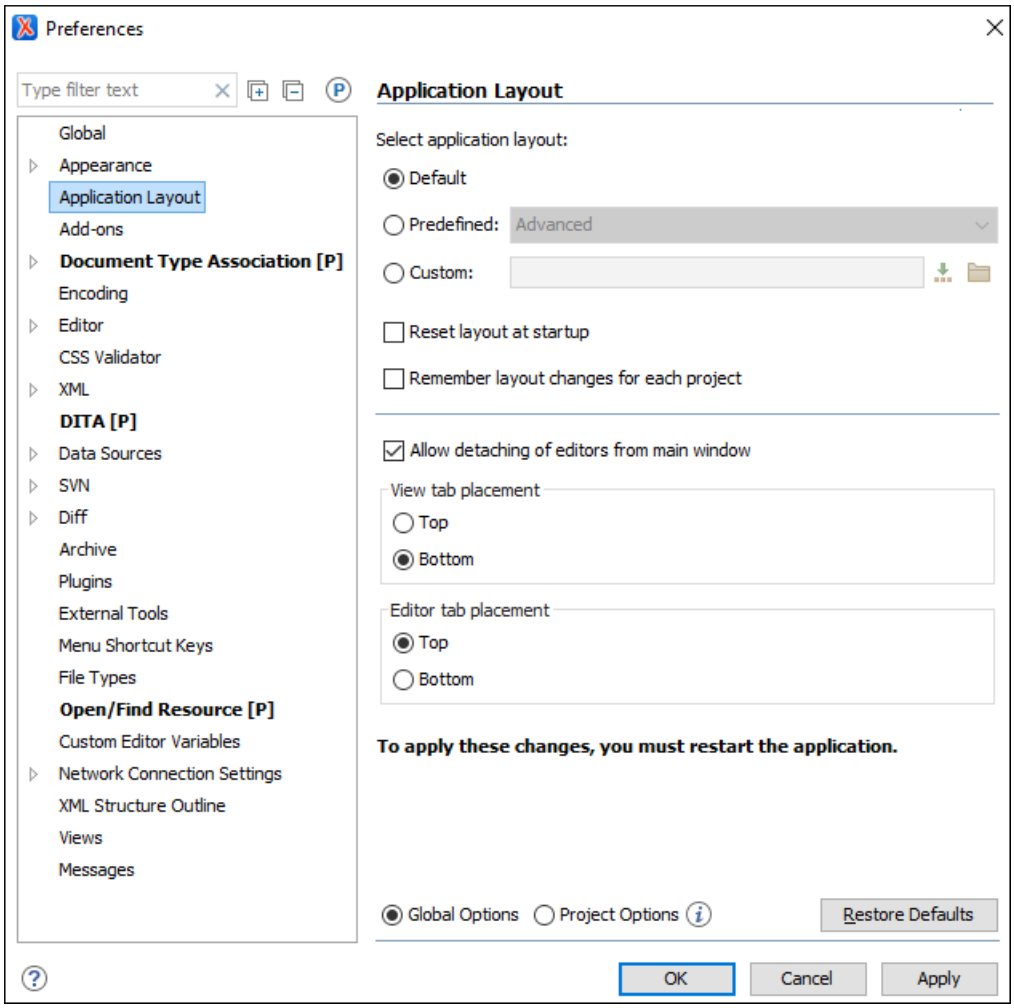


Figure 17: Preferences Dialog Box

Press ? or **F1** for help on any preferences page.

Some preference pages include a option to control how the options are stored, either as ***Global Options*** or ***Project Options***.

Figure 18: Controlling the Storage of the Preferences

You can restore options to their default values by pressing the **Restore Defaults** button, available in each preferences page.

Preferences Directory Location

A variety of resources (such as global options, license information, and history files) are stored in a preferences directory (`com.oxygenxml`) that is in the following locations:

- **Windows (Vista, 7, 8, 10)** - `[user_home_directory]\AppData\Roaming\com.oxygenxml`
- **Windows XP** - `[user_home_directory]\Application Data\com.oxygenxml`
- **Mac OS X** - `[user_home_directory]/Library/Preferences/com.oxygenxml`
- **Linux/Unix** - `[user_home_directory]/.com.oxygenxml`

Global Preferences

The global options cover a number of aspects of the overall operation of Oxygen XML Editor. To configure the **Global** options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Global**.

The following options are available in the **Global** preferences page:

Automatic Version Checking

If this option is selected, Oxygen XML Editor will check for a new version on startup.

Check for notifications

If selected (default value), the application will check for various types of messages from the Oxygen XML Editor website and they will be displayed in the status bar. The types of messages include the addition of new videos on the website, the announcement of upcoming webinars and conferences where the Oxygen XML Editor team will participate, and more.

Language

This option specifies the language used in the user interface. You can choose between English, French, German, Dutch, or Japanese. You must restart Oxygen XML Editor for the change to take effect.

Other language

This option sets the language used in the user interface using an interface localization file. For details about creating this file, see *Localizing the User Interface* on page 177. You can use this option to set the language of the user interface to a language that is not shipped with Oxygen XML Editor.

Note: If some interface labels are not rendered correctly after restarting the application, (for example, Chinese or Korean characters are not displayed correctly), make sure that your operating system has the appropriate language pack installed (for example, the East-Asian language pack).

Line separator

This option sets the line separator used when saving files. Use **System Default** to select the normal line separator for your OS. If you want the existing file separator of a file to be maintained, regardless of your current OS, select **Detect the line separator on file open**.

Detect the line separator on file open

When this option is selected, the editor detects the line separator when a file is loaded and it uses it when the file is saved. New files are saved using the line separator defined by the ***Line separator** option*.

Default Internet browser

This option sets the Web browser that Oxygen XML Editor will use to do the following:

- Open (X)HTML or PDF transformation results.
- Open a web page (for example, pointing to specific paragraphs in the W3C recommendation of XML Schema if there are XML Schema validation errors).

If you leave this setting blank, the system default browser will be used.

Open last edited files from project

When this option is selected, Oxygen XML Editor opens the files you had open the last time you used a project whenever you open the application or switch to that project.

Beep on operation finished

When this option is selected, Oxygen XML Editor beeps when a validation or transform action ends. Different tones are used for success and failure. The tones used may depend on the sound settings in your operating system.

Show memory status

When this option is selected, the memoryOxygen XML Editor uses is displayed in the status bar. To free memory, click the **Free unused memory** button located at the right side of the status bar. The memory status bar turns yellow or red when Oxygen XML Editor uses too much memory. You can change the amount of memory available to Oxygen XML Editor by *changing the parameters of the application launcher*.

Check opened files for file system changes

When this option is selected, Oxygen XML Editor checks the content of the all opened editors to see if they have been updated by another application. If the file has changed, Oxygen XML Editor will ask you if you want to reload the file.

Auto update unmodified editors on file system changes

If this option is selected, Oxygen XML Editor automatically updates unmodified editors if the edited file changes externally.

Show Java vendor warning at startup

If this option is selected, Oxygen XML Editor displays a warning on startup if a non-recommended version of the Java virtual machine is being used.

File Chooser Dialog

This options specifies the directory that will be shown when the *Open file dialog box* is displayed. You can choose between:

- **Last visited directory** - The last visited folder will be displayed.
- **Directory of the edited file** - The folder where the currently edited file is stored will be displayed.

Show hidden files and directories

If this option is selected, Oxygen XML Editor shows system hidden files and folders in the file browser dialog box and the folder browser dialog box. This setting is not available on OS X.

Appearance Preferences

This preferences page contains various options that allow you to change the appearance of the user interface of Oxygen XML Editor. To configure the **Appearance** options, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Appearance**.

The following options are available in the **Appearance** preferences page:

Look and Feel

This option allows you to change the graphic style (look and feel) of the user interface. Depending on the operating system, you can choose between various predefined style options.

Theme

This option allows you to choose predefined color themes that will be applied over the entire user interface. You can select between the following:

- **Light** (default theme in Windows)
- **Classic** (default theme in MAC OS)

  Note: In Windows, if a high contrast theme is detected and the **Theme** option is set to `Classic` and the ***Look and Feel** option* is set to `Default` or `Windows`, Oxygen XML Editor inherits the high contrast theme colors that are set in the operating system.
- **Graphite**
- **Ultramarine**

You can also change various appearance related options in other preference pages for the selected theme by clicking on the various links in this section.

Custom Themes

You can also create custom themes to share with others or use in other installations of Oxygen XML Editor. To create a custom theme, follow these steps:

1. Select a **Theme** to use as a base.
2. Configure the desired options in any of the option pages listed in this preferences page.
3. Click **Export** and specify a name for your custom theme. If you save the theme to the default file path, your custom theme will immediately appear in the **Theme** drop-down list. Otherwise, if you save it to another location, you can use the ***Import** button* to make it appear in the drop-down list.

Note: In OS X (starting with Yosemite), if you choose `Graphite` for the **Theme**, it is recommended that you select the **Use dark menu and Dock** option that is found in **System Preferences** > **General**.

Theme preview area

Displays a preview of the current ***Theme** selection* (available for predefined color themes).

Theme management section

Reset

Resets the theme to its default values (this option is available when the theme is modified).

Rename

Changes the name of the theme (not available for default or predefined themes).

Delete

Removes the selected theme (not available for default or predefined themes).

Import

Allows you to import a color theme from an XML theme file. You can use this option to load an exported *custom theme*.

Export

Allows you to export the current color theme into an XML theme file that can then be shared with others or imported into another installation of Oxygen XML Editor.

Configure icon saturation and brightness link

This link is available if you are using a dark *theme* (such as **Graphite** or **Ultramarine**). It opens a dialog box that allows you to configure the saturation and brightness for all the icons in Oxygen XML Editor.

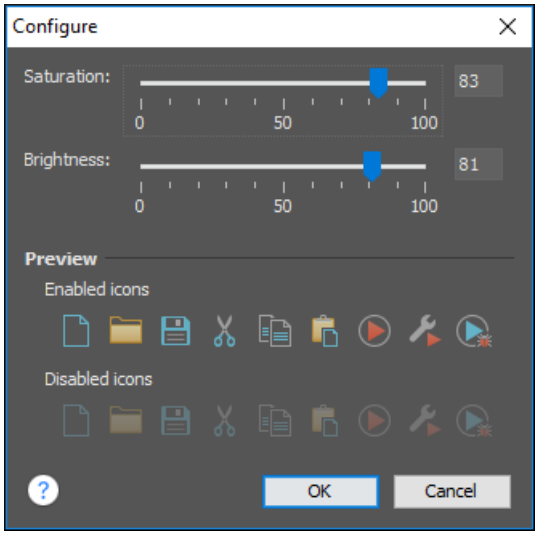


Figure 19: Configure Icon Saturation and Brightness Dialog Box

Colors Preferences

Oxygen XML Editor allows you configure the colors for frames, dialog boxes, controls, and commands. To configure the **Colors**, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Appearance** > **Colors**.

Clicking the color button for any of the options opens a **Choose color** dialog box. It includes several tabs that allow you to configure the color in numerous ways. This page allows you to select and configure the color for the following:

Background Colors

Background

Background color for various general user interface items.

Components background

Background color for various components (such as text fields, views, tables, and dialog boxes).

Components selection background

Background color for the current selections in certain components, such as some views and panes.

Components inactive selection background

Background color for a selection in a view that is not the current focus.

Menus, toolbars and frame backround

Background color for specific components such as menus, toolbars, and the application frame.

Menus and toolbars selection background

(This option is not available for Mac OS) Background color for menu selections and toolbar buttons.

View titles background

Background color for the titles of view and tabs.

Status bar background

Background color of the status bar at the bottom of the editor.

Foreground Colors

Foreground

Foreground color for various general user interface items.

Component selection foreground

Foreground color for the current selection.

Disabled foreground

Foreground color for various components that are not the current focus (such as views other than the currently selected one).

Link foreground

Foreground color for links in views and dialog boxes.

View titles foreground

Foreground color for the title bar of views.

Status bar foreground

Foreground color for the text in the status bar at the bottom of the editor.

Other Colors

Borders and table grids

Color for certain borders and table grid lines.

Text component border

Color for the borders of text fields and drop-down lists.

View/Editor tabs border

Color for the borders of views and tabs.

Scroll bars, chevrons

Color for scroll bars (navigation bars) and chevrons (button to expand a non-visible area).

Separator

Color for the separators in toolbars, menus, and dialog boxes.

Note: You must restart the application for your changes to be applied.

Fonts Preferences

Oxygen XML Editor allows you to choose the fonts to be used in the **Text**, **Design**, and **Grid** editor modes, and fonts for the **Author** mode that are not specified in the associated CSS stylesheet. To configure the font options, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Appearance** > **Fonts**.

The following options are available:

Editor

Allows you to choose the font that will be used in the editor.

Note: On Mac OS X, the default font, Monaco, cannot be rendered in bold.

Author mode default font

This option allows you to choose the default font that will be used in **Author** mode. The default font will be overridden by the fonts specified in any CSS file associated with the opened document.

Schema default font

This option allows you to choose the font to be used in:

- The **Design** mode of the *XML Schema editor*.
- Images with schema diagram fragments that are included in the HTML documentation generated from an XML Schema.

Text antialiasing

This option allows you to set the text anti-aliasing behavior:

- **Default** - Allows the application to use the setting of the operating system, if available.
- **On** - Sets the text anti-aliasing to pixel level.
- **Off** - Disables text anti-aliasing.
- Sub-pixel anti-aliasing modes, such as GASP, LCD_HRGB, LCD_HBGR, LCD_VRGB, and LCD_VBGR.

Text components

This option allows you to choose the font to be used in text boxes within the interface.

GUI

This option allows you to choose the font to be used for user interface labels.

View titles font

This option allows you to choose the font to be used in the titles of the various views that are opened within the interface.

Note: You must restart the application for your changes to be applied.

Related Information:

Changing the Font Size in the Editor on page 292

Application Layout Preferences

Oxygen XML Editor offers various *perspectives* and views that you can arrange in a variety of layouts to suit your needs.

To configure the application layout options, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Application Layout**. The following options are available:

Select application layout

You can choose between the following three layouts:

Default

Uses the default layout for all *perspectives*. Any modification of this layout (such as closing views, displaying views, or a new view arrangement) is saved on exit and reloaded at start-up.

Predefined

Allows you to choose one of the predefined layouts:

- **Advanced** - All views are displayed.
- **Author** - An authoring-oriented layout that includes the following views: ***Project***, ***Archive Browser***, ***DITA Maps Manager***, ***Outline***, ***Attributes***, ***Model***, and ***Elements***.
- **Basic** - Only the ***Project** view* and ***Outline** view* are visible. Recommended when you edit XML content and you need maximum screen space.
- **Schema development** - The ***Project***, ***Component Dependencies***, ***Resource Hierarchy/Dependencies***, ***Outline***, ***Palette***, and ***Attributes*** views are displayed.
- **XQuery development** - The ***Project***, ***Outline***, ***XSLT/XQuery Input***, ***XPath/XQuery Builder***, and ***Transformation Scenarios*** views are displayed.
- **XSLT development** - The ***Project***, ***Component Dependencies***, ***Resource Hierarchy/Dependencies***, ***Outline***, ***Attributes***, ***Model***, ***XSLT/XQuery Input***, ***XPath/XQuery Builder***, and ***Transformation Scenarios*** views are displayed.

Custom

Allows you to specify a custom layout to be used. You can save your preferred layout using **Window** > **Export Layout**, then enter the location of the saved layout file in this setting.

Reset layout at startup

When this option is selected, Oxygen XML Editor forgets any changes made to the layout during a session and reloads the default layout the next time it is started. This is useful when you want to keep a fixed layout from one session to another.

Remember layout changes for each project

When this options is selected, Oxygen XML Editor saves layouts individually for each project. When you switch projects, the layout you last used for that project is loaded automatically.

Allow detaching of editors from main window

When this options is selected, you can drag and drop an editor window outside of the main screen. This is useful especially when you are using two monitors and you want to view files side by side.

Note: If the main screen is maximized, you cannot drag and drop an editor outside of it.

View tab placement

Specifies whether the *View* tabs are located at the top or bottom of the window.

Editor tab placement

Specifies whether the *Editor* tabs are located at the top or bottom of the window.

The changes you make to any layout are preserved between working sessions. The predefined *layout* files are saved in the `preferences` directory of Oxygen XML Editor.

To watch our video demonstration about configuring the user interface of Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/Dockable_Views.html*.

Add-ons Preferences

You can use *add-ons* to enhance the functionality of Oxygen XML Editor. To configure the **Add-ons** options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Add-ons**.

The following options are available in this preferences page:

Enable automatic updates checking

When this option is selected, Oxygen XML Editor will automatically search for available updates.

Add-on Sites URLs

This is a list of the URLs for the add-on sites. You can add, edit, and delete sites in this list by using the buttons below the list.

Document Type Association Preferences

Oxygen XML Editor uses *document type associations* to associate a *document type* with a set of functionality provided by a *framework*. To configure the **Document Type Association** options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Document Type Association**.

The following actions are available in this preferences page:

Discover more frameworks by using add-ons update sites

Click on this link to specify URLs for *framework* add-on update sites.

Document Type Table

This table presents the currently defined *frameworks* (*document type associations*), sorted by priority and alphabetically. Each edited document type has a set of association rules (used by the application to detect the proper document type association to use for an opened XML document). A rule is described by:

- **Namespace** - Specifies the namespace of the root element from the association rules set (** (any)* by default). If you want to apply the rule only when the root element has no namespace, leave this field empty (remove the **ANY_VALUE** string).
- **Root local name** - Specifies the local name of the root element (** (any)* by default).

- **File name** - Specifies the name of the file (** (any)* by default).
- **Public ID** - Represents the Public ID of the matched document.
- **Java class** - Presents the name of the Java class, which is used to determine if a document matches the rule. This Java class should implement the `ro.sync.ecss.extensions.api.DocumentTypeCustomRuleMatcher` interface.

New

Opens a ***Document type*** *configuration dialog box* that allows you to add a new *framework*.

Edit

Opens a ***Document type*** *configuration dialog box* that allows you to edit an existing *framework*.

Note: If you try to edit an existing *framework* when you do not have write permissions to its storage location, a dialog box will be shown asking if you want to extend it.

Duplicate

Opens a ***Document type*** *configuration dialog box* that allows you to duplicate the configuration of an existing *framework*. This will create a snapshot of the *framework* in its current form. It is merely a copy of the document type and will not *evolve* along with the base document type like the **Extend** action does.

Extend

Opens a ***Document type*** *configuration dialog box* that allows you to extend an existing *framework*. You can add or remove functionality starting from a base document type. All of these changes will be saved as a patch. When the base document type is modified and evolves (for example, from one application version to another) the extension will evolve along with the base document type, allowing it to use the new actions added in the base document type.

Delete

Deletes the selected *framework* (document type).

Enable DTD/XML Schema processing in document type detection

When this option is selected (default value), the matching process also examines the DTD/XML Schema associated with the document. For example, the fixed attributes declared in the DTD for the root element are also analyzed, if this is specified in the association rules. This is especially useful if you are writing DITA customizations. DITA topics and maps are also matched by looking for the `DITAArchVersion` attribute of the root element. This attribute is specified as `default` in the DTD and it is detected in the root element, helping Oxygen XML Editor to correctly match the DITA customization.

Only for local DTDs/XML Schemas

When this option is selected (default value), only the local DTDs / XML Schemas will be processed.

Enable DTD/XML Schema caching

When this option is selected (default value), the associated DTDs or XML Schema are cached when parsed for the first time, improving performance when opening new documents with similar schema associations.

Related Information:

Extending and Sharing a Framework (Document Type) on page 1240

Locations Preferences

Oxygen XML Editor allows you to change the location where *frameworks* (document types) are stored, and to specify additional *framework* directories. The **Locations** preferences page allows you to specify the main `frameworks` folder location. You can choose between the **Default** directory (*`[OXYGEN_INSTALL_DIR]/`*`frameworks`) or a **Custom** specified directory. You can also change the current `frameworks` folder location value using the `com.oxygenxml.editor.frameworks.url` system property set in either the *`.vmoptions` configuration files* or in the *startup scripts*.

A list of additional `frameworks` directories can also be specified. The application will look in each of those folders for additional document type configurations to load. Use the **Add**, **Edit** and **Delete** buttons to manage the list of folders.

A document type configuration (*framework*) can be loaded from the following locations:

- Internal preferences - The document type configuration is stored in the application ***Internal*** *preferences*.

- Additional *framework* directories - The document type configuration is loaded from one of the specified **Additional frameworks directories** list.
- Add-ons - An *add-on* can contribute a *framework*. You can manage the add-ons locations in the *Add-ons preferences page*.
- The `frameworks` folder - The main folder containing *framework* configurations.

All loaded document type configurations are first sorted by priority, then by document type name and then by load location (in the exact order specified above). When an XML document is opened, the application chooses the first document type configuration from the sorted list that matches the specific document.

All loaded document type configurations are first sorted by priority, then by document type.

Document Type Configuration Dialog Box

The **Document Type** Configuration dialog box allows you to create or edit a *framework* (document type). It is displayed when you use the **New**, **Edit**, **Duplicate**, or **Extend** buttons in the ***Document Type Association*** *preferences page* (*open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Document Type Association**).

The configuration dialog box includes the following fields and sections:

- **Name** - The name of the *framework*. This will be displayed as its name in the **Document Type** column in the ***Document Type Association*** *preferences page*.
- **Priority** - Depending on the priority level, Oxygen XML Editor establishes the order in which the existing *frameworks* are evaluated to determine the type of a document you are opening. It can be one of the following: Lowest, Low, Normal, High, or Highest. You can set a higher priority to *frameworks* you want to be evaluated first.
- **Description** - A detailed description of the *framework*.
- **Storage** - Displays the type of location where the *framework* configuration file is stored. Can be one of: **External** (*framework* configuration is saved in a file) or **Internal** (*framework* configuration is stored in the application's internal options).

  Note: If you set the **Storage** to **Internal** and the *framework* configuration is already stored in a *framework* file, the file content is saved in the application's internal options and the file is removed.
- **Initial edit mode** - Sets the default edit mode when you open a document for the first time.
- **Configuration Tabs** - The bottom section of the dialog box includes various tabs where you can configure numerous options for the *framework*.

Related Information:

Sharing a Framework (Document Type) on page 1239

Association Rules Tab

By combining multiple association rules you can instruct Oxygen XML Editor to identify the type of a document. An Oxygen XML Editor *association rule* holds information about **Namespace**, **Root local name**, **File name**, **Public ID**, **Attribute**, and **Java class**. Oxygen XML Editor identifies the type of a document when the document matches at least one of the *association rules*. This tab give you access to a **Document type rule** dialog box that you can use to create *association rules* that activate on any document matching all the criteria defined in the dialog box.

To create a new association rule, click the **New** button at the bottom of the **Association Rules** tab, or to edit an existing rule, click the **Edit** button.

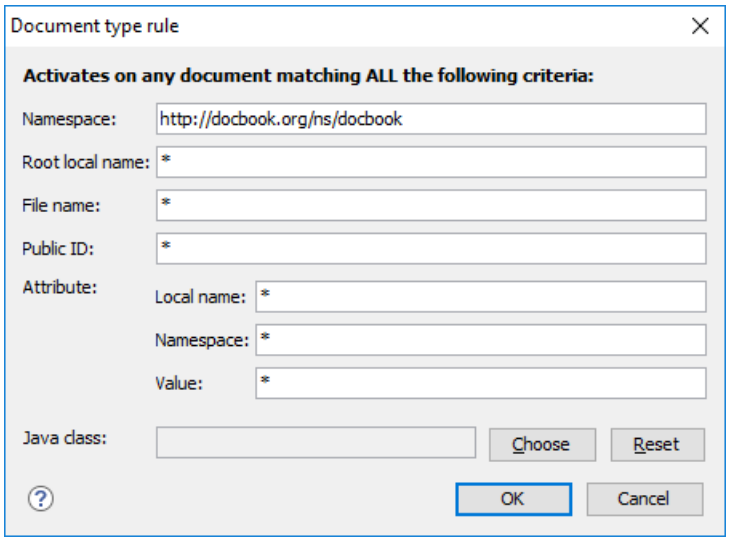


Figure 20: Document Type Rule Dialog Box

Tip: You can use wildcards (*?* and *) or *editor variables* in the **Document Type Rule** dialog box, and you can enter multiple values by separating them with a comma.

To open the **Association Rules** tab of the **Document type** configuration dialog box, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New**, **Edit**, **Duplicate**, or **Extend** button*, and click on the **Association Rules** tab.

In the **Association rules** tab, you can perform the following actions:

New

Opens the **Document type rule** dialog box allowing you to create *association rules*.

Edit

Opens the **Document type rule** dialog box allowing you to edit the properties of the currently selected *association rule*.

Delete

Deletes the currently selected *association rules* from the list.

Move Up

Moves the selected *association rule* up one spot in the list.

Move Down

Moves the selected *association rule* down one spot in the list.

Schema Tab

In the **Schema** tab, you can specify a schema that Oxygen XML Editor uses if an XML document does not contain a schema declaration and no default validation scenario is associated with it.

To open the **Schema** tab of the **Document type** configuration dialog box, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New**, **Edit**, **Duplicate**, or **Extend** button*, and click on the **Schema** tab.

This tab includes the following options for defining a schema to be used if no schema is detected in the XML file:

Schema type

Use this drop-down list to select the type of schema.

Schema URI

You can specify the URI of the schema file. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.

Tip: It is a good practice to store all resources in the *framework* directory and use the *`${framework}` editor variable* to reference them. This is a recommended approach to designing a self-contained document type that can be easily maintained and shared between multiple users.

Classpath Tab

The **Classpath** tab displays a list of folders and *JAR* libraries that hold implementations for API extensions, implementations for custom **Author** mode operations, various resources (such as stylesheets), and *framework* translation files. Oxygen XML Editor loads the resources looking in the folders in the order they appear in the list.

To open the **Classpath** tab of the **Document type** configuration dialog box, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New**, **Edit**, **Duplicate**, or **Extend** button*, and click on the **Classpath** tab.

The **Classpath** tab includes the following actions:

New

Opens a dialog box that allows you to add a resource to the table in the **Classpath** tab. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.

Tip: The path can also contain wildcards (for example, `${framework}/lib/*.jar`).

Edit

Opens a dialog box that allows you to edit a resource in the **Classpath** tab. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.

Tip: The path can also contain wildcards (for example, `${framework}/lib/*.jar`).

Delete

Deletes the currently selected resource from the list.

Move Up

Moves the selected resource up one spot in the list.

Move Down

Moves the selected resource down one spot in the list.

Use parent classloader from plugin with ID

Use this option to specify the ID of a *plugin*. The current *framework* has access to the classes loaded for the *plugin*.

Related Information:

Extensions Tab on page 75
Author Tab on page 61
Localizing Frameworks on page 1203

Author Tab

The **Author** tab is a container that holds information regarding the CSS file used to render a document in the **Author** mode, and regarding *framework*-specific actions, menus, contextual menus, toolbars, and content completion list of proposals.

To open the **Author** tab of the **Document type** configuration dialog box, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New**, **Edit**, **Duplicate**, or **Extend** button*, and click on the **Author** tab.

The options that you configure in the **Author** tab are grouped in subtabs.

CSS Subtab

The **CSS** subtab contains the CSS files that Oxygen XML Editor uses to render a document in the **Author** mode. In this subtab, you can set *main* and *alternate* CSS files. When you are editing a document in the **Author** mode, you can switch between these CSS files from the **Styles** drop-down menu on the **Author Styles** toolbar.

To open the **CSS** subtab, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New, Edit, Duplicate, or Extend** button*, click on the **Author** tab, and then the **CSS** subtab.

The following actions are available in the **CSS** subtab:

New

Opens a dialog box that allows you to add a CSS file. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.

Edit

Opens a dialog box that allows you to edit the current selection.

Delete

Deletes the currently selected CSS file.

Move Up

Moves the selected CSS file up in the list.

Move Down

Moves the selected CSS file down in the list.

Enable multiple selection of alternate CSSs

Allows users to apply multiple alternate styles, as layers, over the main CSS style. This option is selected by default for DITA document types.

If there are CSSs specified in the document then

You can choose between the following options for controlling how the CSS files that are set in this subtab will be handled if a CSS is specified in the document itself:

- **Ignore CSSs from the associated document type** - The CSS files set in this CSS subtab are overwritten by the CSS files specified in the document itself.
- **Merge them with CSSs from the associated document type** - The CSS files set in this CSS subtab are merged with the CSS files specified in the document itself.

Related Information:

CSS Stylesheet on page 1165
Configuring and Managing Multiple CSS Styles on page 1256

Actions Subtab

The **Actions** subtab contains a sortable table that includes all the *framework*-specific actions. Each action has a unique ID, a name, a description, and a shortcut key.

To open the **Actions** subtab, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New, Edit, Duplicate, or Extend** button*, click on the **Author** tab, and then the **Actions** subtab.

The following actions are available in this subtab:

New

Opens the ***Action** dialog box* that allows you to add an action.

Duplicate

Duplicates the currently selected action.

Edit

Opens the ***Action** dialog box* that allows you to edit the existing action.

Delete

Deletes the currently selected action.

Action Dialog Box

To edit an existing document type action or create a new one, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New, Edit, Duplicate, or Extend** button*, click on the

Author tab, and then the **Actions** subtab. At the bottom of this subtab, click **New** to create a new action, or **Edit** to modify an existing one.

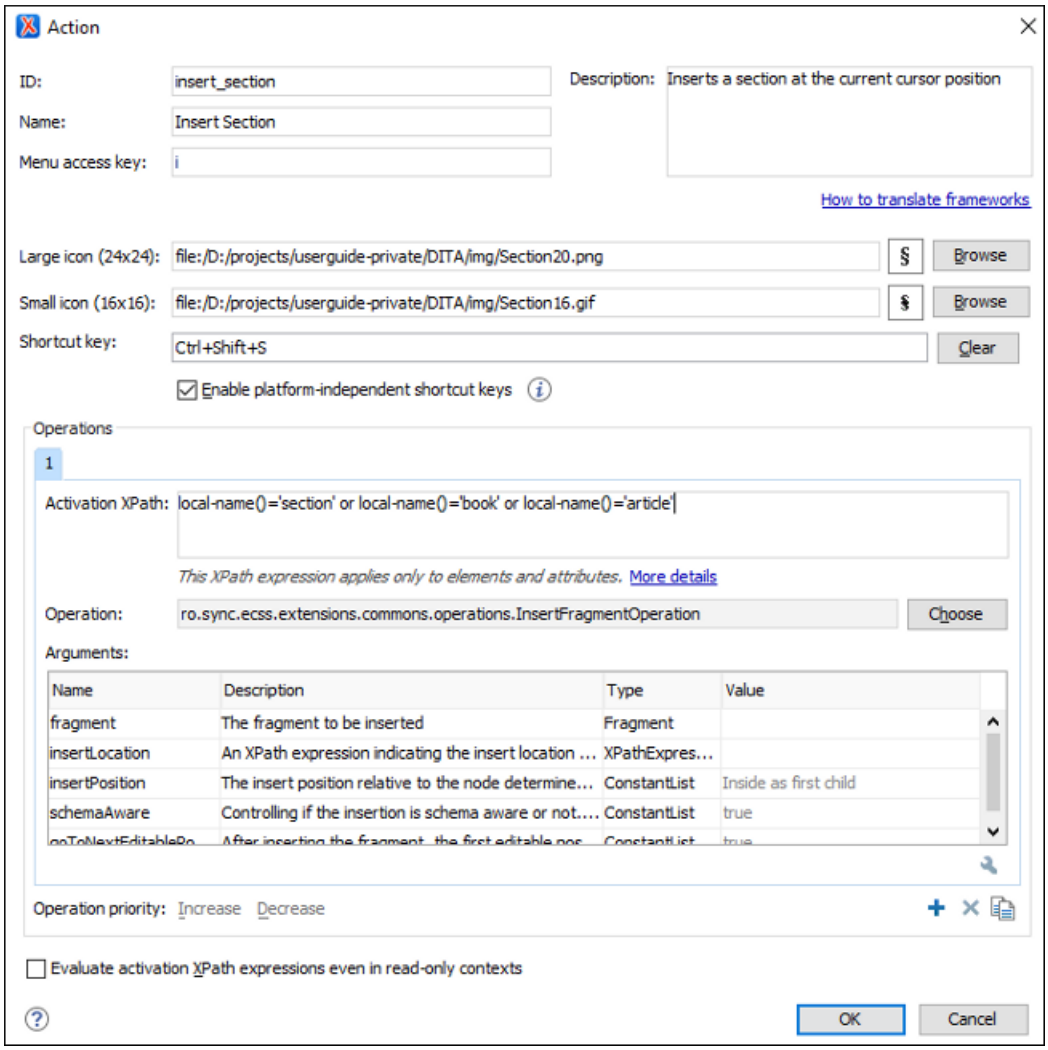


Figure 21: Action Dialog Box

The following options are available in the **Action** dialog box:

ID

Specifies a unique action identifier.

Name

Specifies the name of the action. This name is displayed as a tooltip or as a menu item.

Tip: You can use the *${i18n('key')} editor variable* to allow for multiple translations of the name.

Menu access key

In Windows, you can access menus by holding down **Alt** and pressing the keyboard key that corresponds to the *Letter* that is underlined in the name of the menu. Then, while still holding down **Alt**, you can select submenus and menu action the same way by pressing subsequent corresponding keys. You can use this option to specify the *Letter* in the name of the action that can be used to access the action.

Description

A description of the action. This description is displayed as a tooltip when hovering over the action.

Tip: You can use the *${i18n('key')} editor variable* to allow for multiple translations of the description.

How to translate frameworks link

Use this link to see more information about *Localizing Frameworks* on page 1203.

Large icon

Allows you to select an image for the icon that Oxygen XML Editor uses for the toolbar action.

Tip: A good practice is to store the image files inside the *framework* directory and use the *`${frameworks}` editor variable* to make the image relative to the *framework* location. If the images are bundled in a *jar* archive (for instance, along with some Java operations implementation), it is convenient to reference the images by their relative path location in the *class-path*.

Small icon

Allows you to select an image for the icon that Oxygen XML Editor uses for the contextual menu action.

Note: If you are using a Retina or HiDPI display, Oxygen XML Editor automatically searches for higher resolution icons in the path specified in both the **Large icon** and **Small icon** options. For more information, see *Adding Retina/HiDPI Icons in a Framework* on page 1196.

Shortcut key

This field allows you to configure a shortcut key for the action that you are editing. The + character separates the keys.

Enable platform-independent shortcut keys

If this checkbox is selected, the shortcut that you specify in this field is platform-independent and the following modifiers are used:

- `M1` represents the **Command** key on MacOS X, and the **Ctrl** key on other platforms.
- `M2` represents the **Shift** key.
- `M3` represents the **Option** key on MacOS X, and the **Alt** key on other platforms.
- `M4` represents the **Ctrl** key on MacOS X, and is undefined on other platforms.

Operations section

In this section of the **Action** dialog box, you configure the functionality of the action that you are editing. An action has one or more operation modes. The evaluation of an XPath expression activates an operation mode. The first selected operation mode is activated when you trigger the action. The scope of the XPath expression must consist only of element nodes and attribute nodes of the edited document. Otherwise, the XPath expression does not return a match and does not fire the action. For more details see: *Activate Multiple Functions for Actions using XPath Expressions* on page 64.

The following options are available in this section:

Activation XPath

An XPath 2.0 expression that applies to elements and attributes. For more details see:*Activate Multiple Functions for Actions using XPath Expressions* on page 64.

Operation

Specifies the invoked operation.

Arguments

Specifies the arguments of the invoked operation. The **Edit** at the bottom of the table allows you to edit the arguments of the operation.

Operation priority

Increases or decreases the priority of an operation. The operations are invoked in the order of their priority. If multiple XPath expressions are true, the operation with the highest priority is invoked.

- **Add** - Adds an operation.
- **Remove** - Removes an operation.
- **Duplicate** - Duplicates an operation.

Evaluate activation XPath expressions even in read-only cotnexts

If this checkbox is selected, the action can be invoked even when the cursor is placed in a read-only location.

Activate Multiple Functions for Actions using XPath Expressions

An **Author** mode action can have multiple functions, each function invoking an **Author** mode operation with certain configured parameters. Each function of an action has an XPath 2.0 expression for activating it.

For each function of an action, the application will check if the XPath expression is fulfilled (when it returns a not empty nodes set or a *true* result). If it is fulfilled, the operation defined in the function will be executed.

Three special XPath extension functions are provided:

- *`oxy:allows-child-element()`* - Use this function to check whether or not an element is valid child element in the current context, according to the associated schema.
- *`oxy:allows-global-element()`* - Use this function to check whether or not an element is a valid global element for the current *framework*, according to the associated schema.
- *`oxy:current-selected-element()`* - Use this function to get the currently selected element.
- *`oxy:is-required-element()`* - Use this function to check if the element returned by the given XPath expression is required (based on the rules declared in the schema).

`oxy:allows-child-element()` Function

The `oxy:allows-child-element()` function allows you to check whether or not an element that matches the arguments of the function is valid as a child of the element at the current cursor position, according to the associated schema. It is evaluated at the cursor position and has the following signature:

`oxy:allows-child-element($childName, ($attributeName, $defaultAttributeValue, $contains?)?)`

The following parameters are supported:

`childName`

The name of the element that you want to check if it is valid in the current context. Its value is a string that supports the following forms:

- The child element with the specified local name that belongs to the default namespace.

  ```
  oxy:allows-child-element("para")
  ```

  The above example verifies if the para element (of the default namespace) is allowed in the current context.
- The child element with the local name specified by any namespace.

  ```
  oxy:allows-child-element("*:para")
  ```

  The above example verifies if the para element (of any namespace) is allowed in the current context.
- A prefix-qualified name of an element.

  ```
  oxy:allows-child-element("prefix:para")
  ```

  The prefix is resolved in the context of the element where the cursor is located. The function matches on the element with the para local name from the previous resolved namespace. If the prefix is not resolved to a namespace, the function returns a value of `false`.
- A specified namespace-URI-qualified name of an element.

  ```
  oxy:allows-child-element("{namespaceURI}para")
  ```

  The `namespaceURI` is the namespace of the element. The above example verifies if the para element (of the specified namespace) is allowed in the current context.
- Any element.

  ```
  oxy:allows-child-element("*")
  ```

  The above function verifies if any element is allowed in the current context.

  Note: A common use case of `oxy:allows-child-element("*")` is in combination with the `attributeName` parameter.

`attributeName`

The attribute of an element that you want to check if it is valid in the current context. Its value is a string that supports the following forms:

- The attribute with the specified name from no namespace.

  ```
  oxy:allows-child-element("*", "class", " topic/topic ")
  ```

  The above example verifies if an element with the `class` attribute and the default value of this attribute (that contains the `topic/topic` string) is allowed in the current context.
- The attribute with the local name specified by any namespace.

  ```
  oxy:allows-child-element("*", "*:localname", " topic/topic ")
  ```

- A qualified name of an attribute.

  ```
  oxy:allows-child-element("*", "prefix:localname", " topic/topic ")
  ```

  The prefix is resolved in the context of the element where the cursor is located. If the prefix is not resolved to a namespace, the function returns a value of `false`.

`defaultAttributeValue`

A string that represents the default value of the attribute. Depending on the value of the next parameter, the default value of the attribute must either contain this value or be equal with it.

`contains`

An optional boolean. The default value is `true`. For the `true` value, the default value of the attribute must contain the `defaultAttributeValue` parameter. If the value is `false`, the two values must be the same.

`oxy:allows-global-element()` Function

The `oxy:allows-global-element()` function allows you to check whether or not an element that matches the arguments of the function is valid for the current *framework*, according to the associated schema. It has the following signature:

`oxy:allows-global-element($elementName, ($attributeName, $defaultAttributeValue, $contains?)?)`

The following parameters are supported:

`elementName`

The name of the element that you want to check if it is valid in the current *framework*. Its value is a string that supports the following forms:

- The element with the specified local name that belongs to the default namespace.

  ```
  oxy:allows-global-element("para")
  ```

  The above example verifies if the para element (of the default namespace) is allowed in the current *framework*.
- The element with the local name specified by any namespace.

  ```
  oxy:allows-global-element("*:para")
  ```

  The above example verifies if the para element (of any namespace) is allowed in the current *framework*.
- A prefix-qualified name of an element.

  ```
  oxy:allows-global-element("prefix:para")
  ```

  The prefix is resolved in the context of the *framework*. The function matches on the element with the para local name from the previous resolved namespace. If the prefix is not resolved to a namespace, the function returns a value of `false`.
- A specified namespace-URI-qualified name of an element.

  ```
  oxy:allows-global-element("{namespaceURI}para")
  ```

The `namespaceURI` is the namespace of the element. The above example verifies if the para element (of the specified namespace) is allowed in the current *framework*.

- Any element.

```
oxy:allows-global-element("*")
```

The above function verifies if any element is allowed in the current *framework*.

`attributeName`

The attribute of an element that you want to check if it is valid in the current *framework*. Its value is a string that supports the following forms:

- The attribute with the specified name from no namespace.

```
oxy:allows-global-element("*", "class", " topic/topic ")
```

The above example verifies if an element with the `class` attribute and the default value of this attribute (that contains the `topic/topic` string) is allowed in the current *framework*.

- The attribute with the local name specified by any namespace.

```
oxy:allows-global-element("*", "*:localname", " topic/topic ")
```

- A qualified name of an attribute.

```
oxy:allows-global-element("*", "prefix:localname", " topic/topic ")
```

The prefix is resolved in the context of the *framework*. If the prefix is not resolved to a namespace, the function returns a value of `false`.

`defaultAttributeValue`

A string that represents the default value of the attribute. Depending on the value of the next parameter, the default value of the attribute must either contain this value or be equal with it.

`contains`

An optional boolean. The default value is `true`. For the `true` value, the default value of the attribute must contain the `defaultAttributeValue` parameter. If the value is `false`, the two values must be the same.

`oxy:current-selected-element()` Function

This function returns the fully selected element. If no element is selected, the function returns an empty sequence.

Example: *oxy:current-selected-element* Function

```
oxy:current-selected-element()[self::p]/b
```

This example returns the b elements that are children of the currently selected p element.

`oxy:is-required-element()` Function

This function checks if the element returned by the given XPath expression is required (based on the rules declared in the schema). It has only one argument, an XPath expression, and the XPath expression must be written in such a way that it returns a single element.

Example: *oxy:is-required-element* Function

```
oxy:is-required-element(.)
```

This example would check to see if the current element is required by the schema.

Localizing Frameworks

Oxygen XML Editor supports *framework* localization (translating *framework* actions, buttons, and menu entries to various languages). This lets you develop and distribute a *framework* to users that speak other languages without changing the distributed *framework*. Changing the language used in Oxygen XML Editor in the Global preferences page is enough to set the right language for each *framework*.

To localize the content of a *framework*, create a `translation.xml` file that contains all the translation (key, value) mappings. The `translation.xml` has the following format:

```
<translation>
    <languageList>
        <language description="English" lang="en_US"/>
        <language description="German" lang="de_DE"/>
        <language description="French" lang="fr_FR"/>
    </languageList>
    <key value="list">
        <comment>List menu item name.</comment>
        <val lang="en_US">List</val>
        <val lang="de_DE">Liste</val>
        <val lang="fr_FR">Liste</val>
    </key>
.......................
</translation>
```

Oxygen XML Editor matches the GUI language with the language set in the `translation.xml` file. If this language is not found, the first available language declared in the `languagelist` tag for the corresponding *framework* is used.

Add the directory where this file is located to the **Classpath** list corresponding to the edited document type.

After you create this file, you can use the keys defined in it to customize the name and description of the following:

- Actions
- Menu entries
- Contextual menus
- Toolbars
- Static CSS content

For example, if you want to localize the bold action, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Document Type Association**. Use the **New** or **Edit** button to open the ***Document type** configuration dialog box*, go to **Author** > **Actions**, and rename the bold action to `${i18n(translation_key)}`. Actions with a name format other than `${i18n(translation_key)}` are not localized. `Translation_key` corresponds to the key from the translation.xml file.

Now open the `translation.xml` file and edit the translation entry if it exists or create one if it does not exist. This example presents an entry in the translation.xml file:

```
<key value="translation_key">
        <comment>Bold action name.</comment>
        <val lang="en_US">Bold</val>
        <val lang="de_DE">Bold</val>
        <val lang="fr_FR">Bold</val>
    </key>
```

To use a description from the `translation.xml` file in the Java code used by your custom *framework*, use the new *`ro.sync.ecss.extensions.api.AuthorAccess.getAuthorResourceBundle()`* API method to request the associated value for a certain key. This allows all the dialog boxes that you present from your custom operations to have labels translated in multiple languages.

You can also reference a key directly in the CSS content:

```
title:before{
    content:"${i18n(title.key)} : ";
}
```

Note: You can enter any language you want in the `languagelist` tag and any number of keys.

The `translation.xml` file for the DocBook *framework* is located here:*`[OXYGEN_INSTALL_DIR]`*`/frameworks/docbook/i18n/translation.xml`. In the **Classpath** list corresponding to the DocBook document type the following entry was added: `${framework}/i18n/`.

To see how the DocBook actions are defined to use these keys for their name and description, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Document Type Association** > **Author** > **Actions**. If you look in the Java class `ro.sync.ecss.extensions.docbook.table.SADocbookTableCustomizerDialog` available in the *oxygen-sample-framework* module of the *Oxygen SDK* Maven archetype, you can see how the new

`ro.sync.ecss.extensions.api.AuthorResourceBundle` API is used to retrieve localized descriptions for various keys.

Menu Subtab

In the **Menu** subtab, you can configure which actions will appear in the *framework*-specific menu. The subtab is divided in two sections: **Available actions** and **Current actions**.

To open the **Menu** subtab, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New, Edit, Duplicate,** or **Extend** button*, click on the **Author** tab, and then the **Menu** subtab.

The **Available actions** section presents a table that displays the actions defined in the **Actions** subtab, along with their icon, ID, and name. The **Current actions** section holds the actions that are displayed in the Oxygen XML Editor menu. To add an action in this section as a sibling of the currently selected action, use the **Add as sibling** button. To add an image in this section as a child of the currently selected action use the **Add as child** button.

The following actions are available in the **Current actions** section:

Edit

Edits an item.

Remove

Removes an item.

Move Up

Moves an item up.

Move Down

Moves an item down.

Contextual Menu Subtab

In the **Contextual menu** subtab you configure what *framework*-specific action the *Content Completion Assistant* proposes. The subtab is divided in two sections: **Available actions** and **Current actions**.

To open the **Contextual Menu** subtab, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New, Edit, Duplicate,** or **Extend** button*, click on the **Author** tab, and then the **Contextual Menu** subtab.

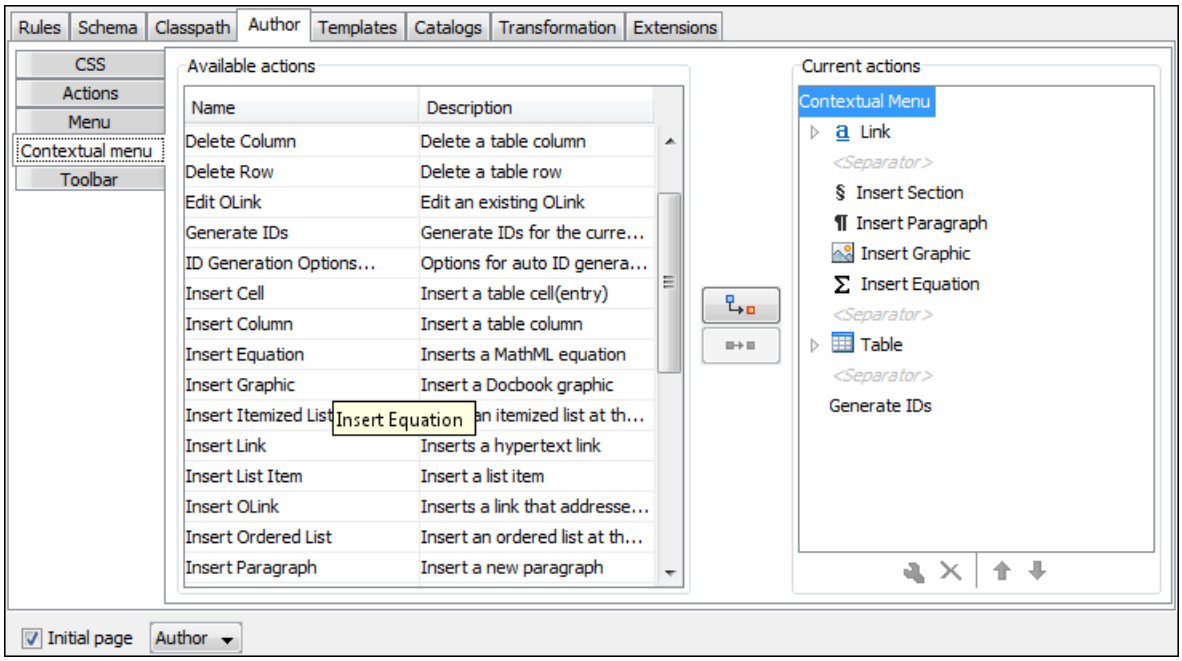


Figure 22: Contextual Menu Subtab

The **Available actions** section presents a table that displays the actions defined in the **Actions** subtab, along with their icon, ID, and name. The **Current actions** section contains the actions that are displayed in the contextual menu for documents that belong to the edited *framework*.

The following actions are available in this subtab:

Add as sibling

Adds the selected action or submenu from the **Available actions** section to the **Current actions** section as a sibling of the selected action.

Add as child

Adds the selected action or submenu from the **Available actions** section to the **Current actions** section as a child of the selected action.

Edit

This option is available for container (submenu) items that are listed in the **Current actions** section. It opens a configuration dialog box that allows you to edit the selected container (submenu).

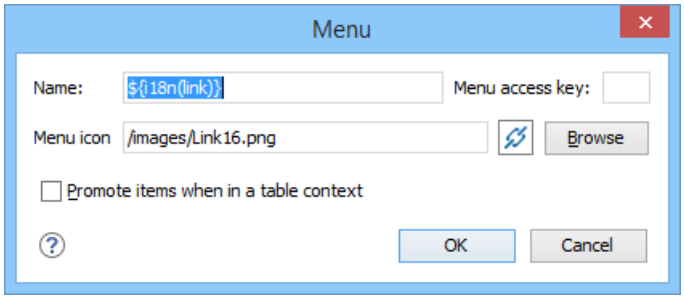


Figure 23: Menu Action Configuration Dialog Box

The following options are available in this dialog box:

Name

Specifies the name of the action. This name is displayed as a tooltip or as a menu item.

Tip: You can use the *${i18n('key')} editor variable* to allow for multiple translations of the name.

Menu access key

In Windows, you can access menus by holding down **Alt** and pressing the keyboard key that corresponds to the *Letter* that is underlined in the name of the menu. Then, while still holding down **Alt**, you can select submenus and menu action the same way by pressing subsequent corresponding keys. You can use this option to specify the *Letter* in the name of the action that can be used to access the action.

Menu icon

Allows you to select an image for the icon that Oxygen XML Editor uses for the container (submenu).

Promote items when in a table context

If this option is selected, when invoking the contextual menu from within a table, all the actions in this container (submenu) will be promoted to the main level in the contextual menu. Actions and submenus that are not promoted are still available in the **Other actions** submenu when invoking the contextual menu within a table.

Remove

Removes the selected action or submenu from the **Current actions** section.

Move Up

Moves the selected item up in the list.

Move Down

Moves the selected item down in the list.

Toolbar Subtab

In the **Toolbar** subtab you configure what *framework*-specific action the Oxygen XML Editor toolbar holds. The subtab is divided in two sections: **Available actions** and **Current actions**.

To open the **Toolbar** subtab, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New**, **Edit**, **Duplicate**, or **Extend** button*, click on the **Author** tab, and then the **Toolbar** subtab.

The **Available actions** section presents a table that displays the actions defined in the **Actions** subtab, along with their icon, ID, and name. The **Current actions** section holds the actions that are displayed in the Oxygen XML Editor toolbar when you work with a document belonging to the edited *framework*. To add an action in this section as a sibling of the currently selected action, use the **Add as sibling** button. To add an action in this section as a child of the currently selected action use the **Add as child** button.

The following actions are available in the **Current actions** section:

Edit
Edits an item.

Remove
Removes an item.

Move Up
Moves an item up.

Move Down
Moves an item down.

Content Completion Subtab

In the **Content Completion** subtab you configure what *framework*-specific the *Content Completion Assistant* proposes. The subtab is divided in two sections: **Available actions** and **Current actions**.

To open the **Content Completion** subtab, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New**, **Edit**, **Duplicate**, or **Extend** button*, click on the **Author** tab, and then the **Content Completion** subtab.

Available and Current Actions

The **Available actions** section presents a table that displays the actions defined in the **Actions** subtab, along with their icon, ID, and name. The **Current actions** section holds the actions that the *Content Completion Assistant* proposes when you work with a document belonging to the edited *framework*. To add an action in this section as a sibling of the currently selected action, use the **Add as sibling** button. To add an action in this section as a child of the currently selected action use the **Add as child** button.

The following actions are available in the **Current actions** section:

Edit
Edits an item.

Remove
Removes an item.

Move Up
Moves an item up.

Move Down
Moves an item down.

Filter Table

The **Filter** section presents a table that allows you to add elements to be filtered from the *Content Completion Assistant* or from some specific helper views or menus. Use the **Add** button to add more filters to the table,

the **Edit** button to modify an existing item in the table, or the **Remove** button to remove a filtered item. The **Add** and **Edit** buttons open a **Remove item** dialog box.

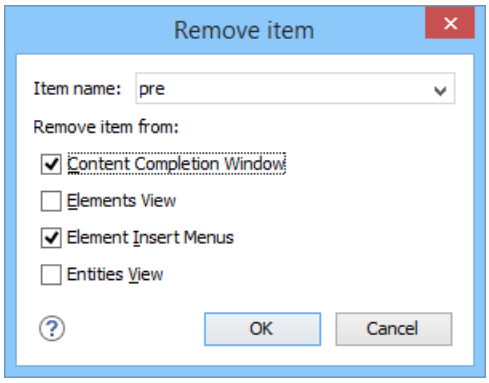


Figure 24: Remove Item Dialog Box

Use this dialog box to add or configure the elements that will be filtered:

Item name

Use this text field to enter the name of the element to be filtered. The drop-down list also includes a few special content completion actions that can be filtered (**<SPLIT>** and **<ENTER>**).

Remove item from

You can choose to filter the element from any of the following:

- **Content Completion Window** - The element will not appear in the *Content Completion Assistant*.
- **Elements View** - The element will not appear in the ***Elements** view*.
- **Element Insert Menus** - The element will not appear in the **Append Child**, **Insert Before**, or **Insert After** menus that are available in certain contextual menus (for example, the contextual menu of the ***Outline** view*).
- **Entities View** - The element will not appear in the ***Entities** view*.

Related Information:

Customizing the Content Completion Assistant on page 1241

Templates Tab

The **Templates** tab specifies a list of directories where new file templates are located. These file templates are gathered from all the document types and presented in the various folders inside the **Framework templates** folder in the ***New** document wizard*.

To open the **Templates** tab of the **Document type** configuration dialog box, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New**, **Edit**, **Duplicate**, or **Extend** button*, and click on the **Templates** tab.

The **Templates** tab includes the following actions:

New

Opens a dialog box that allows you to specify the path to the directory of the template. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.

Tip: The path can also contain wildcards. For example, using `${frameworkDir}/templates/*` would add all the template folders found inside the `templates` directory.

Edit

Opens a dialog box that allows you to edit the path of the selected template. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.

Tip: The path can also contain wildcards. For example, using `${frameworkDir}/templates/*` would add all the template folders found inside the `templates` directory.

Delete

Deletes the currently selected template from the list.

Move Up

Moves the selected template up one spot in the list.

Move Down

Moves the selected template down one spot in the list.

Catalogs Tab

The **Catalogs** tab specifies a list of *XML Catalogs*, specifically for the edited *framework*, that are added to list of catalogs that Oxygen XML Editor uses to resolve resources.

To open the **Catalogs** tab of the **Document type** configuration dialog box, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New**, **Edit**, **Duplicate**, or **Extend** button*, and click on the **Catalogs** tab.

You can perform the following actions:

Add

Opens a dialog box that allows you to add a catalog to the list. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.

Edit

Opens a dialog box that allows you to edit the path of an existing catalog.

Delete

Deletes the currently selected catalog from the list.

Move Up

Moves the selected catalog up one spot in the list.

Move Down

Moves the selected catalog down one spot in the list.

Transformation Tab

In the **Transformation** tab, you can configure the transformation scenarios associated with the particular *framework* you are editing. These transformation scenarios are presented in the ***Configure Transformation Scenarios** dialog box* when transforming a document and you can specify which scenarios will be used by default for a particular document type.

To open the **Transformation** tab of the **Document type** configuration dialog box, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New**, **Edit**, **Duplicate**, or **Extend** button*, and click on the **Transformation** tab.

The **Transformation** tab offers the following options:

Default checkbox

You can set one or more of the scenarios listed in this tab to be used as the default transformation scenario when another specific scenario is not specified. The scenarios that are set as default are rendered bold in the ***Configure Transformation Scenarios** dialog box*.

New

Opens the **New scenario** dialog box allowing you to *create a new transformation scenario for the particular document type*.

Duplicate

Allows you to duplicate the configuration of an existing transformation scenario. It opens the **Edit scenario** dialog box where you can *configure the properties of the duplicated scenario*.

Edit

Opens the **Edit scenario** dialog box allowing you to *edit the properties of the currently selected transformation scenario*.

Delete

Deletes the currently selected transformation scenario.

Import scenarios

Imports transformation scenarios.

Export selected scenarios

Export transformation scenarios.

Move Up

Moves the selection to the previous scenario.

Move Down

Moves the selection to the next scenario.

Validation Tab

In the **Validation** tab, you can configure the validation scenarios associated with the particular *framework* you are editing. These validation scenarios are presented in the **Configure Validation Scenarios** dialog box when validating a document and you can specify which scenarios will be used by default for a particular document type.

Note: If a *master file* is associated with the current file, the validation scenarios defined in the *master file*, along with any Schematron schema defined in the default scenarios for that particular *framework*, are used for the validation. These take precedence over other types of validation units defined in the default scenarios for the particular *framework*. For more information on *master files*, see *Master Files Support* on page 286 or *Working with Modular XML Files in the Master Files Context* on page 481.

To open the **Validation** tab of the **Document type** configuration dialog box, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New**, **Edit**, **Duplicate**, or **Extend** button*, and click on the **Validation** tab.

The **Validation** tab offers the following options:

Default checkbox

You can set one or more of the scenarios listed in this tab to be used as the default validation scenario when another specific scenario is not specified in the validation process. The scenarios that are set as default are rendered bold in the **Configure Validation Scenarios** dialog box.

New

Opens the **New scenario** dialog box allowing you to *create a new validation scenario*.

Duplicate

Allows you to duplicate the configuration of an existing validation scenario. It opens the **Edit scenario** dialog box where you can *configure the properties of the duplicated scenario*.

Edit

Opens the **Edit scenario** dialog box allowing you to *edit the properties of the currently selected validation scenario*.

Delete

Deletes the currently selected validation scenario.

Import scenarios

Imports validation scenarios.

Export selected scenarios

Export validation scenarios.

Move Up

Moves the selected scenario up one spot in the list.

Move Down

Moves the selected scenario down one spot in the list.

Extensions Tab

The **Extensions** tab specifies implementations of Java interfaces used to provide advanced functionality to the document type.

To open the **Extensions** tab of the **Document type** configuration dialog box, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Document Type Association**, use the ***New**, **Edit**, **Duplicate**, or **Extend** button*, and click on the **Extensions** tab.

Libraries containing the implementations must be present in the *classpath* of your document type. The Javadoc available at *https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/* contains details about how each API implementation functions.

Encoding Preferences

Oxygen XML Editor lets you configure how character encodings are recognized when opening files and which encodings are used when saving files. To configure encoding options, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Encoding**.

The following encoding options are available:

Fallback character encoding

Specifies the default character encoding of non-XML documents if their character encoding cannot be determined from other sources (for example, it is not specified in the document or determined by the file type).

Note: For certain document types, the following encoding detection rules are used:

- For XML, DTD, and CSS documents, Oxygen XML Editor tries to collect the character encoding from the document. If no such encoding is found, then *UTF-8* is used.
- For JavaScript, JSON, SQL, XQuery, and RNC, the *UTF-8* encoding is used.

UTF-8 BOM handling

This setting specifies how to handle the *Byte Order Mark* (BOM) when Oxygen XML Editor saves a UTF-8 XML document:

- **Keep** (default) - Do not alter the BOM declaration of the currently open file.
- **Write** - Save the BOM bytes.
- **Don't Write** - Do not save the BOM bytes. Loaded BOM bytes are ignored.

Note: The UTF-16 BOM is always preserved. UTF-32 documents have a *big-endian* byte order.

Encoding errors handling

This setting specifies how to handle characters that cannot be represented in the character encoding that is used when the document is opened. The available options are:

- **REPORT** (default) - Displays an error identifying the character that cannot be represented in the specified encoding. Unrecognized characters are rendered as an empty box.
- **REPLACE** - The character is replaced with a standard replacement character. For example, if the encoding is UTF-8, the replacement character has the Unicode code FFFD, and if the encoding is ASCII, the replacement character code is 63.
- **IGNORE** - The error is ignored and the character is not included in the document displayed in the editor.

Attention: If you edit and save the document, the characters that cannot be represented in the specified encoding are dropped.

Encoding for Base64, Base32, Hex conversions

Specifies the encoding to be used when invoking the **Encode Selection** or **Decode Selection** actions for *Base64*, *Base32*, or *Hex conversions*. The default setting is *UTF8*.

Editor Preferences

Oxygen XML Editor lets you configure the appearance of various components and features of the main editor. To access these options, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Editor**.

The following options are available:

Selection background color

Allows you to set the background color of selected text.

Selection foreground color

Allows you to set the color of selected text.

Completion proposal background

Allows you to set the background color of the *Content Completion Assistant*.

Completion proposal foreground

Allows you to set the color of the text in the *Content Completion Assistant*.

Documentation window background

Allows you to set the background color of the documentation of elements suggested by the *Content Completion Assistant*.

Documentation window foreground

Allows you to set the color of the text for the documentation of elements suggested by the *Content Completion Assistant*.

Find highlight color

Allows you to set the color of the highlights generated by the **Find** and **Find all** actions.

XPath highlight color

Allows you to set the color of the highlights generated when you run an XPath expression.

Declaration highlight color

Allows you to set the color of the highlights generated by the **Find declaration** action.

Reference highlight color

Allows you to set the color of the highlights generated by the **Find reference** action.

Maximum number of highlights

Allows you to set the maximum number of highlights that Oxygen XML Editor displays.

Show TAB/NBSP/EOL/EOF marks

Makes the **TAB**/**NBSP**/**EOL**/**EOF** characters visible in the editor. You can use the color picker to choose the color of the marks.

Show SPACE marks

Makes the space character visible in the editor.

Can edit read-only files

If this option is selected, Oxygen XML Editor will let you edit read-only files. When you try to save them, a **Save As** dialog box will be displayed to avoid overwriting the initial resource. If the option is not selected, a warning message is displayed when you try to edit a read-only file.

Display quick-assist and quick-fix side hints

Displays the *Quick Assist* icon ( ) and *Quick Fix* icon ( ) in the line number stripe on the left side of the editor.

Undo history size

Allows you to set the maximum amount of undo operations you can perform in any of the editor modes (**Text**, **Author**, **Design**, **Grid**).

Enable mouse-wheel zooming

If selected, you can use **Ctrl + MouseWheelForward (Command + MouseWheelForward on OS X)** to increase the editor font (zoom in) or **Ctrl + MouseWheelBackwards (Command + MouseWheelBackwards on OS X)** to decrease the editor font (zoom out). It is enabled by default on Windows and Linux, while it is disabled by default on Mac OS X, due to the way inertia affects the mouse wheel on this operating system.

Print Preferences

Oxygen XML Editor lets you configure how files are printed out of the editor. Note that these setting cover how files are printed directly from Oxygen XML Editor itself, not how they are printed after the XML source has been transformed by a publishing stylesheet. To configure the **Print** options, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Editor** > **Print**.

This page allows you to customize the headers and footers added to a printed page when you print from the ***Text** mode* or ***Author** mode* editors. These settings do not apply to the ***Grid*** and *schema **Design*** mode.

You can specify what is printed on the **Left**, **Middle**, and **Right** of the header and footer using plain text of any of the following variables:

- *${currentFileURL}* - Current file as URL, that is the absolute file path of the current edited document represented as URL.
- *${cfne}* - Current file name with extension. The current file is the one currently opened and selected.
- *${cp}* - Current page number. Used to display the current page number on each printed page in the **Editor / Print** Preferences page.
- *${tp}* - Total number of pages in the document. Used to display the total number of pages on each printed page in the **Editor / Print** Preferences page.
- *${env(VAR_NAME)}* - Value of the *VAR_NAME* environment variable. The environment variables are managed by the operating system. If you are looking for Java System Properties, use the *${system(var.name)}* editor variable.
- *${system(var.name)}* - Value of the *var.name* Java System Property. The Java system properties can be specified in the command line arguments of the Java runtime as *-Dvar.name=var.value*. If you are looking for operating system environment variables, use the *${env(VAR_NAME)}* editor variable instead.
- *${date(pattern)}* - Current date. The allowed patterns are equivalent to the ones in the *Java SimpleDateFormat class*. **Example:** *yyyy-MM-dd*;

  Note: This editor variable supports both the xs:date and xs:datetime parameters. For details about xs:date, go to *http://www.w3.org/TR/xmlschema-2/#date*. For details about xs:datetime, go to *http://www.w3.org/TR/xmlschema-2/#dateTime*.

For example, to show the current page number and the total number of pages in the top right corner of the page, write the following pattern in the **Right** text area of the **Header** section: `${cp} of ${tp}`.

You can also set the **Color** and **Font** used in the header and footer. Default font is `SansSerif`.

You can place a line below the header or above the footer by selecting **Underline/Overline**.

Edit Modes Preferences

Oxygen XML Editor lets you configure which *edit mode* a file is opened in the first time it is opened. This setting only applies the first time a file is opened. The current editing mode of each file is saved when the file is closed and restored the next time it is opened. To configure the options for editing modes, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Editor** > **Edit Modes** .

Allow Document Type specific edit mode setting to override the general mode setting

If selected, the initial edit mode setting set in the ***Document Type** configuration dialog box* overrides the general edit mode setting from the table below.

Select the initial edit mode (page) for each editor

This table specifies the default editing mode that will be opened for each type of document when the **Allow Document Type specific edit mode setting to override the general mode setting** option is not selected. Use the **Edit** button to change the initial edit mode for each type of document (editor). The initial edit mode can be one of the following:

- ***Text***
- ***Author***
- ***Grid***
- **Design** (available only for the XSD editor).

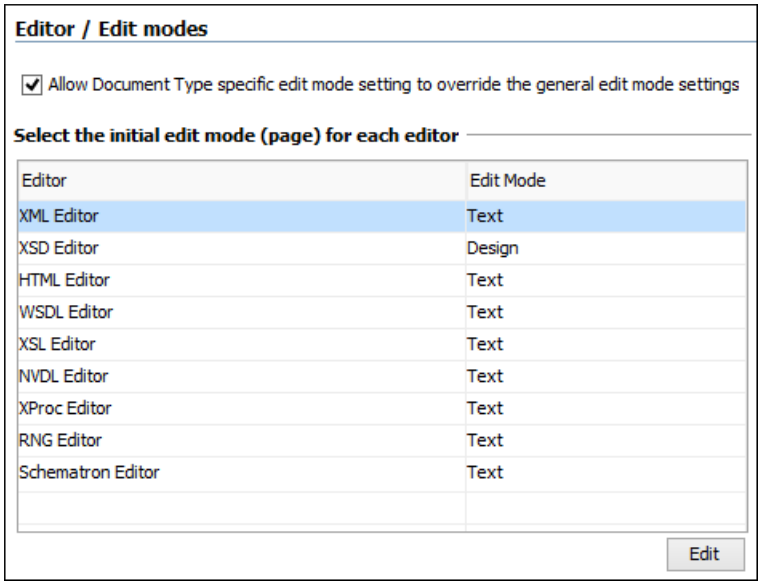


Figure 25: Edit Modes Preferences Page

Text Preferences

Oxygen XML Editor lets you configure how the ***Text*** *mode editor* appears. To configure these options, *open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Editor** > **Edit modes** > **Text**.

The following options are available:

Editor background color

Sets the background color for the **Text** editing mode, ***Outline*** *view*, and some external tool editors (***Large File Viewer***, ***Compare Files***, ***Compare Directories***).

Editor cursor color

Sets the color for the cursor in **Text** mode.

Highlight current line

If selected, the current line is highlighted with the foreground color specified with the color chooser.

Show line numbers

If selected (default value), line numbers are shown in the editor panels and in the ***Output*** *view* of the debugger *perspectives*. You can also specify the color for the line numbers using the color chooser. Printed output will also include the line numbers.

Show print margin

If selected, it allows you to set a safe print limit in the form of a vertical line displayed in the right side of the editor pane. You can also customize the print margin line color.

Print margin column

Allows you to specify a limit for the print width, measured in the number of characters.

Line wrap

If selected, long lines are automatically wrapped in edited documents. The line wrap does not alter the document content since the application does not use *new-line* characters to break long lines.

Cut / Copy whole line when nothing is selected

If selected, **Cut** and **Copy** actions operate on the entire current line when nothing is selected in the editor.

Enable folding

If selected (default value), the vertical stripe that holds the *folding markers* is displayed in **Text** mode.

Highlight matching tag

If selected, when you place the cursor on a start or end tag, Oxygen XML Editor highlights the corresponding member of the pair. You can also customize the highlight color.

Lock the XML tags

If selected, XML are locked and cannot be edited in **Text** mode.

Diagram Preferences

For certain XML languages, Oxygen XML Editor provides a diagram view as part of the text mode editor. To configure the **Diagram** preferences, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Edit modes** > **Text** > **Diagram**.

The following options are available in this preference page:

Show Full Model XML Schema diagram

When this option is selected, the **Text** mode editor for XML Schemas includes a split screen view that shows a diagram of the schema structure. This is useful for seeing the effects of schema changes you make. For editing a the schema using a diagram instead of text, use the *schema **Design** view*.

Note: When handling very large schemas, displaying the schema diagram might affect the performance of your system. In such cases, disabling the schema diagram view improves the speed of navigation through the edited schema.

Enable Relax NG diagram and related views

Enables the Relax NG schema diagram and synchronization with the related views (***Attributes***, ***Model***, ***Elements***, ***Outline***).

Show Relax NG diagram

Displays the Relax NG schema diagram in the split screen views (***Full Model View*** and ***Logical Model View***).

Enable NVDL diagram and related views

Enables the NVDL schema diagram and synchronization with the related views (***Attributes***, ***Model***, ***Elements***, ***Outline***).

Show NVDL diagram

Displays the NVDL schema diagram in the split screen views (***Full Model View*** and ***Logical Model View***).

Location relative to editor

Allows you to specify the location of the schema diagram panel relative to the diagram **Text** editor.

Show/Hide Annotations link

Use this link to navigate to the ***Schema Design** preferences page* where you can choose to show or hide annotations in schema diagrams.

Zoom link

Use this link to navigate to the ***Schema Design** preferences page* where you can adjust the default zoom level of schema diagrams.

Grid Preferences

Oxygen XML Editor provides a ***Grid** view* of an XML document. To configure the **Grid** options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Edit modes** > **Grid**.

The following options are available:

Compact representation

If selected, the *compact representation* of the grid is used: a child element is displayed beside the parent element. In the *non-compact representation*, a child element is nested below the parent.

Format and indent when passing from grid to text or on save

If selected, the content of the document is *formatted and indented* each time you switch from the **Grid** view to the **Text** view.

Default column width (characters)

Sets the default width (in characters) of a table column of the grid. A column may contain the following:

- Element names
- Element text content
- Attribute names
- Attribute values

If the total width of the grid structure is too large you can resize any column by dragging the column margins with the mouse pointer, but the change is not persistent. To make it persistent, set the new column width with this option.

Active cell color

Allows you to set the background color for the *active cell* of the grid. The keyboard input always goes to the *active cell* and the selection always contains it.

Selection color

Allows you to set the background color for the selected cells of the grid except the *active cell*.

Border color

Allows you to set the color used for the lines that separate the grid cells.

Background color

Allows you to set the background color of grid cells that are not selected.

Foreground color

Allows you to set the text color of the information displayed in the grid cells.

Row header colors

Background color

Allows you to set the background color of row headers that are not selected.

Active cell color

Allows you to set the background color of the row header cell that is currently active.

Selection color

Allows you to set the background color of the header cells corresponding to the currently selected rows.

Column header colors

The column headers are painted with two color gradients, one for the upper 1/3 part of the header and the other for the lower 2/3 part. The start and end colors of the first gradient are set with the first two color buttons. The start and end colors of the second gradient are set with the last two color buttons.

Background color

Allows you to set the background color of column headers that are not selected.

Active cell color

Allows you to set the background color of the column header cell that is currently active.

Selection color

Allows you to set the background color of the header cells corresponding to the currently selected columns.

Author Preferences

Oxygen XML Editor provides an ***Author*** *editing mode* that provides a configurable graphical interface for editing XML documents. To configure the options for the **Author** mode, *open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Editor** > **Edit modes** > **Author**.

The following options are available:

Author default background color

Sets the default background color of the **Author** editing mode. The `background-color` property set in the CSS file associated with the currently edited document overwrites this option.

Author default foreground color

Sets the default foreground color of the **Author** editing mode. The `color` property set in the CSS file associated with the current edited document overwrites this option.

Show XML comments

When this option is selected, XML comments are displayed in **Author** mode. Otherwise, they are hidden.

Show processing instructions

When this option is selected, XML processing instructions are displayed in **Author** mode. Otherwise they are hidden.

Show doctype

When this option is selected, the **doctype** declaration is displayed in **Author** mode. Otherwise it is hidden.

Show placeholders for empty elements

When this option is selected, placeholders are displayed for elements with no content to make them clearly visible. The placeholder is rendered as a light gray box and displays the element name.

Show Author layout messages

When this option is selected, all errors reported while rendering the document in **Author** mode are presented in the **Errors** view at the bottom of the editor.

Show block range

When this option is selected, a *block range indicator* is displayed in a stripe located in the left side of the editor. It is displayed as a heavy line that spans from the first line to the last line of the block.

Display referenced content (entities, XInclude, DITA conref, etc.)

When selected, the references (such as entities, XInclude, DITA conrefs) also display the content of the resources they reference. If you toggle this option while editing, you need to reload the file for the modification to take effect.

Images Section

The following options in regards to images in **Author** mode are available in this section:

Auto-scale images wider than (pixels)

Sets the maximum width that an image will be displayed. Wider images will be scaled to fit.

Show very large images

When this option is selected, images larger than 6 megapixels are displayed in **Author** mode. Otherwise, they are not displayed.

Important: If you select this option and your document contains many such images, Oxygen XML Editor may consume all available memory, throwing an **OutOfMemory error**. To resolve this, increase the *available memory limit.* and restart the application.

Tags Section

In this section you can configure the following options in regards to tags that are displayed in **Author** mode:

Tags display mode

Sets the default display mode for element tags presented in **Author** mode. You can choose between the following:

- **Full Tags with Attributes** - All XML tags are displayed, with attribute names and values, making it easier to transition from a Text-based editing to **Author** mode editing.
- **Full Tags** - All XML tags are displayed, but without attributes.
- **Block Tags** - The XML tags that enclose *block elements* are displayed in full. Compact tags (no element names) are displayed for *inline elements*.
- **Inline Tags** - The XML tags that enclose *inline elements* are displayed in full. *Block* tags are not displayed.
- **Partial Tags** - Partial tags (no names) are displayed for all elements.

- **No Tags** - No tags are displayed. This representation is as close as possible to a word-processor view.

Tags background color

Sets the **Author** mode tags background color.

Tags foreground color

Sets the **Author** mode tags foreground color.

Tags font

Allows you to change the font used to display tags text in the **Author** visual editing mode. The *default* font is computed based on the setting of the ***Author default font*** *option*.

Compact tag layout

If this option is not selected, the **Author** mode displays the tags in a more decompressed layout, where block tags are displayed on separate lines.

Serialization Section

In this section you can configure options in regards to the formatting and indenting that is applied when a document is saved in **Author** mode, or when switching the editing mode from **Author** to **Text**. The following options are available:

Format and indent

Use this option to specify what should be formatted and indented when you save a document (or switch from **Author** to **Text** mode). You can choose between the following two options:

Only the modified content

The **Save** operation only formats the nodes that were modified in the **Author** mode. The rest of the document preserves its original formatting.

Note: This option also applies to the *DITA maps* opened in the ***DITA Maps Manager***.

The entire document

The **Save** operation applies the formatting to the entire document regardless of the nodes that were modified in **Author** mode.

Also apply 'Format and Indent' action as in 'Text' edit mode

If this option is selected, the content of the document is formatted by applying the **Format and Indent** rules from the *Editor/Format/XML* option page. In this case, the result of the **Format and indent** operation will be the same as when it is applied in **Text** editing mode.

Compatibility with other tools

Use this option to control how line breaks are handled when a document is serialized. This will help to obtain better compatibility with other tools. You can choose one of the following:

- **None** - Choose this option if compatibility with other tools can be ignored.
- **Do not break lines, do not indent** - Choose this option to avoid breaking lines after element start or end tags and indenting will not be used.

  Note: New lines that are added by the user in elements where the `xml:space` attribute is set to `preserve` (such as `pre` elements in HTML, or `codeblock` elements in DITA) are still inserted. Also, selecting this option automatically disables the ***Also apply 'Format and Indent' action as in 'Text' edit mode*** *option*, since the formatting from **Text** mode does not take the CSS styles into account.
- **Break lines only after elements displayed as blocks, do not indent** - Choose this option to instruct Oxygen XML Editor to insert new lines only after elements that have a CSS display property set to anything other than `inline` or none (for example, `block`, `list-item`, `table`, etc.) and indenting will not be used. When selecting this option, the formatting is dictated by the CSS.

  Note: New lines that are added by the user in elements where the `xml:space` attribute is set to `preserve` (such as `pre` elements in HTML, or `codeblock` elements in DITA) are still inserted. Also, selecting this option automatically disables the ***Also apply 'Format and Indent' action as in 'Text' edit mode*** *option*, since the formatting from **Text** mode does not take the CSS styles into account.

For advanced Author configuration see the Document Type Association settings

Click this link to open the ***Document Type Association*** *preferences page*.

Cursor Navigation Preferences

Oxygen XML Editor allows you to configure the appearance and behavior of the cursor in the ***Author*** *mode editor*. To set cursor navigation preferences, *open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Author** > **Cursor Navigation**.

The following options are available:

Highlight elements near cursor

When this option is selected, the element containing the cursor is highlighted. You can use the color picker to choose the color of the highlight.

Show cursor position tooltip

Oxygen XML Editor uses *tool tips in* ***Author*** *mode to indicate the position of the cursor in the element structure* of the underlying document. Depending on context, the tool tips may show the current element name or the names of the elements before and after the current cursor position.

Show location tooltip on mouse move

When this option is selected, Oxygen XML Editor displays *Location Tooltips* when you are editing the document in certain tags display modes (**Inline Tags**, **Partial Tags**, **No Tags**) or when the mouse pointer is moved between *block elements*.

Quick up/down navigation

This option is deselected by default and this means that when you navigate using the up and down arrow keys in **Author** mode, the cursor is placed within each of the underlying XML elements between two blocks of text (the cursor changes to a horizontal line when it is between blocks of text). This allows you to easily insert elements and manage the structure of your XML content. However, if this option is selected, the cursor ignores the XML structure and jumps from one line of text to another, similar to how the cursor behaves in a word processor.

Quick navigation in tables

This option is selected by default and this means that when navigating between table cells with the arrow keys, the cursor jumps from one cell to another. If this option is not selected, the cursor navigates between XML nodes when navigating between table cells with the arrow keys.

Arrow keys move the cursor in the writing direction

This setting determines how the left and right arrow keys behave in **Author** mode for bidirectional (BIDI) text. When this option is selected (default value), the right arrow key advances the cursor in the reading direction and the left arrow moves it in the opposite direction. When this option is not selected, pressing the right arrow will simply move the cursor to the right (and the left arrow moves it to the left), regardless of the text direction.

Schema Aware Preferences

Oxygen XML Editor can use the schema of your XML language to improve the way the ***Author*** mode editor handles your content. To configure the **Schema Aware** options, *open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Editor** > **Edit modes** > **Author** > **Schema Aware**.

The following options are available:

Schema aware normalization, format, and indent

When you open or save a document in **Author** mode, white space is normalized using the `display` property of the current CSS stylesheet and the values of the *settings* for **Preserve space elements**, **Default space elements**, and **Mixed content elements**. When this option is selected, the schema will also be used to normalize white space, based on the content model (*element-only*, *simple-content*, or *mixed*) . Note that the schema information takes precedence.

Indent blocks-only content

To avoid accidentally introducing inappropriate white space around *inline elements*, Oxygen XML Editor does not normally apply indenting to the source of an element with mixed content. If this option is

selected, Oxygen XML Editor will apply indenting to the source of mixed content elements that only contain *block elements*.

Schema Aware Editing

The options in this section determine how Oxygen XML Editor will use the schema of a document to control the behavior of the **Author** mode.

- **On** - Enables all schema-aware editing options.
- **Off** - Disables all schema-aware editing options.
- **Custom** - Allows you to select custom schema-aware editing options from the following:

Schema Aware Actions section

Delete element tags with backspace and delete

Controls what happens when you attempt to delete an element tag. The two options are:

- **Smart delete** - If deleting the tag would make the document invalid, Oxygen XML Editor will attempt to make the document valid by unwrapping the current element or by appending it to an adjacent element where the result would be valid. For instance, if you delete a bold tag, the content can be unwrapped and become part of the surrounding paragraph, but if you delete a list item tag, the list item content cannot become part of the list container. However, the content could be appended to a preceding list items.
- **Reject action when its result is invalid** - A deletion that would leave the document in an invalid state is rejected.

Paste and Drag and Drop

Controls the behavior for paste and drag and drop actions. Available options are:

- **Smart paste and drag and drop** - If the content inserted by a paste or drop action is not valid at the cursor position, according to the schema, Oxygen XML Editor tries to find an appropriate insert position. The possibilities include:
  - Creating a sibling element that can accept the content (for example, if you tried to paste a paragraph into an existing paragraph).
  - Inserting the content into a parent or child element (for example, if you tried to paste a list item into an existing list item, or into the space above or below and existing list).
  - Inserting the content into an ancestor element where it would be valid.
- **Reject action when its result is invalid** - If selected, Oxygen XML Editor will not let you paste content into a position where it would be invalid.

Typing

Controls the behavior that takes place when typing. Available options are:

- **Smart typing** - If typed characters are not allowed in the element at the cursor position, but the previous element does allow text, then a similar element will be inserted, along with your content.
- **Reject action when its result is invalid** - If selected, and the result of the typing action is invalid, the action will not be performed.

Content Completion

Controls the behavior that takes place when inserting elements using content completion. Available options are:

- **Press ENTER to show available content completion proposals** - If selected, pressing **Enter** will open the *Content Completion Assistant in **Author** mode*. If deselected, there are three possibilities:
  - The current element will be split (if possible).
  - A new element with the same name will be inserted (if possible).
  - Otherwise, a new paragraph will be inserted.
- **Show all possible elements in the content completion list** - If selected, the content completion list will show all the elements in the schema, even those that cannot be entered validly at the current position. If you select an element that is not valid at the current position, Oxygen XML Editor will attempt to find a valid location to insert it and may present you with several options.

- **Allow only insertion of valid elements and attributes** - If selected, the content completion list shows only the elements that can be inserted at the current position and will not allow you to enter any other element.
- **Allow only insertion of valid attribute values** - If selected, you cannot enter an attribute value that is not valid (according to the schema) in the ***Attributes*** *view* or ***In-place Attributes Editor***. If the attribute has a choice of values, you can select a possible value from a drop-down list in the combo box, but you cannot enter a value manually.

Warn on invalid content when performing action

A warning message will be displayed when performing an action that will result in invalid content. Available options are:

- **Delete Element Tags** - If selected, a warning message will be displayed if the *Delete Element Tags* action will result in an invalid document. You will be asked to confirm the deletion.
- **Join Elements** - If selected, a warning message will be displayed if the *Join Elements* action will result in an invalid document. You will be asked to confirm the join.

Automatically apply the best schema-aware insertion operation

If selected, Oxygen XML Editor automatically uses what it considers to be the best insertion solution, when there is an attempt to insert content that is not valid in a specific context. If not selected, Oxygen XML Editor will ask the user to choose from a list of proposed solutions.

Convert external content on paste

If selected, the *Smart Paste feature* is enabled when external content is pasted in **Author** mode.

Convert even when pasting inside space-preserve elements

If selected, the *Smart Paste feature* will be used even when external content is pasted inside a *space-preserve* element (such as a `codeblock`).

Convert pasted URLs to links

If selected, when a URL is pasted into **Author** mode, a link will be inserted (the type of link depends on the type of document). For example, in DITA documents, an `xref` is inserted.

Related Information:

Smart Paste Mechanism on page 343

Customizing Smart Paste Support on page 1211

Review Preferences

Oxygen XML Editor allows you to *add review comments and track changes* in your documents. The **Review** preferences page allows you to control how the Oxygen XML Editor review features work. To configure these options, *open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Editor** > **Edit modes** > **Author** > **Review**.

The available options are as follows:

Author

Specifies the name to be attached to all comments and to changes made while **Track Changes** is active. By default, Oxygen XML Editor uses the system user name.

Track Changes section (applies for all authors)

Initial State

Specifies whether or not the *Track Changes feature* is enabled when you open a document. You may have the *Track Changes* feature enabled in some documents and disabled in others, or you can choose to always enable or disable the feature for all documents. You can choose between the following options:

- **Stored in document** - The current state of the *Track Changes* feature is stored in the document itself, meaning that it is on or off depending on the state the last time the document was saved. This is the recommended setting when multiple authors work on the same set of documents as it will make it obvious to other authors that changes have been made in the document.
- **Always On** - The *Track Changes* feature is always on when you open a document. You can turn it off for an opened document, but it will be turned on for the next document you open.

- **Always Off** - The *Track Changes* feature is always off when you open a document. You can turn it on for an opened document, but it will be turned off for the next document you open.

Display changed lines marker

A changed line maker is a vertical line on the left side of the editor window indicating where changes have been made in the document. To hide the changed lines marker, deselect this option.

Inserted content color

When the *Track Changes feature* is on, the newly inserted content is highlighted with an *insertion marker* that uses a color to adjust the following display properties of the inserted content: *foreground*, *background*, and *underline*. This section allows you to customize the following color options:

- **Automatic** - If this option is selected, Oxygen XML Editor automatically assigns a color to each user who inserted content in the current document. The colors are picked from the ***Colors for automatic assignment** list*, the priority being established by the type of change (deletion, insertion, or comment) and in the order that you see in the list.
- **Fixed** - If this option is selected, Oxygen XML Editor uses the specified color for all insertion markers, regardless of who the author is.
- **Use same color for text foreground** - If selected, Oxygen XML Editor uses the color defined above (**Automatic** or **Fixed**) to render the foreground of the inserted content.
- **Use same color for background** - If selected, Oxygen XML Editor uses the color defined above (**Automatic** or **Fixed**) to render the background of the inserted content. A slider control allows you to set the transparency level of the background.

Deleted content color

When the *Track Changes feature* is on, the deleted content is highlighted with a *deletion marker* that uses a color to adjust the following display properties of the deleted content: *foreground*, *background*, and *strikethrough*. This section allows you to customize the following color options:

- **Automatic** - If this option is selected, Oxygen XML Editor automatically assigns a color to each user who deleted content in the current document. The colors are picked from the ***Colors for automatic assignment** list*, the priority being established by the type of change (deletion, insertion, or comment) and in the order that you see in the list.
- **Fixed** - If this option is selected, Oxygen XML Editor uses the specified color for all deletion markers, regardless of who the author is.
- **Use same color for text foreground** - If selected, Oxygen XML Editor uses the color defined above (**Automatic** or **Fixed**) to render the foreground of the deleted content.
- **Use same color for background** - If selected, Oxygen XML Editor uses the color defined above (**Automatic** or **Fixed**) to render the background of the deleted content. A slider control allows you to set the transparency level of the background.

Comments color section (applies for all authors)

Sets the background color of the text that is commented on. The options are:

- **Automatic** - If this option is selected, Oxygen XML Editor automatically assigns a color to each user who adds a comment in the current document. The colors are picked from the ***Colors for automatic assignment** list*, the priority being established by the type of change (deletion, insertion, or comment) and in the order that you see in the list.
- **Fixed** - If this option is selected, Oxygen XML Editor uses the specified color for all changes, regardless of who the author is. A slider control allows you to set the transparency level of the background.

Colors for automatic assignment list

These are the colors that will be automatically assigned for tracked insertion changes, tracked deletion changes, and comments if the **Automatic** option is selected in any of the sections in this preferences page.

The colors are assigned in the order that you see in this list. You can use the **Add**, **Edit**, or **Remove** buttons to modify the list of colors.

Related Information:

Reviewing Documents on page 357

Callouts Preferences

Oxygen XML Editor can display *callouts* for *review items such as comments and tracked changes*. To customize options for review callouts, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Edit modes** > **Author** > **Review** > **Callouts**.

The available options are as follows:

Show Review Callouts section

Comments

If selected, callouts are displayed for comments, including comments that are added to *tracked changes*. This option is selected by default.

Track Changes deletions

If selected, callouts are displayed for *tracked change* deletions and the following additional option becomes available:

Show deleted content in callout

If selected, the deleted content is also displayed in the callout.

Track Changes insertions

If selected, callouts are displayed for *tracked change* insertions and the following additional option becomes available:

Show inserted content in callout

If selected, the inserted content is also displayed in the callout.

Rendering section

Show review time

When selected, timestamp information is displayed in callouts.

Show all connecting lines

When selected, lines are shown that connect the callout to the location of the change.

Initial width (px)

Specifies the initial width of the callouts each time the document is opened. The default is 250 pixels.

Text lines count limit

Specifies the maximum number of lines to be shown in the callouts. The default is 5 lines. Note that this does not limit the number of lines in the actual comment. It only limits the number of lines shown without opening or editing it. To see the full comment, right-click on the callout and select **Edit Comment** or **Show Comment**.

Profiling/Conditional Text Preferences

Oxygen XML Editor lets you configure how *profiling and conditional text* is displayed in **Author** mode. It has built-in support for the standard conditional text features of DITA and DocBook that you can customize for your own projects. You can also add conditional support for other XML vocabularies, including your custom vocabularies.

To configure **Profiling/Conditional Text** options, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Edit modes** > **Author** > **Profiling/Conditional Text**.

Note: Note the following when configuring these settings:

- This preferences page is used to define how profiled elements are treated in **Author** mode. It does not create profiling or conditional text attributes or values in the underlying XML vocabulary. It just changes how the editor displays them.
- This preferences page should be used for profiling / conditional text elements only. To change how other types of attributes are displayed in the text, use a CSS file.
- If you are using the DITA XML vocabulary and a DITA *subject scheme map* is defined in the *root map* of your document, it will be used in place of anything defined using this dialog box.

This preferences page contains the following options and sections:

Import from DITAVAL

This button allows you to import profiling attributes from *DITAVAL files*. You can merge these new profiling attributes with the existing ones, or replace them completely. If the imported attributes conflict with the existing ones, Oxygen XML Editor displays a dialog box that contains two tables. The first one previews the imported attributes and the second one previews the already defined attributes. You can choose to either keep the existing attributes or replace them with the imported ones.

Note: When importing profiling attributes from DITAVAL files, Oxygen XML Editor automatically creates condition sets based on these files.

Profiling Attributes section

Allows you to specify a set of allowable values for each profiling or conditional attribute. You can use the **New** button at the bottom of the table *to add profiling attributes*, the **Edit** button to edit existing ones, or the **Delete** button to delete entries from the table. Use the **Up** and **Down** buttons to change the priority of the entries. If you have multiple entries with identical names that match the same document type, Oxygen XML Editor uses the one that is positioned highest in the table.

Report invalid profiling attribute values

If selected, it means the following:

- In DITA, the automatic validation will display a warning when a value that is not defined is found in the document.
- In the DITA **Validate and Check for Completeness** dialog box, the ***Report attributes and values that conflict with profiling preferences*** *option* is not displayed. This means that the validation will behave the same as if that option was selected and it will always report such values.

Allow contributing extra profiling attribute values

This option is selected by default, which means that users are allowed to add values that are not defined in preferences to profiling attributes. If a user inserts such a value, when invoking the **Edit Profiling Attributes** action from the contextual menu in **Author** mode (or for DITA topics, the **Edit Properties** action in the ***DITA Maps Manager***), the ***Profiling Values Conflict*** *dialog box* will appear and it includes an **Add these values to the configuration** action that will automatically add the new value to the particular profiling attribute. If deselected, Oxygen XML Editor behaves as if the **Preserve the configuration** option has been chose in the ***Profiling Values Conflict*** *dialog box* and that dialog box will never appear.

Configure profiling colors and styles link

Use this link to open the *profiling* ***Colors and Styles*** *preference page*.

Profiling Condition Sets section

Allows you to specify a specific set of profiling attributes to be used to specify a particular build configuration for your content. You can use the **New** button at the bottom of the table *to add condition sets*, the **Edit** button to edit existing ones, or the **Delete** button to delete entries from the table. Use the **Up** and **Down** buttons to change the priority of the entries. If you have multiple entries with identical names that match the same document type, Oxygen XML Editor uses the one that is positioned highest in the table.

Show excluded content

Toggles whether or not the ***Show Excluded Content*** *option* in the **Profiling / Conditional Text** drop-down menu is enabled by default.

Colors and Styles Preferences

Oxygen XML Editor lets you set the colors and styles used to display *profiling / conditional text* in the ***Author*** *mode editor*. To set Colors and Styles preferences, *open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Editor** > **Edit modes** > **Author** > **Profiling/Conditional Text** > **Colors and Styles**.

The preference page includes the following options and sections:

Show profiling colors and styles

Toggles whether or not the ***Show Profiling Colors and Styles*** *option* in the **Profiling / Conditional Text** drop-down menu is enabled by default.

Import from DITAVAL

Allows you to import profiling styles from `.ditaval` files. You can merge these new profiling styles with the existing ones, or replace them completely. If the imported styles conflict with the existing ones, Oxygen XML Editor displays a dialog box containing two tables: the first one previews the imported styles and the second one previews the already defined styles. You can choose to either keep the existing styles or replace them with the imported ones.

Profiling Colors and Styles Table

You can use buttons below this table to set specific colors and styles for the listed profiling attribute values. The table includes two categories:

- **Defined attributes values** - Contains the styles for profiling attribute values defined in the ***Profiling / Conditional Text*** preferences page. Each profiling attribute value has an associated style. To ease the process of customizing styles, the **Defined attributes values** category contains by default the list of empty styles. All you have to do is to adjust the colors and decorations, thus skipping the process of manually defining the association rules (document type, attribute name and value). This is the reason why a style from this category can only be *reset*, not deleted.
- **Other** - This category contains styles for attribute values that are not marked as profiling values, in the ***Profiling / Conditional Text*** preferences page. In this category are listed:
  - All the styles that were defined in other projects (with other profiling attribute value sets).
  - All the styles set for the profiling attributes defined in a *subject scheme map*.

Automatic styling button

If you click this button, Oxygen XML Editor will apply automatic styling to the profiling attribute values that do not have a style defined.

New button

Opens the **Add Profiling Style** dialog box that allows you to associate a set of coloring and styling properties to a profiling value.

Note: You can define a default style for a specific attribute by setting the **Attribute value** field to `<ANY>`. This style is applied for attribute values that do not have a specific style associated with it.

Edit button

Open the **Edit Profiling Style** dialog box that allows you to edit the colors or style for an existing profiling value. You can also double-click the value to open this dialog box.

Clear style button

Resets the style for the selected value to its default setting (no color or decoration).

Delete button

Delete the selected style from the **Other** category.

Related Information:

Filtering Attribute Values with a DITAVAL File on page 1709

Styling the Rendering of Profiled Content Using a DITAVAL File on page 1711

Attributes Rendering Preferences

Oxygen XML Editor lets you *display the profiling attributes applied to your content* in the **Author** mode editor. To configure how the profiling attributes appear, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Editor** > **Edit modes** > **Author** > **Profiling/Conditional Text** > **Attributes Rendering**. When the ***Show Profiling Attributes** option* is selected, the **Author** mode displays conditional text markers at the end of conditional text blocks. Use the options in this page to customize the rendering of these text markers.

The following options are available:

Show profiling attributes

Toggles whether or not the ***Show Profiling Attributes** option* in the **Profiling / Conditional Text** drop-down menu is enabled by default.

Show profiling attribute name

If selected, the names of the profiling attributes are displayed with their values. If unchecked, only the values are displayed.

Background color

Sets the background color used to display the profiling attributes.

Attribute name foreground color

Sets the foreground color used to display the names of the profiling attributes.

Attribute values foreground color

Sets the foreground color used to display values of the profiling attributes.

Border color

Sets the color of the border of the block that displays the profiling attributes.

MathML Preferences

Oxygen XML Editor allows you to *edit MathML* equations and displays the results in a preview window. For a more specialized *MathML* editor, you can *install Design Science MathFlow*, which is a commercial product that requires a separate license.

To configure the *MathML* editor or to enter your *MathFlow* license information, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Edit modes** > **Author** > **MathML**.

You can configure the following options:

Equation minimum font size

The minimum size of the font used for rendering mathematical symbols when editing in the **Author** mode.

MathFlow installation directory

The installation folder for the *MathFlow* components product (*MathFlow* SDK).

MathFlow license file

The license file for the *MathFlow* components product (*MathFlow* SDK).

MathFlow preferred editor

A *MathML* formula can be edited in one of three editors of *MathFlow* components product (*MathFlow* SDK).

- **Structure Editor** (default selection) - Targets professional XML workflow users.
- **Style Editor** - Tailored to the needs of content authors.
- **Simple Editor** - Designed for applications where end-users can enter mathematical equations without prior training and only the meaning of the math matters.

Save special characters

Specifies how special characters are saved in the XML file.

- **As entity names** - Saves the characters in `&name;` format. It refers to a character by the name of the entity that has the desired character as its replacement text. For example, the Greek *Omega* character is saved as `Ω`.
- **As character entities** (default selection) - Saves the characters in a hexadecimal value, using the `&#xNNN` format. For example, the Greek *Omega* character is saved as `Ω`.
- **As character values** - Saves the characters as the actual symbol. For example, the Greek *Omega* character is saved as Ω.

More documentation is available on the *Design Science MathFlow* website.

AutoCorrect Preferences

Oxygen XML Editor includes an option to automatically correct misspelled words as you type in **Author** mode. To enable and configure this *AutoCorrect feature*, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Edit Modes** > **Author** > **AutoCorrect**.

The following options are available:

Enable AutoCorrect

When selected (default state), while editing in **Author** mode, if you type anything that is listed in the **Replace** column of the Replacements table displayed in this preferences page, Oxygen XML Editor will automatically replace it with the value listed in the **With** column.

Use additional suggestions from the spell checker

If selected, in addition to anything listed in the Replacements table displayed in this preferences page, Oxygen XML Editor will also use suggestions from the Spell Checker to automatically correct misspelled words. Suggestions from the Spell Checker will only be used if the misspelled word is not found in the Replacements table.

Note: The *AutoCorrect* feature shares the same options configured in the ***Language options*** and ***Ignore elements*** sections in the **Spell Check** preferences page.

Spell Check options link

Use this link to navigate to the *Spell Check Preferences page*.

Replacements Table section

The *AutoCorrect* feature uses the Replacements table to automatically replace anything that is listed in the **Replace** column with the value listed in the **With** column for each language.

Replacements for language drop-down menu

You can specify the language for the Replacements table, and for each language, you can configure the items listed in the table. The language selected in this page is not the language that will be used by the *AutoCorrect* feature. It is simply the language for the Replacements table.

Replacements Table

You can double-click on cells in either column to edit the listed items. Use the **Add** button to insert new items and the **Remove** button to delete rows from the table.

Note: Any changes, additions, or deletions you make to this table are saved to a path that is specified in the ***AutoCorrect Dictionaries*** *preferences page*.

Smart quotes section

You can also choose to automatically convert double and single quotes to a quotation characters of your choice by using the following options in the **Smart quotes** section:

- **Replace "Single quotes"** - Replaces single quotes with the quotation symbols you select with the **Start quote** and **End quote** buttons.
- **Replace "Double quotes"** - Replaces double quotes with the quotation symbols you select with the **Start quote** and **End quote** buttons.

Global Options

If this option is selected, the options are stored on your local computer, in a folder that is not accessible to other users.

Project Options

If this option is selected, the options are stored in the project file and can be shared with other users. Selecting ***Project Options*** will only save your selections in ***Enable AutoCorrect***, ***Use additional suggestions from the spell checker***, and the options in the ***Smart quotes*** *section*. Changes to the Replacements table are not saved in this page. To save changes to the Replacements table at project level you need to specify a custom location in *the* ***User-defined replacements*** *section of the* ***AutoCorrect Dictionaries*** *preferences page* and select **Project Options** from that preferences page instead.

Restore Defaults

Restores the options in this preferences page to their default values and **also deletes any changes you have made to the** ***Replacements table***.

AutoCorrect Dictionaries Preferences

To set the Dictionaries preferences for the *AutoCorrect feature*, *open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Editor** > **Edit Modes** > **Author** > **AutoCorrect** > **Dictionaries**. This page allows you to specify the location of the dictionaries that Oxygen XML Editor uses for the *AutoCorrect* feature and the location for saving user-defined replacements.

The following options are available in this preferences page:

Dictionaries default folder

Displays the default location where the dictionaries that Oxygen XML Editor uses for the *AutoCorrect* feature are stored.

Include dictionaries from

Selecting this option allows you to specify a location where you have stored *AutoCorrect* dictionaries that you want to include, along with the default ones.

Important: Consider the following notes in regards to this option:

- The *AutoCorrect* mechanism takes into account *AutoCorrect* dictionaries collected both from the default and custom locations and multiple dictionaries from the same language are merged (for example, `en_UK.dat` from the default location is merged with `en_US.dat` from a custom location).
- If you have a generic *AutoCorrect* dictionary file (one that just has a two letter language code for its file name, such as `en.dat`) saved in either the default or custom location, the other more specific dictionaries (for example, `en_UK.dat` and `en_US.dat`) will not be merged and the existing generic dictionary will simply be used instead.
- If the additional location contains a file with the same name as one from the default location, the file in the additional location takes precedence over the file from the default location.

How to add more dictionaries link

Use this link to open a topic in the Oxygen XML Editor User Guide that explains how to *add dictionaries for the AutoCorrect feature*.

Save user-defined replacements in the following location

Specifies the target where added, edited, or deleted replacements are saved. By default, the target is the application preferences folder, but you can also choose a custom location.

Tip: To save changes to *the Replacement table (in the **AutoCorrect** preferences page)* at *project level*, select a custom location for the **User-defined replacements** and select ***Project Options*** at the bottom of the page.

Related Information:

Add Dictionaries for the AutoCorrect Feature on page 734

Schema Design Preferences

Oxygen XML Editor provides a *graphical schema design editor* to make editing XML Schema easier. To configure the **Schema Design** options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Edit modes** > **Schema Design**.

The following options are available in the **Schema Design** preferences page:

Show annotation in the diagram

When selected, Oxygen XML Editor displays the content of `xs:documentation` elements in schema diagrams.

When trying to edit components from another schema

The schema diagram editor will combine schemas imported by the current schema file into a single schema diagram. You can choose what happens if you try to edit a component from an imported schema. The options are:

- **Always go to its definition** - Oxygen XML Editor opens the imported schema file so that you can edit it.
- **Never go to its definition** - The imported schema file is not opened and the component cannot be edited in place.
- **Always ask** - Oxygen XML Editor asks if you want to open the imported schema file.

Zoom

Allows you to set the default zoom level of the schema diagram.

Properties Preferences

Oxygen XML Editor lets you control which properties to display for XML Schema components in the *XML Schema **Design** view*. To configure the schema design properties displayed, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Edit modes** > **Schema Design** > **Properties**.

This preferences page contains the following:

Show additional properties in the diagram

If this option is selected, the properties selected in the property table are shown in the XML Schema **Design** mode. This option is selected by default.

Properties Table

Show

Use this column in the table to select the properties that you want to be displayed in the XML Schema **Design** mode.

Only if specified

Use this column to select if you want the property to be displayed only if it is defined in the schema.

Spell Check Preferences

Oxygen XML Editor provides support for spell checking in the ***Text*** and ***Author*** editing modes. To configure the **Spell Check** options, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Spell Check**.

The following options are available:

Automatic spell check

This option is not selected by default. When selected, Oxygen XML Editor automatically checks the spelling as you type and highlights misspelled words in the document.

- **Select editors** - You can select which editors (and therefore which file types) will be automatically spell checked. File types (such as CSS and DTD), in which automatic spell checking is not usually helpful, are excluded by default.

Spell check highlight color

Use this option to set the color used by the spell check engine to highlight spelling errors.

Language options section

This section includes the following language options:

Default language

The default language list allows you to choose the language used by the spell check engine when the language is not specified in the source file. You can *add additional dictionaries to the spell check engines*.

Use "lang" and "xml:lang" attributes

When this option is selected, the contents of an element with one of the `lang` or `xml:lang` attributes is checked in that language. Choose between the following two options for instances when these attributes are missing:

- **Use the default language** - If the `lang` and `xml:lang` attributes are missing, the selection in the ***Default language** list* is used.
- **Do not check** - If the `lang` and `xml:lang` attributes are missing, the element is not checked.

XML spell checking in section

You can choose to check the spelling inside the following XML items:

- **Comments**
- **Attribute values**
- **Text**
- **CDATA**

Options section

This section includes the following other options:

Check capitalization

When selected, the spell checker reports capitalization errors (for example, a word that starts with lowercase after *etc.* or *i.e.*).

Check punctuation

When selected, the spell checker checks punctuation. Misplaced white space and unusual sequences, such as a period following a comma, are highlighted as errors.

Ignore mixed case words

When selected, the spell checker does not check words containing mixed case characters (for example, *SpellChecker*).

Ignore acronyms

Available only for the **Hunspell Spell Checker**. When selected, acronyms are not reported as errors.

Ignore words with digits

When selected, the spell checker does not check words containing digits (for example, *b2b*).

Ignore duplicates

When selected, the spell checker does not signal two successive identical words as an error.

Ignore URL

When selected, the spell checker ignores words recognized as URLs or file names (for example, *www.oxygenxml.com* or *c:\boot.ini*).

Allow compounds words

When selected, all words formed by concatenating two legal words with a hyphen (hyphenated compounds) are accepted. If recognized by the language, two words concatenated without hyphen (closed compounds) are also accepted.

Allow file extensions

When selected, the spell checker accepts any word ending with recognized file extensions (for example, *myfile.txt* or *index.html*).

Ignore elements section

You can use the **Add** and **Remove** buttons to configure a list of element names or XPath expressions to be ignored by the spell checker. The following restricted set of XPath expressions are supported:

- '/' and '//' separators
- '*' wildcard

An example of an allowed XPath expression is: */a/*/b*.

AutoCorrect options link

Use this link to navigate to the ***AutoCorrect*** *preferences page*.

Spell Check Dictionaries Preferences

To set the Dictionaries preferences, *open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Editor** > **Spell Check** > **Dictionaries**. This page allows you to configure the dictionaries (`.dic` files) and term lists (`.tdi` files) that Oxygen XML Editor uses and to choose where to save new learned words.

The following options are valid when Oxygen XML Editor uses the Hunspell spell checking engine:

Dictionaries and term lists default folder

Displays the default location where the dictionaries and term lists that Oxygen XML Editor uses are stored.

Include dictionaries and term list from

Selecting this option allows you to specify a location where you have stored dictionaries and term lists that you want to include, along with the default ones.

Important: Consider the following notes in regards to this option:

- The spell checker takes into account dictionaries and term lists collected both from the default and custom locations and multiple dictionaries and term lists from the same language are merged (for example, `en_UK.dic` from the default location is merged with `en_US.dic` from a custom location).

- If you have a generic dictionary file (one that just has a two letter language code for its file name, such as `en.dic`) saved in either the default or custom location, the other more specific dictionaries (for example, `en_UK.dic` and `en_US.dic`) will not be merged and the existing generic dictionary will simply be used instead.
- If the additional location contains a file with the same name as one from the default location, the file in the additional location takes precedence over the file from the default location.

How to add more dictionaries and term lists link

Use this link to open a topic in the Oxygen XML Editor User Guide that explains how to *add more dictionaries and term lists*.

Save learned words in the following location

Specifies the target where the newly learned words are saved. By default, the target is the application preferences folder, but you can also choose a custom location.

Delete learned words

Opens the list of learned words, allowing you to select the items you want to remove, without deleting the dictionaries and term lists.

Related Information:

Adding Spell Check Dictionaries on page 728

Adding Spell Check Term Lists on page 729

Document Checking Preferences

To configure the **Document Checking** (validation) options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Document Checking**. This page contains preferences for configuring how a document is checked for both well-formedness and validation errors.

The following options are available:

Maximum number of validation highlights

If a validation generates more errors than the number specified in this option, only the errors up to this number are highlighted in the editor panel and on the stripe that is displayed at the right side of the editor panel. This option applies to both *automatic validation* and *manual validation*.

Validation error highlight color

The color used to highlight validation errors in the document.

Validation warning highlight color

The color used to highlight validation warnings in the document.

Validation info highlight color

The color used to highlight validation info messages in the document.

Validation success color

The color used to highlight the success indicator of the validation operation in the vertical ruler bar.

Always show validation status

If this option is selected, the current validation error or warning is always visible in the message line at the bottom of the editor panel. This is useful when the **Enable automatic validation** option is selected and the vertical scroll bar changes position due to an error message being displayed.

Enable automatic validation

This causes the validation to be automatically executed in the background as the document is modified in Oxygen XML Editor.

Delay after the last key event (s)

The period of keyboard inactivity before starting a new validation (in seconds).

At the bottom of the preferences page you can *choose whether or not the saved options will be shared with other users by selecting **Global** or **Project** storage options*.

Format Preferences

This preferences page contains various formatting options that influence editing and formatting in both the ***Text*** and ***Author*** editing modes. To control additional options specifically for the **Author** mode editor, see *Whitespace Handling in Author Mode* on page 238.

Note: These settings apply to the formatting of source documents. The formatting of output documents is determined by the *transformation scenarios that create them*.

To configure the **Format** options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Format**.

The following options are available:

Detect indent on open

If selected, Oxygen XML Editor detects how a document is indented when it is opened. Oxygen XML Editor uses a heuristic method of detection by computing a weighted average indent value from the initial document content. You can deselect this setting if the detected value does not work for your particular case and you want to use a fixed-size indent for all the edited documents. If this option is selected, Oxygen XML Editor detects the following:

- When TAB characters are used to indent content, the size of the TAB characters is used for the indent size.
- Otherwise, the detected size of SPACE characters is used for the indent size.

Tip: If you want to minimize the formatting differences created by the **Format and Indent** operation in a document edited in the **Text** edited mode, make sure that both the **Detect indent on open** and ***Detect line width on open*** options are selected.

Use zero-indent, if detected

By default, if no indent was detected in the document, the fixed-size indent is used. Select this option if all your document have no indentation and you want to keep them that way.

Indent with tabs

If selected, indents are created using TAB characters. If unchecked, lines are indented using space characters. Selecting this option automatically disables the ***Detect indent on open*** option.

Indent size

The meaning of this setting depends on the following:

- If the ***Detect indent on open*** *option* is selected and TAB characters are detected at the beginning of the line, the *indent size* is the width of a TAB character. Otherwise, the *indent size* value is ignored and Oxygen XML Editor uses the number of detected SPACE characters.
- If the ***Indent with tabs*** *option* is selected, the *indent size* is the width of a TAB character.
- If neither of these options are selected, the *indent size* is the number of SPACE characters used for indenting the text lines.

For additional information about changing the *indent size*, see *Setting an Indent Size to Zero* on page 315.

For information about when this setting is used, see *Where Indent Size and Line Width Settings are Used in Oxygen XML Editor* on page 97.

Indent on enter

If selected, when you press `Enter` to insert a line break in the **Text** editing mode, an indentation will be added to the new line.

Enable smart enter

If selected, when you press the `Enter` key between a start and an end XML tag in the **Text** editing mode, the cursor is placed in an indented position on the empty line formed between the start and end tag.

Format and indent the document on open

If selected, an XML document is formatted and indented before opening it in Oxygen XML Editor.

Note: Some specialized types of XML documents do not benefit from this feature, including Relax NG, XSD, XSL, and Ant. However, the feature is available for some non-XML types of documents, such as CSS and JSON.

Detect line width on open

If selected, Oxygen XML Editor automatically detects the line width when the document is opened.

Hard line wrap (Limit to "Line width - Format and Indent")

If selected, when typing content in the **Text** editing mode and the maximum line width is reached, a line break is automatically inserted.

Line width - Format and Indent

Defines the number of characters after which the **Format and Indent** (pretty-print) action performs hard line-wrapping. For example, if set to 100, after a **Format and Indent** action, the longest line will have a maximum of 100 characters. This setting is also used when saving XML content edited in the **Author** editing mode.

Note: To avoid having an indent that is longer than the line width setting and without having sufficient space available for the text content, the indent limit is actually set at half the value of the **Line width - Format and Indent** setting. The remaining space is reserved for text.

For information about when this setting is used, see *Where Indent Size and Line Width Settings are Used in Oxygen XML Editor*.

Clear undo buffer before Format and Indent

The **Format and Indent** operation can be *undone*, but if used intensively, a considerable amount of the memory allocated for Oxygen XML Editor will be used for storing the undo states. If this option is selected, Oxygen XML Editor empties the undo buffer before doing a **Format and Indent** operation. This means you will not be able to undo any changes you made before the format and indent operation. Select this option if you encounter out of memory problems (**OutOfMemoryError**) when performing the **Format and Indent** operation.

Where Indent Size and Line Width Settings are Used in Oxygen XML Editor

The values set in the **Indent Size** and **Line Width - Format and Indent** options are used in various places in the application, including the following:

- When the **Format and Indent** action is used in the **Text** editing mode.
- When you press ENTER to break a line in the **Text** editing mode.
- When the **Hard line wrap (Limit to "Line width - Format and Indent")** option is selected and the maximum line width is reached while editing in the **Text** mode.
- When the XML is serialized by saving content in the **Author** editing mode.

To watch our video demonstration about the formatting options offered by Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/Autodetect_Formating.html*.

XML Preferences

To configure the **XML** Formatting options, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Editor** > **Format** > **XML**.

The following options are available:

Format Section

This section includes the following drop-down boxes:

Preserve empty lines

The **Format and Indent** operation preserves all empty lines found in the document.

Preserve text as it is

The **Format and Indent** operation preserves text content as it is, without removing or adding any white space.

Preserve line breaks in attributes

Line breaks found in attribute values are preserved.

Note: When this option is selected, the ***Break long attributes** option* is automatically disabled.

Break long attributes

The **Format and Indent** operation breaks long attribute values.

Indent inline elements

The *inline elements* are indented on separate lines if they are preceded by white spaces and they follow another element start or end tag. For example:

Original XML:

```
<root>
  text <parent> <child></child> </parent>
</root>
```

Indent inline elements selected:

```
<root> text <parent>
    <child/>
  </parent>
</root>
```

Indent inline elements not selected:

```
<root> text <parent> <child/> </parent> </root>
```

Expand empty elements

The **Format and Indent** operation outputs empty elements with a separate closing tag (for example, `<a atr1="v1"></a>`). When not selected, the same operation represents an empty element in a more compact form (`<a atr1="v1"/>`).

Sort attributes

The **Format and Indent** operation sorts the attributes of an element lexicographically.

Add space before slash in empty elements

Inserts a space character before the trailing / and > of empty elements.

Break line before an attribute name

The **Format and Indent** operation breaks the line before the attribute name.

Element Spacing Section

This section controls how the application handles whitespaces found in XML content. You can **Add** or **Remove** element names or simplified XPath expressions in the various tabs.

The XPath expressions can accept multiple attribute conditions and inside each condition you can use *AND/OR* boolean operators and parentheses to override the priority.

You can use one or more of the following attribute conditions (default attribute values are not taken into account):

- *element[@attr]* - Matches all instances of the specified element that include the specified attribute.
- *element[not(@attr)]* - Matches all instances of the specified element that do not include the specified attribute.
- *element[@attr = "value"]* - Matches all instances of the specified element that include the specified attribute with the given value.
- *element[@attr != "value"]* - Matches all instances of the specified element that include the specified attribute and its value is different than the one given.

Example: The following is an example of how you could use multiple boolean operators and parentheses inside an attribute condition:

```
*[@a and @b or @c and @d]
*[@a and (@b or @c) and @d]
```

The following are just examples of how simplified XPath expressions might look like:

- `elementName`
- `//elementName`
- `/elementName1/elementName2/elementName3`
- `//xs:localName` Note: The namespace prefixes (such as `xs`) are treated as part of the element name without taking its binding to a namespace into account.

- `//xs:documentation[@lang="en"]`

The tabs are as follows:

Preserve space

List of elements for which the **Format and Indent** operation preserves the whitespaces (such as blanks, tabs, and newlines).

Default space

List of elements for which the content is normalized (multiple contiguous whitespaces are replaced by a single space), before applying the **Format and Indent** operation.

Mixed content

The elements from this list are treated as mixed content when applying the **Format and Indent** operation. The lines are split only when whitespaces are encountered.

Line break

List of elements for which line breaks will be inserted, regardless of their content. You can choose to break the line *before* the element, *after*, or both.

Schema aware format and indent

The **Format and Indent** operation takes the schema information into account with regards to the *space preserve*, *mixed*, or *element only* properties of an element.

Indent Section

Includes the following options:

Indent (when typing) in preserve space elements

Normally, the *Preserve space* elements (identified by the `xml:space` attribute set to `preserve` or by their presence in the ***Preserve space*** *tab of the* ***Element Spacing*** *list*) are ignored by the **Format and Indent** operation. When this option is selected and you edit one of these elements, its content is formatted.

Indent on paste - sections with number of lines less than 300

When you paste a chunk of text that has fewer than 300 lines, the inserted content is indented. To keep the original indent style of the document you copy content from, deselect this option.

Whitespaces Preferences

When Oxygen XML Editor formats and indents XML documents, a whitespace normalization process is applied, thus replacing whitespace sequences with single space characters. Oxygen XML Editor allows you to configure which Unicode characters are treated as spaces during the normalization process.

To configure the **Whitespace** preferences, *open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Editor** > **Format** > **XML** > **Whitespaces**.

This table lists the Unicode whitespace characters. Select any that you want to have treated as whitespace when formatting and indenting an XML document.

The whitespaces are normalized when:

- The **Format and Indent** action is applied on an XML document.
- You switch from **Text** mode to **Author** mode.
- You switch from **Author** mode to **Text** mode.

Note: The whitespace normalization process replaces any sequence of characters declared as whitespaces in the **Whitespaces** table with a single space character (`U+0020`). If you want to be sure that a certain whitespace character will not be removed in the normalization process, deselect it in the **Whitespaces** table.

Important: The characters with the codes `U+0009` (TAB), `U+000A` (LF), `U+000D` (CR) and `U+0020` (SPACE) are always considered to be whitespace characters and cannot be deselected.

XQuery Preferences

To configure the **XQuery** Formatting options, *open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Editor** > **Format** > **XQuery**.

The following options are available:

- **Preserve line breaks** - All initial line breaks are preserved.
- **Break line before an attribute name** - Each attribute of an XML element is written on a new line and properly indented.

XPath Preferences

To configure the **XPath** Formatting options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Format** > **XPath**.

The following option is available:

- **Format XPath code embedded in XSLT, XSD and Schematron files** - If selected, the **Format and Indent** action applied on an XSD, XSLT, or Schematron document will perform an XPath-specific formatting on the values of the attributes that accept XPath expressions.

  Note: For XSLT documents, the formatting is not applied to *attribute value templates*.

CSS Preferences

Oxygen XML Editor can format and indent your CSS files. To configure the **CSS** formatting options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Format** > **CSS**.

The following options control how your CSS files are formatted and indented:

Class body on new line

If selected, the *class* body (including the curly brackets) is placed on a new line. This option is not selected by default.

Indent class content

When selected (default state), the *class* content is indented.

Add space before the value of a CSS property

When selected (default state), whitespaces are added between the `:` (colon) and the value of a style property.

Add new line between classes

If selected, an empty line is added between two classes. This option is not selected by default.

Preserve empty lines

When selected (default state), the empty lines from the CSS content are preserved.

Allow formatting embedded CSS

When selected (default state), CSS content that is embedded in XML is also formatted when the XML content is formatted.

JavaScript Preferences

To configure the **JavaScript** format options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Format** > **JavaScript**.

The following options control the behavior of the **Format and Indent** action:

- **Start curly brace on new line** - Opening curly braces start on a new line.
- **Preserve empty lines** - Empty lines in the JavaScript code are preserved. This option is selected by default.
- **Allow formatting embedded JavaScript** - Applied only to XHTML documents, this option allows Oxygen XML Editor to format embedded JavaScript code, taking precedence over the *Schema aware format and indent* option. This option is selected by default.

Content Completion Preferences

Oxygen XML Editor provides a *Content Completion Assistant* that provides a list of available options at any point in a document and can auto-complete structures, elements, and attributes. To configure the **Content Completion** preferences, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Content Completion**. These options control how the *Content Completion Assistant* works.

The following options are available:

Auto close the last opened tag

When selected, Oxygen XML Editor automatically closes the last open tag when you type `</`.

Automatically rename/delete/comment matching tags

If you rename, delete, or comment out a start tag, Oxygen XML Editor automatically renames, deletes, or comments out the matching end tag.

Note: If you select **Toggle comment** for multiple starting tags and the matching end tags are on the same line as other start tags, the end tags are not commented.

Enable content completion

Toggles the content completion feature on or off.

Close the inserted element

When you choose an entry from the *Content Completion Assistant* list of proposals, Oxygen XML Editor inserts both start and end tags. The following additional options are available in regards to closing the element:

- **If it has no matching tag** - The end tag of the inserted element is automatically added only if it is not already present in the document.
- **Add element content** - Oxygen XML Editor inserts the required elements specified in the DTD, XML Schema, or RELAX NG schema that is *associated with the edited XML document*.
  - **Add optional content** - If selected, Oxygen XML Editor inserts the optional elements specified in the DTD, XML Schema, or RELAX NG schema.
  - **Add first Choice particle** - If selected, Oxygen XML Editor inserts the first **choice** particle specified in the DTD, XML Schema, or RELAX NG schema.

Case sensitive search

When selected, the search in the *Content Completion Assistant* is case-sensitive when you type a character ('a' and 'A' are different characters).

Note: This option is ignored when the current language itself is not case sensitive. For example, the case is ignored in the CSS language.

Position cursor between tags

When selected, Oxygen XML Editor automatically moves the cursor between the start and end tag after inserting the element. This only applies to:

- Elements with only optional attributes or no attributes at all.
- Elements with required attributes, but only when the ***Insert the required attributes*** *option* is not selected.

Show all entities

Oxygen XML Editor displays a list with all the internal and external entities declared in the current document when you type the start character of an entity reference (for example, &).

Insert the required attributes

Oxygen XML Editor inserts automatically the required attributes taken from the DTD or XML Schema.

Insert the fixed attributes

If selected, Oxygen XML Editor automatically inserts any `FIXED` attributes from the DTD or XML Schema for an element inserted with the help of the *Content Completion Assistant*.

Show recently used items

When selected, Oxygen XML Editor remembers the last inserted items from the *Content Completion Assistant* window. The number of items to be remembered is limited by the **Maximum number of recent items shown** list box. These most frequently used items are displayed on the top of the content completion window and are separated from the rest of the suggestions by a thin gray line .

Maximum number of recent items shown

Specifies the limit for the number of recently used items presented at the top of the *Content Completion Assistant* window.

Learn attributes values

When selected, Oxygen XML Editor learns the attribute values used in a document.

Learn on open document

Oxygen XML Editor automatically learns the document structure when the document is opened.

Learn words (Dynamic Abbreviations, available on CTRL-SPACE (COMMAND-SPACE on OS X))

When selected, Oxygen XML Editor learns the typed words and makes them available in a content completion fashion by pressing **Ctrl + Space (Command + Space on OS X)** on your keyboard;

Note: In order to be learned, the words need to be separated by space characters.

Activation delay of the proposals window (ms)

Delay in milliseconds from last key press until the *Content Completion Assistant* is displayed.

Related Information:

Configuring the Proposals for Attribute and Element Values on page 1241

Annotations Preferences

Certain types of schemas (XML Schema, DTDs, Relax NG) can include annotations that document the various elements and attributes that they define. Oxygen XML Editor can display these annotations when offering content completion suggestions. To configure the **Annotations** preferences, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Content Completion** > **Annotations**.

The following options are available:

Show annotations in Content Completion Assistant

Oxygen XML Editor displays the schema annotations of an element, attribute, or attribute value currently selected in the *Content Completion Assistant* proposals list.

Show annotations in tooltip

Oxygen XML Editor displays the annotation of elements and attributes as a tooltip when you hover over them with the cursor in the editing area or in the ***Elements** view*.

Show annotation in HTML format, if possible

This option allows you to view the annotations associated with an element or attribute in HTML format. It is available when editing XML documents that have associated an XML Schema or Relax NG schema. When this option is not selected, the annotations are converted and displayed as plain text.

Prefer DTD comments that start with "doc:" as annotations

To address the lack of dedicated annotation support in DTD documents, Oxygen XML Editor recommends prefixing with the `doc:` particle all comments intended to be shown to the developer who writes an XML validated against a DTD schema.

When this option is selected, Oxygen XML Editor uses the following mechanism to collect annotations:

- If at least one `doc:` comment is found in the entire DTD, only comments of this type are displayed as annotations.
- If no `doc:` comment is found in the entire DTD, all comments are considered annotations and displayed as such.

When the option is not selected, all comments, regardless of their type, are considered annotations and displayed as such.

Use all Relax NG annotations as documentation

When this option is selected, any element outside the Relax NG namespace, that is `http://relaxng.org/ns/structure/1.0`, is considered annotation and is displayed in the annotation window next to the *Content Completion Assistant* window and in the ***Model** view*. When this option is not selected, only elements from the Relax NG annotations namespace, that is `http://relaxng.org/ns/compatibility/annotations/1.0` are considered annotations.

Related Information:

Schema Annotations in Text Mode on page 302

XSLT Preferences

XSLT stylesheets are often used to create output in XHTML or XSL-FO. In addition to suggesting content completion options for XSLT stylesheet elements, Oxygen XML Editor can suggest elements from these vocabularies. To configure the XSLT content completion options, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Content Completion** > **XSLT**.

The following options are available:

Include elements declared in the schema section

This section includes options in regards to detecting elements from the declared schema.

Automatically detect HTML or Formatting Objects

Detects if the output being generated is HTML or FO and provides content completion for those vocabularies. Oxygen XML Editor analyzes the namespaces declared in the root element to find an appropriate schema.

If the detection fails, Oxygen XML Editor uses one of the following options:

- **None** - The *Content Completion Assistant* suggests only XSLT elements.
- **HTML** - The *Content Completion Assistant* includes HTML elements, including HTML5 elements (such as `video`, `canvas`, etc.).
- **Formatting objects** - The *Content Completion Assistant* includes Formatting Objects (XSL-FO) elements as substitutes for `xsl:element`.
- **Custom schema** - If you want content completion hints for another output vocabulary, you can use this option to specify the path to the schema for that vocabulary. The supported schema types are DTD, XML Schema, RNG schema, or NVDL schema for inserting elements from the target language of the stylesheet.

Documentation schema section

This section specifies an additional schema that will be used for documenting XSL stylesheets. You can choose between the following:

- **Build-in schema** - Uses the built-in schema for documentation.
- **Custom schema** - Allows you to specify a custom schema for documentation. The supported schema types are XSD, RNG, RNC, DTD, and NVDL.

XPath Preferences

Oxygen XML Editor provides content-completion support for XPath expressions. To configure the options for the content completion in XPath expressions, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Content Completion** > **XPath**.

The following options are available:

- **Enable content completion for XPath expressions** - Enables the *Content Completion Assistant in XPath expressions* that you enter in the `match`, `select`, and `test` XSL attributes and also in the *XPath toolbar*.
  - **Include XPath functions** - When this option is selected, XPath functions are included in the content completion suggestions.
  - **Include XSLT functions** - When this option is selected, XSLT functions are included in the content completion suggestions.
  - **Include axes** - When this option is selected, XSLT axes are included in the content completion suggestions.
- **Show signatures of XSLT / XPath functions** - Makes the editor indicate the signature of the XPath function located at the cursor position in a tooltip. See the *XPath Tooltip Helper* section for more information.
- **Function signature window background** - Specifies the background color of the tooltip window.
- **Function signature window foreground** - Specifies the foreground color of the tooltip window.

XSD Preferences

Oxygen XML Editor provides content completion assistance when you are writing XML Schema (XSD). To configure XSD preferences, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Content Completion** > **XSD**. The option in this preferences page allows you to define additional elements to be suggested by the *Content Completion Assistant* in `xs:appinfo` elements (in addition to the elements defined in the XML Schema).

The following option is available:

When in "xs:appinfo" context, also include elements declared in the schema

You can choose between the following:

- **None** - The *Content Completion Assistant* offers only the XML Schema schema information.
- **ISO Schematron** - The *Content Completion Assistant* also includes ISO Schematron elements in `xs:appinfo`.
- **Schematron 1.5** - The *Content Completion Assistant*also includes Schematron 1.5 elements in `xs:appinfo`.
- **Other** - The *Content Completion Assistant* also includes elements from an XML Schema identified by a URL in `xs:appinfo` elements.

JavaScript Preferences

Oxygen XML Editor can provide content completion suggestions when you are writing JavaScript files. To configure content completion support for JavaScript, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Content Completion** > **JavaScript**. You can configure the following options:

Enable content completion

Enables the content completion support for JavaScript files.

Use built-in libraries

Allows Oxygen XML Editor to include components (object names, properties, functions, and variables) collected from the built-in JavaScript library files when making suggestions.

Use defined libraries

Oxygen XML Editor can also use JavaScript libraries to when making suggestions. List the paths (URIs) of any JavaScript files you want Oxygen XML Editor to use when making suggestions.

Note: The paths can contain *editor variables* such as `${pdu}`, or `${oxygenHome}`. You can make these paths relative to the project directory or installation directory.

Syntax Highlight Preferences

Oxygen XML Editor supports syntax highlighting in the **Text** mode editors for numerous types of documents, including XML, XHTML, JavaScript, XQuery, XPath, PHP, CSS, LESS, Markdown, Text, DTD, RNC, Java, JSON, Ant, and more.

To configure syntax highlighting, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Syntax Highlight**.

To set syntax colors for a language, expand the listing for that language in the top panel to show the list of syntax items for that type of document. Use the color and style selectors to change how each syntax item is displayed. The results of your changes are displayed in the **Preview** panel. If you do not know the name of the syntax token that you want to configure, click that token in the **Preview** area to select it.

Note: All default color sets come with a high-contrast variant that is automatically used when you switch to a black-background or white-background high-contrast theme in your Windows operating system settings. The high-contrast theme will not overwrite any default color you set in **Editor** > **Syntax Highlight** preferences page.

The settings for XML documents are also used in XSD, XSL, RNG documents and the **Preview** area has a separate tab for each of them when **XML** is selected in the top pane.

The **Enable nested syntax highlight** option controls whether or not content types that are nested in the same file (such as PHP, JS, or CSS scripts inside an HTML file) are highlighted according to the color schemes defined for each content type.

Elements/Attributes by Prefix Preferences

Oxygen XML Editor allows you to specify syntax highlighting colors for XML elements and attributes with specific namespace prefixes. To configure the **Elements/Attributes by Prefix** preferences, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Syntax Highlight** > **Elements/Attributes by Prefix**.

To change the syntax coloring for a specific namespace prefix, choose the prefix from the list, or add a new one using the **New** button, and use the color and style selectors to set the syntax highlighting style for that namespace prefix.

Note: Syntax highlighting is based on the literal namespace prefix, not the namespace that the prefix is bound to in the document.

If you only want the prefix (and not the whole element or attribute name) to be styled with a particular color, select the **Draw only the prefix with a separate color** option.

Open/Save Preferences

Oxygen XML Editor lets you control how files are opened and saved. To configure the options for opening and saving documents, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Open/Save**.

The following options are available:

Open section

Lock local resources

When this option is selected and you open a file from the local file system or a shared network drive, Oxygen XML Editor locks the file for the current user and the file becomes read-only for other users while the lock exists. Locked and read-only files have a lock icon ( ) displayed on their editor tabs. Newly created files are *locked* when you first save them. If you select this option with files already opened in Oxygen XML Editor, it will *lock* all the currently opened files. If you deselect this option with files already opened, it will unlock them by deleting the corresponding `.lock` files. When you try to save locked (read-only) files, a **Save As** dialog box will be displayed to avoid overwriting the initial resource.

Restore cursor position

Selected by default, it ensures that the last position of the cursor will be remembered when a document is re-opened. If this option is not selected, the cursor will always be positioned at the beginning of the document.

Open each document in a tab next to the current one

When selected (default), each new document is opened in a tab next to the currently opened tab. If not selected, each new document is opened in a tab at the end of the current tab stack.

Support for Special Characters section

When bidirectional text, Asian languages, or other special characters are detected

You can choose how you want Oxygen XML Editor to handle bidirectional text, Asian languages, or other special characters when they are detected. You can choose one of the following:

- **Enable support for special characters** - The support for special characters will always be enabled.
- **Disable support for special characters** - The support for special characters will always be disabled.
- **Prompt for each document** - You will be prompted to choose whether or not to enable the support for special characters whenever they are detected in a newly opened document.

Disable special characters support for documents larger than (characters)

Enabling bidirectional text editing support can affect performance on large files. When this option is selected, bidirectional editing is disabled for files exceeding the specified size, even if the ***Enable support for special characters** option* is selected.

Save section

Show "Save as" option to save newly created documents in the "New" document wizard

It is selected by default, but if you deselect this option, the ***Save as** option* will not be available in the ***New Document** wizard*, so you will not have the ability to change the default name and path of the new file.

Safe save (only for local files)

In the unlikely event of a failure when attempting a **Save** action, this option provides increased protection from corruption of the original file. When this option is selected, it saves the content to a temporary file and if the save is unsuccessful, the editor preserves its unsaved state status.

On Save, make backup copy with extension (only for local files)

If selected, a backup copy is made when saving the edited document. This option is available only for local files (files that are stored on the local file system). The default backup file extension is `.bak`, but that can be changed in the text field.

Automatically save the document every

If selected, your documents are automatically saved after a preset time interval that is specified in the drop-down list.

Save all files before transformation or validation

Saves all opened files before validating or transforming an XML document. This ensures that any dependencies are resolved when modifying the XML document and its XML Schema.

Check errors on save

If selected, Oxygen XML Editor runs a validation that checks your document for errors before saving it.

Save all files before calling external tools

If selected, all files are saved before executing an *external tool*.

Performance section

Optimize loading in the Text edit mode for files over (MB)

File loading is optimized for reduced memory usage for any file whose size is larger than the value specified in this drop-down list. This is useful for editing large files, but there are *several restrictions* for memory-intensive operations.

Show warning when loading large documents

Oxygen XML Editor will warn you if you open a file that is bigger than the specified size.

Optimize loading for documents with lines longer than (Characters)

Line wrap is automatically performed for documents that contain lines that exceed the length specified in this text field. For a list of the restrictions applied to a document with long lines, see *Editing Documents with Long Lines*.

Show warning when loading documents with long lines

When selected, Oxygen XML Editorwill warn you when you open a file with lines longer than the specified length. To reduce the length of lines in a file, *format and indent the document* after it is opened in the editor panel. For a list of the restrictions applied to a document with long lines, see *Editing Documents with Long Lines* on page 736.

Clear undo buffer on save

If selected, Oxygen XML Editor clears its undo buffer when you save a document. Thus, modifications made prior to saving the document cannot be undone. Select this option if you frequently encounter **out of memory** errors when editing large documents.

Consider application bundles to be directories when browsing (OS X only)

This option is available only on the Mac OS X platform. When selected, the file browser dialog box allows you to browse inside an application bundle, as in a regular folder. Otherwise, it is not allowed (the same as the Finder application on Mac OS X).

Note: The same effect can be obtained by setting the `apple.awt.use-file-dialog-packages` property to **true** or **false** in the `Info.plist` descriptor file of the Oxygen XML Editor application:

```
<key>apple.awt.use-file-dialog-packages</key>
<string>false</string>
```

Save Hooks Preferences

Oxygen XML Editor includes an option for automatically compiling LESS stylesheets. To set this option, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Editor** > **Open/Save** > **Save Hooks**.

The following option is available:

Automatically compile LESS to CSS when saving

If selected, when you save a LESS file it will automatically be compiled to CSS (deselected by default).

Important: If this option is selected, when you save a LESS file, the CSS file that has the same name as the LESS file is overwritten without warning. Make sure all your changes are made in the LESS file. Do not edit the CSS file directly, as your changes might be lost.

Templates Preferences

This page simply allows you to navigate to the preference pages for code templates or document templates.

Code Templates Preferences

Code templates are code fragments that can be inserted at the current editing position. Oxygen XML Editor includes a set of built-in templates for CSS, LESS, Schematron, XSL, XQuery, and XML Schema document types. You can also define your own code templates for any type of file and share them with your colleagues using the template export and import functions.

To configure **Code Templates**, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Templates** > **Code Templates**.

This preferences page contains a list of all the available code templates (both built-in and custom created ones) and a code preview area. You can disable any code template by deselecting it.

The following actions are available:

New

Opens the **Code template** dialog box that allows you to define a new code template. You can define the following fields:

- **Name** - The name of the code template.
- **Description** - The description of the code template that will appear in the **Code Templates** preferences page and in the tooltip message when selecting it from the *Content Completion Assistant*. HTML markup can be used for better rendering.
- **Associate with** - You can choose to set the code template to be associated with a specific type of editor or for all editor types.
- **Shortcut key** - Allows you to configure a shortcut key that can be used to insert the code template. The **+** character separates keys. If the **Enable platform-independent shortcut keys** checkbox is selected, the shortcut is platform-independent and the following modifiers are used:
  - `M1` represents the **Command** key on MacOS X, and the **Ctrl** key on other platforms.
  - `M2` represents the **Shift** key.
  - `M3` represents the **Option** key on MacOS X, and the **Alt** key on other platforms.
  - `M4` represents the **Ctrl** key on MacOS X, and is undefined on other platforms.
- **Content** - Text box where you define the content that is used when the code template is inserted.

Edit

Opens the **Code template** dialog box and allows you to edit an existing code template. You can edit the following fields:

- **Description** - The description of the code template that will appear in the **Code Templates** preferences page and in the tooltip message when selecting it from the *Content Completion Assistant*. HTML markup can be used for better rendering.
- **Shortcut key** - Allows you to configure a shortcut key that can be used to insert the code template. The **+** character separates keys. If the **Enable platform-independent shortcut keys** checkbox is selected, the shortcut is platform-independent and the following modifiers are used:
  - `M1` represents the **Command** key on MacOS X, and the **Ctrl** key on other platforms.
  - `M2` represents the **Shift** key.
  - `M3` represents the **Option** key on MacOS X, and the **Alt** key on other platforms.
  - `M4` represents the **Ctrl** key on MacOS X, and is undefined on other platforms.
- **Content** - Text box where you define the content that is used when the code template is inserted.

Duplicate

Creates a duplicate of the currently selected code template.

Delete

Deletes the currently selected code template. This action is not available for the built-in code templates.

Export

Exports a file with code templates.

Import

Imports a file with code templates that was created by the **Export** action.

You can use the following *editor variables* when you define a code template in the **Content** text box:

- *${caret}* - The position where the cursor is located. This variable can be used in a code template, in **Author** mode operations, or in a **selection** *plugin*.
- *${selection}* - The current selected text content in the current edited document. This variable can be used in a code template, in **Author** mode operations, or in a **selection** *plugin*.
- *${ask('message', type, ('real_value1':'rendered_value1'; 'real_value2':'rendered_value2'; ...), 'default_value')}* - To prompt for values at runtime, use the *ask('message', type, ('real_value1':'rendered_value1'; 'real_value2':'rendered_value2'; ...), 'default-value")* editor variable. You can set the following parameters:
  - *'message'* - The displayed message. Note the quotes that enclose the message.
  - *type* - Optional parameter, with one of the following values:

    Note: The title of the dialog box will be determined by the type of parameter and as follows:

    - For *url* and *relative_url* parameters, the title will be the name of the parameter and the value of the *'message'*.
    - For the other parameters listed below, the title will be the name of that respective parameter.
    - If no parameter is used, the title will be "Input".

Parameter	
url	**Format:** *${ask('message', url, 'default_value')}*
	Description: Input is considered a URL. Oxygen XML Editor checks that the provided URL is valid.
	Example: • *${ask('Input URL', url)}* - The displayed dialog box has the name *Input URL*. The expected input type is URL. • *${ask('Input URL', url, 'http://www.example.com')}* - The displayed dialog box has the name *Input URL*. The expected input type is URL. The input field displays the default value *http://www.example.com*.
password	**Format**: *${ask('message', password, 'default')}*
	Description: The input is hidden with bullet characters.
	Example: • *${ask('Input password', password)}* - The displayed dialog box has the name *'Input password'* and the input is hidden with bullet symbols. • *${ask('Input password', password, 'abcd')}* - The displayed dialog box has the name *'Input password'* and the input hidden with bullet symbols. The input field already contains the default abcd value.
generic	**Format**: *${ask('message', generic, 'default')}*
	Description: The input is considered to be generic text that requires no special handling.

Parameter	
	Example: • *${ask('Hello world!')}* - The dialog box has a *Hello world!* message displayed. • *${ask('Hello world!', generic, 'Hello again!')}* - The dialog box has a *Hello world!* message displayed and the value displayed in the input box is *'Hello again!'*.
relative_url	**Format:** *${ask('message', relative_url, 'default')}*
	Description: Input is considered a URL. Oxygen XML Editor tries to make the URL relative to that of the document you are editing. **Note:** If the *$ask* editor variable is expanded in content that is not yet saved (such as an *untitled* file, whose path cannot be determined), then Oxygen XML Editor will transform it into an absolute URL.
	Example: • *${ask('File location', relative_url, 'C:/example.txt')}* - The dialog box has the name *'File location'*. The URL inserted in the input box is made relative to the current edited document location.
combobox	**Format:** *${ask('message', combobox, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default')}*
	Description: Displays a dialog box that offers a drop-down menu. The drop-down menu is populated with the given *rendered_value* values. Choosing such a value will return its associated value (*real_value*). **Note:** The *'default'* parameter specifies the default selected value and can match either a key or a value.
	Example: • *${ask('Operating System', combobox, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'osx')}* - The dialog box has the name *'Operating System'*. The drop-down menu displays the three given operating systems. The associated value will be returned based upon your selection. **Note:** In this example, the default value is indicated by the *osx* key. However, the same result could be obtained if the default value is indicated by *Mac OS X*, as in the following example: *${ask('Operating System', combobox, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'Mac OS X')}* • *${ask('Mobile OS', combobox, ('win':'Windows Mobile';'ios':'iOS';'and':'Android'), 'Android')}*
editable_combobox	**Format:** *${ask('message', editable_combobox, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default')}*
	Description: Displays a dialog box that offers a drop-down menu with editable elements. The drop-down menu is populated with the given *rendered_value* values. Choosing such a value will return its associated real value (*real_value*) or the value inserted when you edit a list entry. **Note:** The *'default'* parameter specifies the default selected value and can match either a key or a value.

Parameter	
	Example: • *${ask('Operating System', editable_combobox, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'osx')}* - The dialog box has the name *'Operating System'*. The drop-down menu displays the three given operating systems and also allows you to edit the entry. The associated value will be returned based upon your selection or the text you input.
radio	**Format:** *${ask('message', radio, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default')}*
	Description: Displays a dialog box that offers a series of radio buttons. Each radio button displays a *'rendered_value* and will return an associated *real_value*. **Note:** The *'default'* parameter specifies the default selected value and can match either a key or a value.
	Example: • *${ask('Operating System', radio, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'osx')}* - The dialog box has the name *'Operating System'*. The radio button group allows you to choose between the three operating systems. **Note:** In this example `Mac OS X` is the default selected value and if selected it would return *osx* for the output.

  - 'default-value' - optional parameter. Provides a default value.
- *${timeStamp}* - Time stamp, that is the current time in Unix format. For example, it can be used to save transformation results in multiple output files on each transformation.
- *${uuid}* - Universally unique identifier, a unique sequence of 32 hexadecimal digits generated by the Java *UUID* class.
- *${id}* - Application-level unique identifier. It is a short sequence of 10-12 letters and digits that is not guaranteed to be universally unique.
- *${cfn}* - Current file name without extension and without parent folder. The current file is the one currently opened and selected.
- *${cfne}* - Current file name with extension. The current file is the one currently opened and selected.
- *${cf}* - Current file as file path, that is the absolute file path of the current edited document.
- *${cfd}* - Current file folder as file path, that is the path of the current edited document up to the name of the parent folder.
- *${frameworksDir}* - The path (as file path) of the `[OXYGEN_INSTALL_DIR]/frameworks` directory.
- *${pd}* - The file path for the parent folder of the current project selected in the **Project** view.
- *${oxygenInstallDir}* - Oxygen XML Editor installation folder as file path.
- *${homeDir}* - The path (as file path) of the user home folder.
- *${pn}* - Current project name.
- *${env(VAR_NAME)}* - Value of the *VAR_NAME* environment variable. The environment variables are managed by the operating system. If you are looking for Java System Properties, use the *${system(var.name)}* editor variable.
- *${system(var.name)}* - Value of the *var.name* Java System Property. The Java system properties can be specified in the command line arguments of the Java runtime as *-Dvar.name=var.value*. If you are looking for operating system environment variables, use the *${env(VAR_NAME)}* editor variable instead.
- *${date(pattern)}* - Current date. The allowed patterns are equivalent to the ones in the *Java SimpleDateFormat class*. **Example:** *yyyy-MM-dd*;

Note: This editor variable supports both the xs:date and xs:datetime parameters. For details about xs:date, go to *http://www.w3.org/TR/xmlschema-2/#date*. For details about xs:datetime, go to *http://www.w3.org/TR/xmlschema-2/#dateTime*.

Related Information:

Code Templates on page 308

Document Templates Preferences

Oxygen XML Editor provides a selection of document templates that make it easier to create new documents in a variety of formats. The list of available templates is presented when you create a new document. You can also *create your own templates* and share them with others. You can store your custom file templates in the existing `templates` folder in the Oxygen XML Editor installation directory or store them in a custom directory. If you store them in a custom direction, you need to add that directory to the list of template directories that Oxygen XML Editor uses.

To add a template directory, follow these steps:

1. *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Templates** > **Document Templates**.
2. Use the **New** button to select a location of the new document template folder.

This will add the folder to the list in this preferences page and it will now appear in the ***New** document wizard*.

Note: For DITA templates, they will also appear in the dialog box for creating new DITA topics from the ***DITA Maps Manager***, but if you create a *corresponding properties file*, you need to set the `type` property to `dita`.

You can also use the **Edit** or **Delete** buttons to manage folders in the list, and you can alter the order in which Oxygen XML Editor looks in these directories by using the **Up** and **Down** buttons.

Mark Occurrences Preferences

This preferences page specifies which types of files will have the *Highlight IDs Occurrences* feature activated. To configure these options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Mark Occurrences**:

The following options are available in this preferences page:

Highlight component occurrences in the current file for:

- **XML files** - Activates the *Highlight IDs Occurrences* feature in XML files.
- **XSLT files** - Activates the *Highlight Component Occurrences* feature in XSLT files.
- **XML Schema files** - Activates the *Highlight Component Occurrences* feature in XSD files.
- **WSDL files** - Activates the *Highlight Component Occurrences* feature in WSDL files.
- **RNG files** - Activates the highlight component occurrences feature in RNG files.
- **Schematron files** - Activates the *Highlight Component Occurrences* feature in Schematron files.
- **Ant files** - Activates the *Highlight Component Occurrences* feature in Ant files.

Declaration highlight color

Allows you to choose the color to be used for highlighting component declarations.

Reference highlight color

Allows you to choose the color to be used for highlighting component references.

Custom Validation Engines Preferences

As the name implies, the **Custom Validation Engines** preferences page displays the list of custom validation engines than can be associated to a particular editor and used for validating documents. To access this page, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Custom Validation Engines**.

If you want to add a new custom validation tool or edit the properties of an exiting one, you can use the **Custom Validator** dialog box displayed by pressing the **New** or **Edit** button.

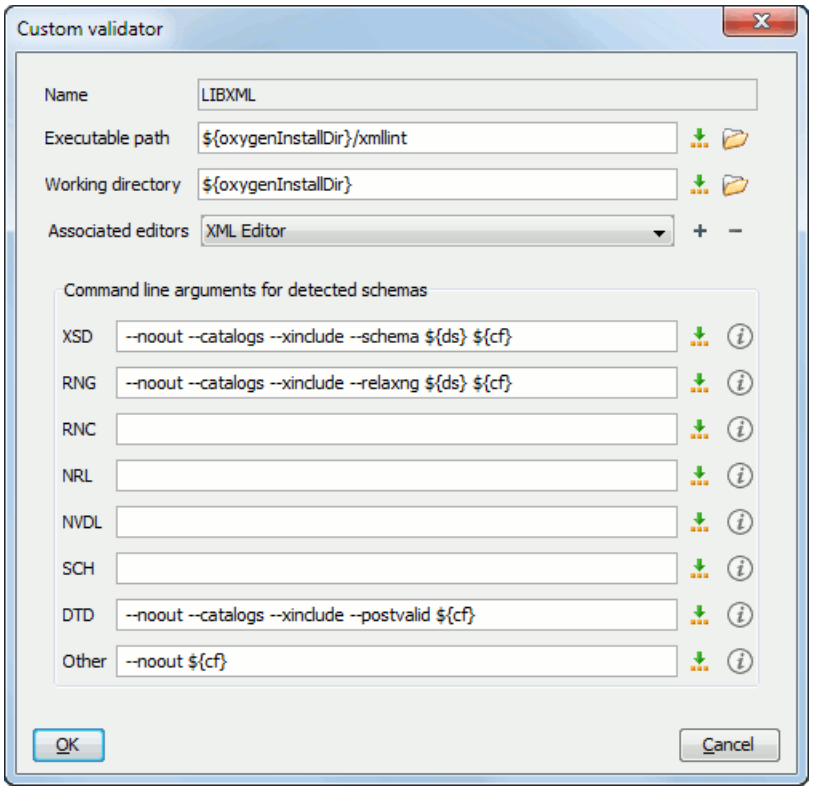


Figure 26: Custom Validator Dialog Box

The **Custom Validator** dialog box allows you to configure the following parameters:

Name

Name of the custom validation engine that will be displayed in the **Validation** toolbar drop-down menu.

Executable path

Path to the executable file of the custom validation tool. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Working directory

The working directory of the custom validation tool. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Associated editors

The editors that can perform validation with the external tool (XML editor, XSL editor, XSD editor, etc.)

Command line arguments for detected schemas

Command line arguments used in the commands that validate the currently edited file against various types of schema (W3C XML Schema, Relax NG full syntax, Relax NG compact syntax, NVDL, Schematron, DTD, etc.) The arguments can include any custom switch (such as -rng) and the following *editor variables*:

- *${cf}* - Current file as file path, that is the absolute file path of the current edited document.
- *${currentFileURL}* - Current file as URL, that is the absolute file path of the current edited document represented as URL.
- *${ds}* - The path of the detected schema as a local file path for the current validated XML document.
- *${dsu}* - The path of the detected schema as a URL for the current validated XML document.

Related Information:

Editor Variables on page 169

Increasing the Stack Size for Validation Engines

To prevent the appearance of a **StackOverflowException** error, use one of the following methods:

- Use the **com.oxygenxml.stack.size.validation.threads** property to increase the size of the stack for validation engines. The value of this property is specified in bytes. For example, to set a value of one megabyte specify 1x1024x1024=1048576.
- Increase the value of the **-Xss** parameter.

  Note: Increasing the value of the **-Xss** parameter affects all the threads of the application.

Related Information:

Setting a Java Virtual Machine Parameter when Launching Oxygen XML Editor on page 178

CSS Validator Preferences

To configure the **CSS Validator** preferences, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **CSS Validator**.

You can configure the following options for the built-in **CSS Validator** of Oxygen XML Editor:

- **Profile** - Selects one of the available validation profiles: **CSS 1**, **CSS 2**, **CSS 2.1**, **CSS 3**, **CSS 3 with Oxygen extensions**, **SVG**, **SVG Basic**, **SVG Tiny**, **Mobile**, **TV Profile**, **ATSC TV Profile**. The **CSS 3 with Oxygen extensions** profile includes all the CSS 3 standard properties plus the *CSS extensions specific for Oxygen* that can be used in *Author mode*. That means all Oxygen specific extensions are accepted in a CSS stylesheet by *the built-in CSS validator* when this profile is selected.
- **Media type** - Selects one of the available mediums: **all**, **aural**, **braille**, **embossed**, **handheld**, **print**, **projection**, **screen**, **tty**, **tv**, **presentation**, **oxygen**.
- **Warning level** - Sets the minimum severity level for reported validation warnings. Can be one of: **All**, **Normal**, **Most Important**, **No Warnings**.
- **Ignore properties** - You can type comma separated patterns that match the names of CSS properties that will be ignored at validation. As wildcards you can use:
  - `*` to match any string.
  - **?** to match any character.
- **Recognize browser CSS extensions (also applies to content completion)** - If selected, Oxygen XML Editor recognizes (no validation is performed) browser-specific CSS properties. The *Content Completion Assistant* lists these properties at the end of its list, prefixed with the following particles:
  - `-moz-` for Mozilla.
  - `-ms-` for Internet Explorer or Edge.
  - `-o-` for Opera.
  - `-webkit-` for Safari/Webkit.

XML Preferences

This section describes the panels that contain the user preferences related with XML.

XML Catalog Preferences

To configure options that pertain to *XML Catalogs*, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XML Catalog**.

The following options are available:

Prefer

Determines whether public identifiers specified in the catalog are used in favor of system identifiers supplied in the document. Suppose you have an entity in your document for which both a public identifier and a system identifier has been specified, and the catalog only contains a mapping for the public identifier (for example, a matching public catalog entry). You can choose between the following:

- **system** - If selected, the system identifier in the document is used.
- **public** - If selected, the URI supplied in the matching public catalog entry is used. Generally, the purpose of catalogs is to override the system identifiers in XML documents, so **public** should usually be used for your catalogs.

Note: If the catalog contains a matching system catalog entry giving a mapping for the system identifier, that mapping would have been used, the public identifier would never have been considered, and this setting would be irrelevant.

Verbosity

When using catalogs it is sometimes useful to see what catalog files are parsed, if they are valid or not, and what identifiers are resolved by the catalogs. This option selects the detail level of such logging messages of the *XML catalog* resolver that will be displayed in the **Catalogs** table at the bottom of the window. You can choose between the following:

- **None** - No message is displayed by the catalog resolver when it tries to resolve a URI reference, a SYSTEM one or a PUBLIC one with the *XML catalogs* specified in this panel.
- **Unresolved entities** - Only the logging messages that track the failed attempts to resolve references are displayed.
- **All messages** - The messages of both failed attempts and successful ones are displayed.

Resolve schema locations also through system mappings

If selected, Oxygen XML Editor analyzes both *uri* and *system* mappings to resolve the location of schema.

Note: This option is not applicable for DTD schemas since the public and system catalog mappings are always considered.

Process "schemaLocation" namespaces through URI mappings for XML Schema

If selected, the target namespace of the imported XML Schema is resolved through the *uri* mappings. The namespace is taken into account only when the schema specified in the *schemaLocation* attribute was not resolved successfully. If not selected, the system IDs are used to resolve the schema location.

Use default catalog

If this option is selected and Oxygen XML Editor cannot resolve the catalog mapping with any other means, the default global catalog (listed below this checkbox) is used. For more information, see *How Oxygen XML Editor Determines which Catalog to Use* on page 485.

Catalogs table

You can use this table to add or manage global user-defined catalogs. The following actions are available at the bottom of the table:

Add

Opens a dialog box that allows you to add a catalog to the list. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.

Edit

Opens a dialog box that allows you to edit an existing catalog. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.

Delete

Deletes the currently selected catalog from the list.

Up

Moves the selection to the previous resource.

Down

Moves the selection to the following resource.

Note: When you add, delete, or edit a catalog in this table, you need to reopen the currently edited files that use the modified catalog or run *a manual **Validate** action* so that the changes take full effect.

You can also add or configure catalogs at *framework* level from the ***Catalogs** tab* in the ***Document Type** configuration dialog box*.

Related Information:

Controlling the Catalog Resolver

Working with XML Catalogs on page 484

XML Parser Preferences

To configure the **XML Parser** options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XML Parser**.

The configurable options of the built-in XML parser are as follows:

Enable parser caching (validation and content completion)

Enables re-use of internal models when validating and provides content completion in opened XML files that reference the same schemas (grammars) such as DTD, XML Schema, or RelaxNG.

Ignore the DTD for validation if a schema is specified

Forces validation against a referenced schema (W3C XML Schema, Relax NG schema) even if the document includes also a DTD DOCTYPE declaration. This option is useful when the DTD declaration is used only to declare DTD entities and the schema reference is used for validation against a W3C XML Schema or a Relax NG schema.

Note: Schematron schemas are treated as additional schemas. The validation of a document associated with a DTD and referencing a Schematron schema is executed against both the DTD and the Schematron schema, regardless of the value of the **Ignore the DTD for validation if a schema is specified** option.

Enable XInclude processing

Enables XInclude processing. If selected, the XInclude support in Oxygen XML Editor is turned on for validation, rendering in **Author** mode and transformation of XML documents.

Base URI fix-up

According to the specification for XInclude, processors must add an `xml:base` attribute to elements included from locations with a different base URI. Without these attributes, the resulting infoset information would be incorrect.

Unfortunately, these attributes make XInclude processing to not be transparent to Schema validation. One solution to this is to modify your schema to allow `xml:base` attributes to appear on elements that might be included from different base URIs.

If the addition of `xml:base` and / or `xml:lang` is not desired by your application, you can deselect this option.

Language fix-up

The processor will preserve language information on a top-level included element by adding an `xml:lang` attribute if its include parent has a different [language] property. If the addition of `xml:lang` is not allowed by your application, you can deselect this option.

DTD post-validation

Select this option to validate an XML file against the associated DTD, after all the content merged to the current XML file using `XInclude` was resolved. If you deselect this option, the current XML file is validated against the associated DTD before all the content merged to the current XML file using `XInclude` is resolved.

XML Schema Preferences

To configure options in regards to XML Schema, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XML Parser** > **XML Schema**.

This preferences page allows you to configure the following options:

Default XML Schema version

Allows you to select the version of W3C XML Schema to be used as the default. You can choose XML Schema **1.0** or XML Schema **1.1**.

Note: You are also able to set the XML Schema version using the **Customize** option in the ***New** document wizard*.

Default XML Schema validation engine

Allows you to select the default validation engine to be used for XML Schema. You can choose **Xerces** or **Saxon EE**.

Xerces validation features section

Enable full schema constraint checking

Sets the *schema-full-checking* feature to `true`. This enables a validation of the parsed XML document against a schema (W3C XML Schema or DTD) while the document is parsed.

Enable honour all schema location feature

Sets the *honour-all-schema-location* feature to `true`. All the files that declare W3C XML Schema components from the same namespace are used to compose the validation model. If this option is not selected, only the first W3C XML Schema file that is encountered in the XML Schema import tree is taken into account.

Enable full XPath 2.0 in assertions and alternative types

When selected (default value), you can use the full XPath support in assertions and alternative types. Otherwise, only the XPath support offered by the Xerces engine is available.

Assertions can see comments and processing instructions

Controls whether or not comments and processing instructions are visible to the XPath expression used for defining an assertion in XSD 1.1.

Saxon EE validation features section

Multiple schema imports

Forces `xs:import` to fetch the referenced schema document. By default, the `xs:import` fetches the document only if no schema document for the given namespace has already been loaded. With this option in effect, the referenced schema document is loaded unless the absolute URI is the same as a schema document already loaded.

Assertions can see comments and processing instructions

Controls whether or not comments and processing instructions are visible to the XPath expression used to define an assertion. By default, they are not made visible (unlike Saxon 9.3).

Relax NG Preferences

To configure options in regards to Relax NG, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XML Parser** > **Relax NG**.

The following options are available in this page:

Check feasibly valid

Checks if Relax NG documents can be transformed into valid documents by inserting any number of attributes and child elements anywhere in the tree.

Note: Selecting this option disables the **Check ID/IDREF** option.

Check ID/IDREF

Checks the ID/IDREF matches when a Relax NG document is validated.

Add default attribute values

Default values are given to the attributes of documents validated using Relax NG. These values are defined in the Relax NG schema.

Schematron Preferences

To configure options in regards to Schematron, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XML Parser** > **Schematron**.

The following options are available in this preferences page:

ISO Schematron Section

Optimize (visit-no-attributes)

If your ISO Schematron assertion tests do not contain the attributes axis, you should select this option for faster ISO Schematron validation.

Allow foreign elements (allow-foreign)

Enables support for `allow-foreign` on ISO Schematron. This option is used to pass non-Schematron elements to the generated stylesheet.

Use Saxon EE (schema aware) for xslt2/xslt3 query language binding

When selected, Saxon EE is used for `xslt2/xslt3` query binding. If this option is not selected, Saxon PE is used.

Enable Schematron Quick Fixes (SQF) support

Allows you to enable or disable the support for *Quick Fixes* in Schematron files. This option is selected by default.

Embedded rules query language binding

You can control the query language binding used by the ISO Schematron embedded rules. You can choose between: **xslt1**, **xslt2**, or **xslt3**.

Note: To control the query language binding for standalone ISO Schematron, you need to set the query language to be used with a `queryBinding` attribute on the schema root element.

Message language

This option allows you to specify the language to be used in Schematron validation messages. You can choose between the following:

- **Use the language defined in the application** - The language that is specified in *the **Global Preferences** page* will be used and only the validation messages that match that language will be presented. You can use the **Change application language** link to navigate to the preferences page where you can specify the language to be used in the application.
- **Use the "xml:lang" attribute set on the Schematron root** - The language specified in the `xml:lang` attribute from the Schematron root will be used and only the validation message that match that language will be presented.
- **Ignore the language and show all message** - All messages are displayed in whatever language they are defined within the Schematron schema.
- **Custom** - Use this option to specify a custom language to be used and only the messages that match the specified language will be presented.

Note: In all cases, if the selected language is not available for a validation error or warning, all messages will be displayed in whatever language they are defined with in the Schematron schema.

Schematron 1.5 Section

XPath Version

Allows you to select the version of XPath for the expressions that are allowed in Schematron assertion tests. You can choose between: **1.0**, **2.0**, or **3.0**. This option is applied in both standalone Schematron 1.5 schemas and embedded Schematron 1.5 rules.

Sample XML Files Generator Preferences

The ***Generate Sample XML Files** tool* (available on the **Tools** menu) allows you to generate XML instance documents based on a W3C XML Schema. There are various options that can be configured within the tool and these options are also available in the **Sample XML Files Generator** preferences page. This allows you to set default values for these options. To configure the options for generating the XML files, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **XML** > **Sample XML Files Generator**.

The following options are available:

Generate optional elements

When selected, all elements are generated, including the optional ones (having the `minOccurs` attribute set to 0 in the schema).

Generate optional attributes

When selected, all attributes are generated, including the optional ones (having the `use` attribute set to `optional` in the schema).

Values of elements and attributes

Controls the content of generated attribute and element values. The following choices are available:

- **None** - No content is inserted.
- **Default** - Inserts a default value depending of data type descriptor of the particular element or attribute. The default value can be either the data type name or an incremental name of the attribute or element (according to the global option from the **Sample XML Files Generator** preferences page). Note that type restrictions are ignored when this option is selected. For example, if an element is of a type that restricts an **xs:string** with the **xs:maxLength** facet to allow strings with a maximum length of 3, the XML instance generator tool may generate string element values longer than 3 characters.
- **Random** - Inserts a random value depending of data type descriptor of the particular element or attribute.

  Important: If all of the following are true, the **Generate Sample XML Files** tool outputs invalid values:

  - At least one of the restrictions is a `regexp`.
  - The value generated after applying the regexp does not match the restrictions imposed by one of the facets.

Preferred number of repetitions

Allows you to set the preferred number of repeating elements related to `minOccurs` and `maxOccurs` facets defined in the XML Schema.

- If the value set here is between `minOccurs` and `maxOccurs`, then that value is used.
- If the value set here is less than `minOccurs`, then the `minOccurs` value is used.
- If the value set here is greater than `maxOccurs`, then `maxOccurs` is used.

Maximum recursion level

If a recursion is found, this option controls the maximum allowed depth of the same element.

Type alternative strategy

Used for the `xs:alternative` element from XML Schema 1.1. The possible strategies are:

- **First** - The first valid alternative type is always used.
- **Random** - A random alternative type is used.

Choice strategy

Used for `xs:choice` or `substitutionGroup` elements. The possible strategies are:

- **First** - The first branch of `xs:choice` or the head element of `substitutionGroup` is always used.
- **Random** - A random branch of `xs:choice` or a substitute element or the head element of a `substitutionGroup` is used.

Generate the other options as comments

If selected, generates the other possible choices or substitutions (for `xs:choice` and `substitutionGroup`). These alternatives are generated inside comments groups so you can uncomment and use them later. Use this option with care (for example, on a restricted namespace and element) as it may generate large result files.

Use incremental attribute / element names as default

If selected, the value of an element or attribute starts with the name of that element or attribute. For example, for an a element the generated values are: a1, a2, a3, and so on. If not selected, the value is the name of the type of that element / attribute (for example: `string`, `decimal`, etc.)

Maximum length

The maximum length of string values generated for elements and attributes.

Discard optional elements after nested level

The optional elements that exceed the specified nested level are discarded. This option is useful for limiting deeply nested element definitions that can quickly result in very large XML documents.

Related Information:

Generating Sample XML Files on page 597

XProc Preferences

Oxygen XML Editor includes a built-in XProc engine called *Calabash*. You can add or configure external XProc engines by using the **XProc** preferences page. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XProc**.

When **Show XProc messages** is selected all messages emitted by the XProc processor during a transformation will be presented in the ***Results** view*.

To add an external engine click the **New** button. To configure an existing engine, click the **Edit** button. This opens the **Custom Engine** dialog box that allows you to configure an external engine.

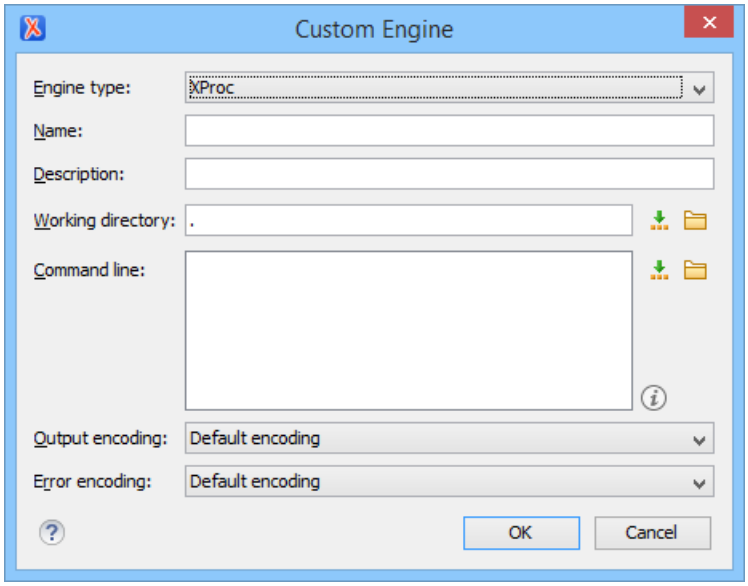


Figure 27: Creating an XProc external engine

The following options can be configure in this dialog box:

- **Name** - The value of this field will be displayed in the XProc transformation scenario and in the command line that will start it.
- **Description** - A textual description that will appear as a tooltip where the XProc engine will be used.
- **Working directory** - The working directory for resolving relative paths. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.
- **Command line** - The command line that will run the XProc engine as an external process. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.
- **Output encoding** - The encoding for the output stream of the XProc engine, used for reading and displaying the output messages.
- **Error encoding** - The encoding for the error stream of the XProc engine, used for reading and displaying the messages from the error stream.

Note: You can configure the built-in Saxon processor using the `saxon.config` file. For further details about configuring this file go to *http://www.saxonica.com/documentation9.5/index.html#!configuration/configuration-file*. You can configure the built-in Calabash processor by using the `calabash.config` file. These files are located in `[OXYGEN_INSTALL_DIR]\lib\xproc\calabash\lib`. If they do not exist, you have to create them.

XSLT-FO-XQuery Preferences

To configure options in regards to XSLT, XQuery, and FO processors, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XSLT-FO-XQuery**. This panel contains only the most generic options for working with XSLT/XSL-FO/XQuery processors. The more specific options are grouped in other panels linked as child nodes of this panel in the tree of this **Preferences** page.

There is only one generic option available:

Create transformation temporary files in system temporary directory

It should be selected only when the temporary files necessary for performing transformations are created in the same folder as the source of the transformation (the default behavior when this option is not selected) and this breaks the transformation. An example of breaking the transformation is when the transformation processes all the files located in the same folder as the source of the transformation (including the temporary files) and the result is incorrect or the transformation fails because of this.

XSLT Preferences

To configure the **XSLT** options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XSLT-FO-XQuery** > **XSLT**.

Oxygen XML Editor offers the possibility of using an XSLT transformer implemented in Java (other than the XSLT transformers that come bundled with Oxygen XML Editor). To use another XSLT transformer, specify the name of the transformer factory class. Oxygen XML Editor sets this transformer factory class as the value of the Java property: `javax.xml.transform.TransformerFactory`.

The XSLT preferences page allows you to customize the following options:

JAXP XSLT Transformer - Value

Allows you to set the value of the `TransformerFactory` Java class.

Validation engine - XSLT 1.0

Allows you to select the XSLT engine to be used for validation of XSLT 1.0 documents.

Validation engine - XSLT 2.0

Allows you to select the XSLT engine to be used for validation of XSLT 2.0 documents.

Validation engine - XSLT 3.0

Allows you to select the XSLT engine to be used for validation of XSLT 3.0 documents.

Note: Saxon-HE does not implement any XSLT 3.0 features. Saxon-PE implements a selection of XSLT 3.0 (and XPath 3.0) features, with the exception of schema-awareness and streaming. Saxon-EE implements additional features relating to streaming (processing of a source document without constructing a tree in memory. For further details about XSLT 3.0 conformance, go to *http://www.saxonica.com/documentation/index.html#!conformance/xslt30*.

XSLT Editor Content Completion Options link

Use this link to switch to the ***XSLT Content Completion** preferences page*, where you can configure the XSLT content completion options.

Saxon6 Preferences

To configure the **Saxon 6** options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XSLT-FO-XQuery** > **XSLT** > **Saxon** > **Saxon6**.

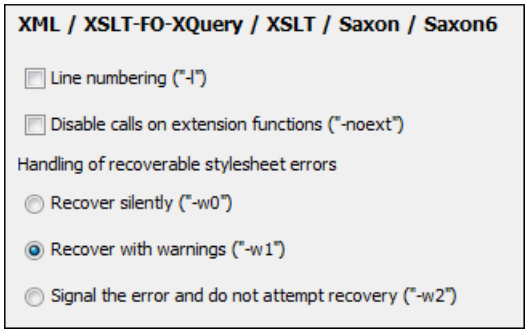


Figure 28: Saxon 6 XSLT Preferences Panel

The built-in Saxon 6 XSLT processor can be configured with the following options:

- **Line numbering** - Specifies whether or not line numbers are maintained and reported in error messages for the XML source document.

- **Disable calls on extension functions** - If selected, external function calls are not allowed. Selecting this option is recommended in an environment where untrusted stylesheets may be executed. It also disables user-defined extension elements and the writing of multiple output files, since they carry similar security risks.
- **Handling of recoverable stylesheet errors** - Allows you to choose how dynamic errors are handled. One of the following options can be selected:
  - **recover silently** - Continue processing without reporting the error.
  - **recover with warnings** - Issue a warning but continue processing.
  - **signal the error and do not attempt recovery** - Issue an error and stop processing.

Saxon-HE/PE/EE Preferences

To configure global options for XSLT transformation and validation scenarios that use the **Saxon HE/PE/EE** engine, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **XML** > **XSLT-FO-XQuery** > **XSLT** > **Saxon** > **Saxon-HE/PE/EE**.

Saxon-HE/PE/EE Options

Oxygen XML Editor allows you to configure the following XSLT options for the Saxon 9.7.0.15 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE):

Use a configuration file ("-config")

Sets a Saxon 9.7.0.15 configuration file that is executed for XSLT transformation and validation processes.

Version warnings ("-versmsg")

Warns you when the transformation is applied to an XSLT 1.0 stylesheet.

Line numbering ("-l")

Line numbers where errors occur are included in the output messages.

Debugger trace into XPath expressions (applies to debugging sessions)

Instructs the *XSLT Debugger* to *step into* XPath expressions.

Expand attributes defaults ("-expand")

Specifies whether or not the attributes defined in the associated DTD or XML Schema are expanded in the output of the transformation you are executing.

DTD validation of the source ("-dtd")

Specifies whether or not the source document will be validated against their associated DTD. You can choose from the following:

- **On** - Requests DTD validation of the source file and of any files read using the `document()` function.
- **Off** - (default setting) Suppresses DTD validation.
- **Recover** - Performs DTD validation but treats the errors as non-fatal.

  Note: Any external DTD is likely to be read even if not used for validation, since DTDs can contain definitions of entities.

Recoverable errors ("-warnings")

Allows you to choose how dynamic errors are handled. The following options can be selected:

- **Recover silently ("silent")** - Continues processing without reporting the error.
- **Recover with warnings ("recover")** - Issues a warning but continues processing.
- **Signal the error and do not attempt recovery ("fatal")** - Issues an error and stops processing.

Strip whitespaces ("-strip")

Allows you to choose how the *strip whitespaces* operation is handled. You can choose one of the following values:

- **All ("all")** - Strips *all* whitespace text nodes from source documents before any further processing, regardless of any `xml:space` attributes in the source document.
- **Ignore ("ignorable")** - Strips all *ignorable* whitespace text nodes from source documents before any further processing, regardless of any `xml:space` attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.

- **None ("none")** - Strips *no* whitespace before further processing.

Optimization level ("-opt")

Allows you to set the optimization level. It is the value is an integer in the range of 0 (no optimization) to 10 (full optimization). This option allows optimization to be suppressed when reducing the compiling time is important, optimization conflicts with debugging, or optimization causes extension functions with side-effects to behave unpredictably.

Saxon-PE/EE Options

The following options are available for Saxon 9.7.0.15 Professional Edition (PE) and Enterprise Edition (EE) only:

Allow calls on extension functions ("-ext")

If selected, the stylesheet is allowed to call external Java functions. This does not affect calls on integrated extension functions, including Saxon and EXSLT extension functions. This option is useful when loading an untrusted stylesheet (such as from a remote site using `http://[URL]`). It ensures that the stylesheet cannot call arbitrary Java methods and thus gain privileged access to resources on your machine.

Register Saxon-CE extension functions and instructions

Registers the Saxon-CE extension functions and instructions when compiling a stylesheet using the Saxon 9.7.0.15 processors.

Note: Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Editor only for developing the Saxon-CE stylesheet, leaving the execution part to a web browser. See more details about *executing such a stylesheet on Saxonica's website*.

Enable assertions ("-ea")

In XSLT 3.0, you can use the **xsl:assert** element to make assertions in the form of XPath expressions, causing a dynamic error if the assertion turns out to be false. If this option is selected, XSLT 3.0 `xsl:assert` instructions are enabled. If it is not selected (default), the assertions are ignored.

Saxon-EE Options

The options available specifically for Saxon 9.7.0.15 Enterprise Edition (EE) are as follows:

Validation of the source file ("-val")

Requests schema-based validation of the source file and of any files read using `document()` or similar functions. It can have the following values:

- **Schema validation ("strict")** - This mode requires an XML Schema and allows for parsing the source documents with strict schema-validation enabled.
- **Lax schema validation ("lax")** - If an XML Schema is provided, this mode allows for parsing the source documents with schema-validation enabled but the validation will not fail if, for example, element declarations are not found.
- **Disable schema validation** - This specifies that the source documents should be parsed with schema-validation disabled.

Validation errors in the result tree treated as warnings ("-outval")

Normally, if validation of result documents is requested, a validation error is fatal. Selecting this option causes such validation failures to be treated as warnings.

Write comments for non-fatal validation errors of the result document

The validation messages for non-fatal errors are written (wherever possible) as a comment in the result document itself.

Generate bytecode ("--generateByteCode:(on|off)")

If you select this option, Saxon-EE attempts to generate Java bytecode for evaluation of parts of a query or stylesheet that are amenable to such an action. For further details regarding this option, go to *http://www.saxonica.com/documentation9.5/index.html#!javadoc*.

Saxon-HE/PE/EE Advanced Preferences

To configure the **Saxon HE/PE/EE Advanced** preferences, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **XML** > **XSLT-FO-XQuery** > **XSLT** > **Saxon** > **Saxon-HE/PE/EE** > **Advanced**.

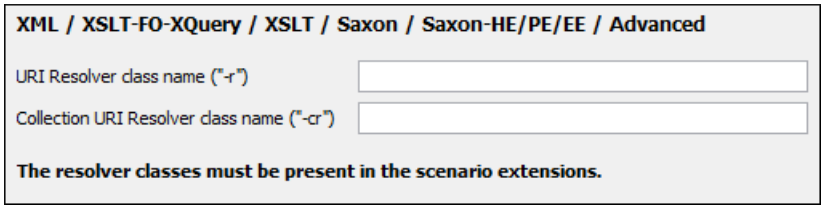


Figure 29: Saxon HE/PE/EE XSLT Advanced Preferences Panel

You can configure the following advanced XSLT options for the Saxon 9.7.0.15 transformer (all three editions: Home Edition, Professional Edition, Enterprise Edition):

- **URI Resolver class name ("-r")** - Specifies a custom implementation for the URI resolver used by the XSLT Saxon 9.7.0.15 transformer (the -r option when run from the command line). The class name must be fully specified and the corresponding `jar` or `class` extension must be configured from *the dialog box for configuring the XSLT extension* for the particular transformation scenario.
- **Collection URI Resolver class name ("-cr")** - Specifies a custom implementation for the Collection URI resolver used by the XSLT Saxon 9.7.0.15 transformer (the -cr option when run from the command line). The class name must be fully specified and the corresponding `jar` or `class` extension must be configured from *the dialog box for configuring the XSLT extension* for the particular transformation scenario.

XSLTProc Preferences

To configure **XSLTProc** options, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **XML** > **XSLT-FO-XQuery** > **XSLT** > **XSLTProc**.

The following options are available in this preferences page:

- **Enable XInclude processing** - If selected, XInclude references will be resolved when XSLTProc is used as transformer in *XSLT transformation scenarios*.
- **Skip loading the document's DTD** - If selected, the DTD specified in the DOCTYPE declaration will not be loaded.
- **Do not apply default attributes from document's DTD** - If selected, the default attributes declared in the DTD and not specified in the document are not included in the transformed document.
- **Do not use Internet to fetch DTD's, entities or docs** - If selected, the remote references to DTD's and entities are not followed.
- **Maximum depth in templates stack** - If this limit of maximum templates depth is reached the transformation ends with an error.
- **Verbosity** - If selected, the transformation will output detailed status messages about the transformation process in the **Warnings** view.
- **Show version of libxml and libxslt used** - If selected, Oxygen XML Editor will display in the **Warnings** view the version of the **libxml** and **libxslt** libraries invoked by XSLTProc.
- **Show time information** - If selected, the **Warnings** view will display the time necessary for running the transformation.
- **Show debug information** - If selected, the **Warnings** view will display debug information about what templates are matched, parameter values, and so on.
- **Show all documents loaded during processing** - If selected, Oxygen XML Editor will display in the **Warnings** view the URL of all the files loaded during transformation.
- **Show profile information** - If selected, Oxygen XML Editor will display in the **Warnings** view a table with all the matched templates, and for each template will display: the match XPath expression, the template name, the number of template modes, the number of calls, the execution time.
- **Show the list of registered extensions** - If selected, Oxygen XML Editor will display in the **Warnings** view a list with all the registered extension functions, extension elements and extension modules.
- **Refuses to write to any file or resource** - If selected, the XSLTProc processor will not write any part of the transformation result to an external file on disk. If such an operation is requested by the processed XSLT stylesheet the transformation ends with a runtime error.

- **Refuses to create directories** - If selected, the XSLTProc processor will not create any directory during the transformation process. If such an operation is requested by the processed XSLT stylesheet the transformation ends with a runtime error.

MSXML Preferences

To configure the MSXML options, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **XML** > **XSLT-FO-XQuery** > **XSLT** > **MSXML**.

The options in this preferences page for the MSXML 3.0 and 4.0 processors are as follows:

Validate documents during parse phase

If selected, and either the source or stylesheet document has a DTD or schema that its content can be checked against, validation is performed.

Do not resolve external definitions during parse phase

By default, MSXML instructs the parser to resolve external definitions such as document type definition (DTD), external subsets or external entity references when parsing the source and style sheet documents. If this option is selected, the resolution is disabled.

Strip non-significant whitespaces

If selected, strips non-significant white space from the input XML document during the load phase. Selecting this option can lower memory usage and improve transformation performance while, in most cases, creating equivalent output.

Show time information

If selected, the relative speed of various transformation steps can be measured, including:

- The time to load, parse, and build the input document.
- The time to load, parse, and build the stylesheet document.
- The time to compile the stylesheet in preparation for the transformation.
- The time to execute the stylesheet.

Start transformation in this mode

Although stylesheet execution usually begins in the empty mode, this default behavior may be changed by specifying another mode. Changing the start mode allows execution to jump directly to an alternate group of templates.

MSXML.NET Preferences

To configure the **MSXML.NET** options, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **XML** > **XSLT-FO-XQuery** > **XSLT** > **MSXML.NET**.

The options in this preferences page for the MSXML.NET processor are as follows:

Enable XInclude processing

If selected, XInclude references will be resolved when MSXML.NET is used as the transformer in the *XSLT transformation scenario*.

Validate documents during parse phase

If selected, and either the source or stylesheet document has a DTD or schema that its content can be checked against, validation is performed.

Do not resolve external definitions during parse phase

By default, MSXML instructs the parser to resolve external definitions such as document type definition (DTD), external subsets or external entity references when parsing the source and style sheet documents. If this option is selected, the resolution is disabled.

Strip non-significant whitespaces

If selected, strips non-significant white space from the input XML document during the load phase. Selecting this option can lower memory usage and improve transformation performance while, in most cases, creating equivalent output.

Show time information

If selected, the relative speed of various transformation steps can be measured, including:

- The time to load, parse, and build the input document.

- The time to load, parse, and build the stylesheet document.
- The time to compile the stylesheet in preparation for the transformation.
- The time to execute the stylesheet.

Forces ASCII output encoding

There is a known problem with the .NET 1.X XSLT processor (`System.Xml.Xsl.XslTransform` class). It does not support escaping of characters as XML character references when they cannot be represented in the output encoding. This means that it will be outputted as `'?'`. Usually this happens when output encoding is set to ASCII. If this option is selected, the output is forced to be ASCII encoded and all non-ASCII characters get escaped as XML character references (&#nnnn; form).

Allow multiple output documents

This option allows you to create multiple result documents using the *`exsl:document` extension element*.

Use named URI resolver class

This option allows you to specify a custom URI resolver class to resolve URI references in `xsl:import` and `xsl:include` instructions (during XSLT stylesheet loading phase) and in `document()` functions (during XSL transformation phase).

Assembly file name for URI resolver class

This option specifies a file name of the assembly where the specified resolver class can be found. The ***Use named URI resolver class** option* specifies a partially or fully qualified URI resolver class name (for example, `Acme.Resolvers.CacheResolver`). Such a name requires additional assembly specification using this option or the ***Assembly GAC name for URI resolver class** option*, but fully qualified class name (which always includes an assembly specifier) is *all-sufficient*. See MSDN for more info about *fully qualified class names*.

Assembly GAC name for URI resolver class

This option specifies partially or fully qualified name of the assembly in the *global assembly cache* (GAC) where the specified resolver class can be found. See MSDN for more info about *partial assembly names*.

List of extension object class names

This option allows to specify *extension object* classes, whose public methods then can be used as extension functions in an XSLT stylesheet. It is a comma-separated list of namespace-qualified extension object class names. Each class name must be bound to a namespace URI using prefixes, similar to providing XSLT parameters.

Use specified EXSLT assembly

MSXML.NET supports a rich library of the *EXSLT* and *EXSLT.NET extension functions* embedded or in a *plugin* EXSLT.NET library. EXSLT support is enabled by default and cannot be disabled in this version. Use this option if you want to use an external EXSLT.NET implementation instead of a built-in one.

Credential loading source xml

This option allows you to specify user credentials to be used when loading XML source documents. The credentials should be provided in the *username:password@domain* format (all parts are optional).

Credential loading stylesheet

This option allows you to specify user credentials to be used when loading XSLT stylesheet documents. The credentials should be provided in the *username:password@domain* format (all parts are optional).

XQuery Preferences

To configure the **XQuery** options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XSLT-FO-XQuery** > **XQuery**.

The following generic XQuery preferences are available:

Validation engine

Allows you to select the processor that will be used to validate XQuery documents. If you are validating an XQuery file that has an associated validation scenario, Oxygen XML Editor uses the processor specified in the scenario. If no validation scenario is associated, but the file has an associated transformation scenario, the processor specified in the scenario is used. If the processor does not support validation or if no scenario is associated, then the value from this combo box will be used as validation processor.

Size limit of Sequence view (MB)

When the result of an XQuery transformation is set as a sequence (***Present as a sequence*** *option*) in the transformation scenario, the size of one chunk of the result that is fetched from the database in lazy mode in one step is set in this option. If this limit is exceeded, go to the ***Sequence*** *view* and click **More results available** to extract more data from the database.

Format transformer output

Specifies whether or not the output of the transformer is formatted and indented (*pretty-print*).

Note: This option is ignored if you choose ***Present as a sequence*** (lazy extract data from a database) from the associated transformation scenario.

Create structure indicating the type nodes

If selected, Oxygen XML Editor takes the results of a query and creates an XML document containing copies of all items in the sequence, suitably wrapped.

Note: This option is ignored if you choose ***Present as a sequence*** (lazy extract data from a database) from the associated transformation scenario.

Saxon-HE/PE/EE Preferences

To configure global options for XQuery transformation and validation scenarios that use the **Saxon HE/PE/EE** engine, *open the* ***Preferences*** *dialog box* ***(Options*** *>* ***Preferences)*** and go to **XML** > **XSLT-FO-XQuery** > **XQuery** > **Saxon-HE/PE/EE**.

Oxygen XML Editor allows you to configure the following XQuery options for the Saxon 9.7.0.15 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE):

Use a configuration file ("-config")

Sets a Saxon 9.7.0.15 configuration file that is used for XQuery transformation and validation scenarios.

Recoverable errors ("-warnings")

Allows you to choose how dynamic errors are handled. The following options can be selected:

- **Recover silently ("silent")** - Continues processing without reporting the error.
- **Recover with warnings ("recover")** - Issues a warning but continues processing.
- **Signal the error and do not attempt recovery ("fatal")** - Issues an error and stops processing.

Strip whitespaces ("-strip")

Allows you to choose how the *strip whitespaces* operation is handled. You can choose one of the following values:

- **All ("all")** - Strips *all* whitespace text nodes from source documents before any further processing, regardless of any `xml:space` attributes in the source document.
- **Ignore ("ignorable")** - Strips all *ignorable* whitespace text nodes from source documents before any further processing, regardless of any `xml:space` attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.
- **None ("none")** - Strips *no* whitespace before further processing.

Optimization level ("-opt")

Allows you to set the optimization level. It is the value is an integer in the range of 0 (no optimization) to 10 (full optimization). This option allows optimization to be suppressed when reducing the compiling time is important, optimization conflicts with debugging, or optimization causes extension functions with side-effects to behave unpredictably.

Use linked tree model ("-tree:linked")

This option activates the linked tree model.

Enable XQuery 3.0 support ("-qversion:(1.0|3.0)")

If selected (default value), Saxon runs the XQuery transformation with the XQuery 3.0 support.

The following option is available for Saxon 9.7.0.15 Professional Edition (PE) and Enterprise Edition (EE) only:

Allow calls on extension functions ("-ext")

If selected, calls on external functions are allowed. Selecting this option is recommended in an environment where untrusted stylesheets may be executed. It also disables user-defined extension elements and the writing of multiple output files, both of which carry similar security risks.

The options available specifically for Saxon 9.7.0.15 Enterprise Edition (EE) are as follows:

Validation of the source file ("-val")

Requests schema-based validation of the source file and of any files read using `document()` or similar functions. It can have the following values:

- **Schema validation ("strict")** - This mode requires an XML Schema and allows for parsing the source documents with strict schema-validation enabled.
- **Lax schema validation ("lax")** - If an XML Schema is provided, this mode allows for parsing the source documents with schema-validation enabled but the validation will not fail if, for example, element declarations are not found.
- **Disable schema validation** - This specifies that the source documents should be parsed with schema-validation disabled.

Validation errors in the result tree treated as warnings ("-outval")

Normally, if validation of result documents is requested, a validation error is fatal. Selecting this option causes such validation failures to be treated as warnings.

Write comments for non-fatal validation errors of the result document

The validation messages for non-fatal errors are written (wherever possible) as a comment in the result document itself.

Generate bytecode ("--generateByteCode:(on|off)")

If you select this option, Saxon-EE attempts to generate Java bytecode for evaluation of parts of a query or stylesheet that are amenable to such an action. For further details regarding this option, go to *http://www.saxonica.com/documentation9.5/index.html#!javadoc*.

Enable XQuery update ("-update:(on|off)")

This option controls whether or not XQuery update syntax is accepted. The default value is off.

Backup files updated by XQuery ("-backup:(on|off)")

If selected, backup versions for any XML files updated with an XQuery Update are generated. This option is available when the **Enable XQuery update** option is selected.

Saxon HE/PE/EE Advanced Preferences

To configure **Saxon HE/PE/EE Advanced** preferences, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XSLT-FO-XQuery** > **XQuery** > **Saxon-HE/PE/EE** > **Advanced**.

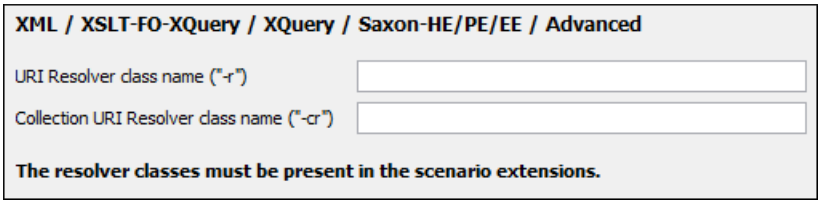


Figure 30: Saxon HE/PE/EE XQuery Advanced Preferences Panel

The advanced XQuery options that can be configured for the Saxon 9.7.0.15 XQuery transformer (all editions: Home Edition, Professional Edition, Enterprise Edition) are as follows:

- **URI Resolver class name** - Allows you to specify a custom implementation for the URI resolver used by the XQuery Saxon 9.7.0.15 transformer (the -r option when run from the command line). The class name must be fully specified and the corresponding *JAR* or class extension must be configured from *the dialog box for configuring the XQuery extension* for the particular transformation scenario.

Note: If your `URIResolver` implementation does not recognize the given resource, the `resolve(String href, String base)` method should return a `null` value. Otherwise, the given resource will not be resolved through the *XML Catalog*.

- **Collection URI Resolver class name** - Allows you to specify a custom implementation for the Collection URI resolver used by the XQuery Saxon 9.7.0.15 transformer (the -cr option when run from the command line). The class name must be fully specified and the corresponding *JAR* or class extension must be configured from *the dialog box for configuring the XQuery extension* for the particular transformation scenario.

Debugger Preferences

To configure the **Debugger** preferences, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XSLT-FO-XQuery** > **Debugger**.

The following options are available:

Show xsl:result-document output

If selected, the debugger presents the output of `xsl:result-document` instructions into the debugger output view.

Infinite loop detection

Select this option to receive notifications when an infinite loop occurs during transformation.

Enable Saxon optimizations

This option is not selected by default and this means that the optimization level for the debugging process is set to zero. If this option is selected, the debugging process will use the optimization level from one of the following:

- The value specified in the ***Optimization level*** *option in an associated XSLT transformation* (or the same *option in an associated XQuery transformation*), if a transformation scenario is used in the debugging process.
- Otherwise, the value specified in the ***Optimization level*** *option in the XSLT Saxon-HE/PE/EE preferences page* (or the same *option in the XQuery Saxon HE/PE/EE preferences page*).

Maximum depth in templates stack

Allows you to set how many `xsl:template` instructions can appear on the current stack. This setting is used by the infinite loop detection.

Debugger layout

If you select the **Horizontal** layout, the stack of XML editors is presented on the left half of the editing area while the stack of XSL editors is on the right half. If you select the **Vertical** layout, the stack of XML editors is presented on the upper half of the editing area while the stack of XSL editors is on the lower half.

Debugger current instruction pointer

Allows you to set the background color of the current execution node, both in the document (XML) and XSLT/XQuery views.

XWatch evaluation timeout (seconds)

Allows you to specify the maximum time that Oxygen XML Editor allocates to the evaluation of XPath expressions while debugging.

Messages

Allows you to specify how to handle the debugging process when the source document involved in a debugging session is edited. You can choose one of the following:

- **Ask me what to do**
- **Always stop the debugging session**
- **Never stop the debugging session**

Profiler Preferences

This section explains the settings available for the XSLT/XQuery Profiler. To access and modify these settings, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XSLT-FO-XQuery** > **Profiler** (see *Debugger Preferences* on page 128).

The following profiler settings are available:

Show time

Shows the total time that was spent in the call.

Show inherent time

Shows the inherent time that was spent in the call. The inherent time is defined as the total time of a call minus the time of its child calls.

Show invocation count

Shows how many times the call was called in this particular call sequence.

Time scale

Determines the unit of time measurement. You can choose between milliseconds or microseconds.

Hotspot threshold

Hotspots are ignored below this specified amount (in milliseconds). For more information, see *Hotspots View* on page 1162.

Ignore invocation less than

Invocations are ignored below this specified amount (in microseconds). For more information, see *Invocation Tree View* on page 1161.

Percentage calculation

The percentage base that determines what time span percentages are calculated against. You can choose between the following:

- **Absolute** - Percentage values show the contribution to the total time.
- **Relative** - Percentage values show the contribution to the calling call.

FO Processors Preferences

Oxygen XML Editor includes a built-in formatting objects processor (Apache FOP), but you can also configure other external processors and use them in the transformation scenarios for processing XSL-FO documents.

Oxygen XML Editor provides an easy way to add two of the most commonly used commercial FO processors: **RenderX XEP** and **Antenna House Formatter**. You can easily add *RenderX XEP* as an external FO processor if you have the XEP installed. Also, if you have the *Antenna House Formatter*, Oxygen XML Editor uses the environment variables set by the XSL formatter installation to detect and use it for XSL-FO transformations. If the environment variables are not set for the XSL formatter installation, you can browse and choose the executable file just as you would for XEP. You can use these two external FO processors for *DITA OT transformations scenarios* and *XML with XSLT transformation scenarios*.

To configure the options for the FO Processors, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XSLT-FO-XQuery** > **FO Processors**. The **FO Processors** preferences page is displayed.

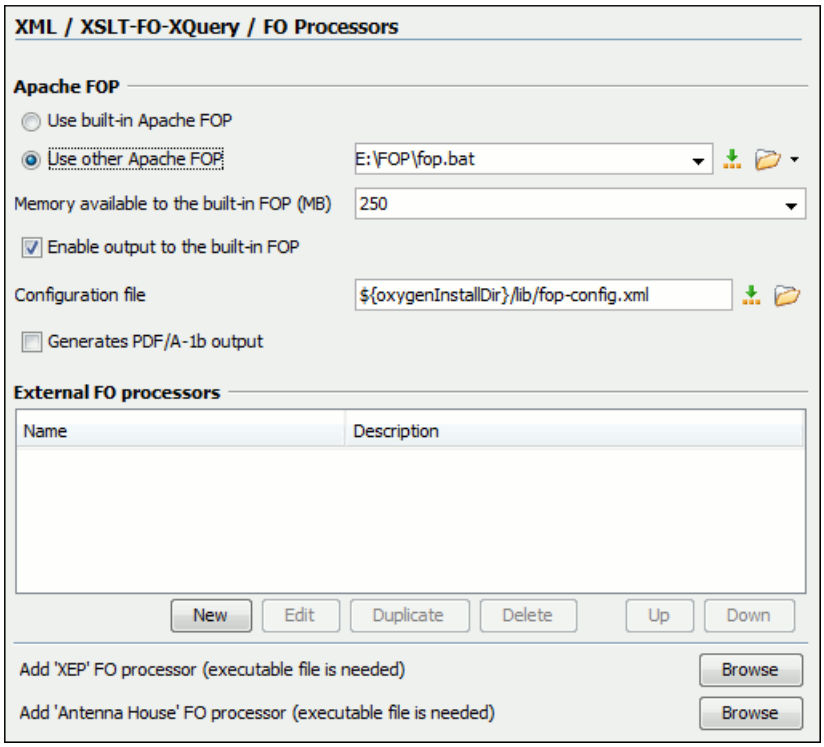


Figure 31: FO Processors Preferences Page

Apache FOP Section

In this section you can configure options for the built-in Apache processor. The following options are available:

Use built-in Apache FOP

Instructs Oxygen XML Editor to use the built-in Apache FO processor.

Use other Apache FOP

Instructs Oxygen XML Editor to use another Apache FO processor that is installed on your computer. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Enable the output of the built-in FOP

All Apache FOP output is displayed in a results pane at the bottom of the Oxygen XML Editor window, including warning messages about FO instructions not supported by Apache FOP.

Memory available to the built-in FOP

If your Apache FOP transformations fail with an Out of Memory error (**OutOfMemoryError**), use this combo box to select a bigger value for the amount of memory reserved for FOP transformations.

Configuration file for the built-in FOP

Use this option to specify the path to an Apache FOP configuration file (for example, to render to PDF a document containing Unicode content using a special *true type* font). You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Generates PDF/A-1b output

When selected, PDF/A-1b output is generated.

Note: All fonts have to be embedded, even the implicit ones. More information about configuring metrics files for the embedded fonts can be found in *Add a font to the built-in FOP*.

Note: You cannot use the `<filterList>` key in the configuration file since the FOP would generate the following error: *The Filter key is prohibited when PDF/A-1 is active*.

External FO Processors Section

In this section you can manage the external FO processors you want to use in transformation scenarios. You can use the two options at the bottom of the section to use the **RenderX XEP** or **Antenna House Formatter** commercial FO processors.

Add 'XEP' FO processor (executable file is needed)

If **RenderX XEP** is already installed on your computer, you can use this button to choose the XEP executable script (`xep.bat` for Windows, `xep` for Linux).

Add 'Antenna House' FO processor (executable file is needed)

If **Antenna House Formatter** is already installed on your computer, you can use this button to choose the Antenna House executable script (`AHFCmd.exe` or `XSLCmd.exe` for Windows, and `run.sh` for Linux/Mac OS).

Note: The built-in **Antenna House Formatter GUI** transformation scenario requires that you configure an external FO processor that runs `AHFormatter.exe` (Windows only). In the *external **FO Processor** configuration dialog box*, you could use `"${env(AHF63_64_HOME)}\AHFormatter.exe" -d ${fo} -s` for the value in the **Command line** field, although the environment variable name changes for each version of the AH Formatter and for each system architecture (you can install multiple versions side-by-side). For more information, see *https://github.com/AntennaHouse/focheck/wiki/focheck*.

You can also add external processors or configure existing ones. Press the **New** button to open a configuration dialog box that allows you to add a new external FO processor. Use the other buttons (**Edit**, **Duplicate**, **Delete**, **Up**, **Down**) to configure existing external processors.

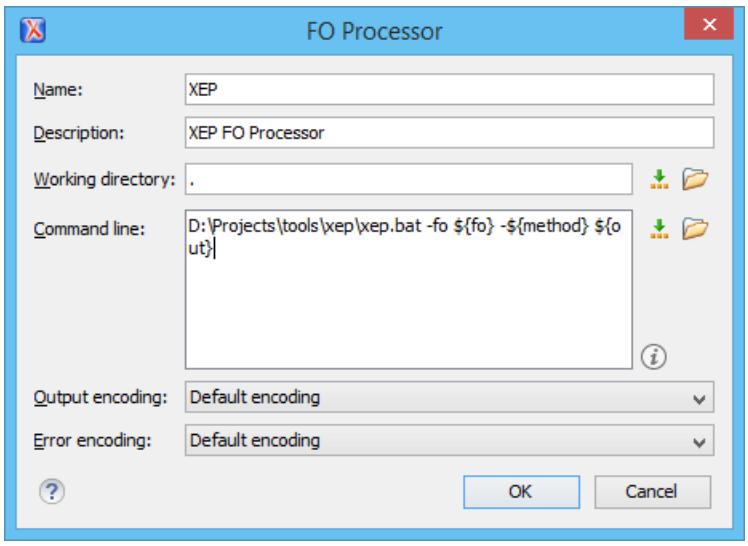


Figure 32: External FO Processor Configuration Dialog Box

The external **FO Processor** configuration dialog box includes the following options:

Name

The name that will be displayed in the list of available FO processors on the FOP tab of the transformation scenario dialog box.

Description

A textual description of the FO processor that will be displayed in the FO processors table and in tooltips of UI components where the processor is selected.

Working directory

The directory where the intermediate and final results of the processing is stored. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button. You can use one of the following *editor variables*:

- **${homeDir}** - The path to the user home directory.

- **${cfd}** - The path of the current file directory. If the current file is not a local file, the target is the user desktop directory.
- **${pd}** - The project directory.
- **${oxygenInstallDir}** -The Oxygen XML Editor installation directory.

Command line

The command line that starts the FO processor, specific to each processor. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button. You can use one of the following *editor variables*:

- **${method}** - The FOP transformation method: **pdf**, **ps**, or **txt**.
- **${fo}** - The input FO file.
- **${out}** - The output file.
- **${pd}** - The project directory.
- **${frameworksDir}** - The path of the `frameworks` subdirectory of the Oxygen XML Editor installation directory.
- **${oxygenInstallDir}** - The Oxygen XML Editor installation directory.
- **${ps}** - The platform-specific path separator. It is used between the library files specified in the class path of the command line.

Output Encoding

The encoding of the FO processor output stream that is displayed in a ***Results*** *panel* at the bottom of the Oxygen XML Editor window.

Error Encoding

The encoding of the FO processor error stream that is displayed in a ***Results*** *panel* at the bottom of the Oxygen XML Editor window.

XPath Preferences

To configure XPath options, *open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **XML** > **XSLT-FO-XQuery** > **XPath**.

Oxygen XML Editor allows you to customize the following options:

Unescape XPath expression

If selected, the entities of an XPath expressions that you type in the ***XPath/XQuery Builder*** and the *XPath toolbar* are unescaped during their execution. For example, the expression:

```
//varlistentry[starts-with(@os,'&#x73;')]
```

is equivalent to:

```
//varlistentry[starts-with(@os,'s')]
```

Multiple XPath results

Select this option to display the results of an XPath expression in separate tabs in the ***Results*** *view*.

XPath Default Namespace (only for XPath version 2.0)

Specifies the default namespace to be used for unprefixed element names. You can choose between the following four options:

- **No namespace** - If selected, Oxygen XML Editor considers unprefixed element names of the evaluated XPath expressions as belonging to no namespace.
- **Use the default namespace from the root element** (default selection) - Oxygen XML Editor considers unprefixed element names of the evaluated XPath expressions as belonging to the default namespace declared on the root element of the XML document you are querying.
- **Use the namespace of the root** - If selected, Oxygen XML Editor considers unprefixed element names of the evaluated XPath expressions as belonging to the same namespace as the root element of the XML document you are querying.
- **This namespace** - If selected, you can use the corresponding text field to enter the namespace of the unprefixed elements.

Default prefix-namespace mappings

You can use this table to associate prefixes with namespaces. Use these mappings when you want to define them globally (not for each document). Use the **New** button to add mappings to the list and the **Delete** button to remove mappings.

Custom Engines Preferences

Oxygen XML Editor allows you to configure custom processors to be used for running XSLT and XQuery transformations.

Note: You can not use these custom engines in *the Debugger perspective*.

To configure the **Custom Engines** preferences, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XSLT-FO-XQuery** > **Custom Engines**.

The table in this preferences page displays the custom engines that have been defined. Use the **New** or **Edit** button at the bottom of the table to open a dialog box that allows you to add or configure a custom engine.

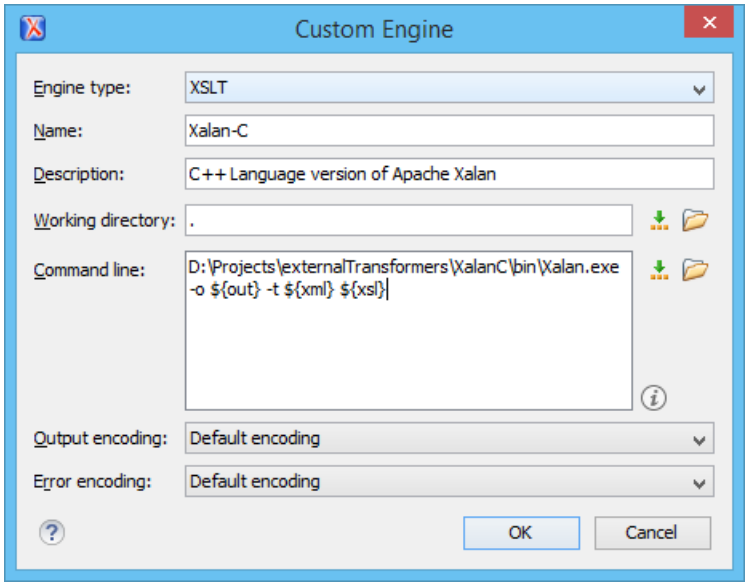


Figure 33: Parameters of a Custom Engine

The following parameters can be configured for a custom engine:

Engine type

Specifies the transformer type. You can choose between XSLT and XQuery engines.

Name

The name of the transformer displayed in the dialog box for editing transformation scenarios.

Description

A textual description of the transformer.

Working directory

The start directory of the executable program for the transformer. The following *editor variables* are available for making the path to the working directory independent of the location of the input files:

- **${homeDir}** - The user home directory in the operating system.
- **${cfd}** - The path to the directory of the current file.
- **${pd}** - The path to the directory of the current project.
- **${oxygenInstallDir}** - The Oxygen XML Editor install directory.

Command line

The command line that must be executed by Oxygen XML Editor to perform a transformation with the engine. The following *editor variables* are available for making the parameters in the command line (the transformer executable, the input files) independent of the location of the input files:

- **${xml}** - The XML input document as a file path.
- **${xmlu}** - The XML input document as a URL.
- **${xsl}** - The XSL / XQuery input document as a file path.
- **${xslu}** - The XSL / XQuery input document as a URL.
- **${out}** - The output document as a file path.
- **${outu}** - The output document as a URL.
- **${ps}** - The platform separator that is used between library file names specified in the class path.

Output Encoding

The encoding of the transformer output stream.

Error Encoding

The encoding of the transformer error stream.

Ant Preferences

To set Ant preferences, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **XML** > **Ant**. This panel allows you to choose the directory containing the *Apache Ant* libraries (the so-called *Ant Home*) that Oxygen XML Editor uses to handle Ant build files.

There are two options available:

- **Built-in** - the path to the Ant distribution that comes bundled with Oxygen XML Editor installation kit.
- **Custom** - the path to an Ant distribution of your choice.

Import Preferences

To configure importing options, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **XML** > **Import**. This page allows you to configure how empty values and `null` values are handled when they are encountered in imported database tables or Excel sheets. Also you can configure the format of date / time values recognized in the imported database tables or Excel sheets.

The following options are available:

Create empty elements for empty values

If selected, an empty value from a database column or from a text file is imported as an empty element.

Create empty elements for null values

If selected, `null` values from a database column are imported as empty elements.

Escape XML content

Selected by default, this option instructs Oxygen XML Editor to escape the imported content to an XML-safe form.

Add annotations for generated XML Schema

If selected, the generated XML Schema contains an annotation for each of the imported table columns. The documentation inside the annotation tag contains the remarks of the database columns (if available) and also information about the conversion between the column type and the generated XML Schema type.

Date / Time Format section

Specifies the format used for importing date and time values from Excel spreadsheets or database tables, and in the generated XML schemas. You can choose from the following format types:

- **Unformatted** - The date and time formats specific to the database are used for import. When importing data from Excel a string representation of date or time values are used. The type used in the generated XML Schema is `xs:string`.
- **XML Schema date format** -The XML Schema-specific format ISO8601 is used for imported date / time data (`yyyy-MM-dd'T'HH:mm:ss` for `datetime`, `yyyy-MM-dd` for `date` and `HH:mm:ss` for `time`). The types used in the generated XML Schema are `xs:datetime`, `xs:date` and `xs:time`.
- **Custom format** - If selected, you can define a custom format for timestamp, date, and time values or choose one of the predefined formats. A preview of the values is presented when a format is used. The type used in the generated XML Schema is `xs:string`.

Table 3: Pattern Letters

Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day in week	Text	Tuesday; Tue
a	Am / pm marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am / pm (0-11)	Number	0
h	Hour in am / pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	RFC 822 time zone	-0800

Pattern letters are usually repeated, as their number determines the exact presentation:

- *Text* - If the number of pattern letters is 4 or more, the full form is used. Otherwise, a short or abbreviated form is used if available.
- *Number* - The number of pattern letters is the minimum number of digits, and shorter numbers are zero-padded to this amount.
- *Year* - If the number of pattern letters is 2, the year is truncated to 2 digits. Otherwise, it is interpreted as a number.
- *Month* - If the number of pattern letters is 3 or more, the month is interpreted as text. Otherwise, it is interpreted as a number.
- *General time zone* - Time zones are interpreted as text if they have names. For time zones representing a GMT offset value, the following syntax is used:
  - *GMTOffsetTimeZone* - GMT Sign Hours : Minutes
  - *Sign* - one of + or -
  - *Hours* - one or two digits
  - *Minutes* - two digits
  - *Digit* - one of 0 1 2 3 4 5 6 7 8 9

  Hours must be between 0 and 23, and Minutes must be between 00 and 59. The format is locale independent and digits must be taken from the Basic Latin block of the Unicode standard.
- *RFC 822 time zone*: The RFC 822 4-digit time zone format is used:
  - *RFC822TimeZone*

- *TwoDigitHours* (must be between 00 and 23)

XML Signing Certificates Preferences

Oxygen XML Editor provides two types of *keystores* for certificates that are used for digital signatures of XML documents: Java Keystore (**JKS**) and Public-Key Cryptography Standards version 12 (**PKCS-12**). A *keystore* file is protected by a password. To configure a certificate *keystore*, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **XML** > **XML Signing Certificates**. You can customize the following parameters of a *keystore*:

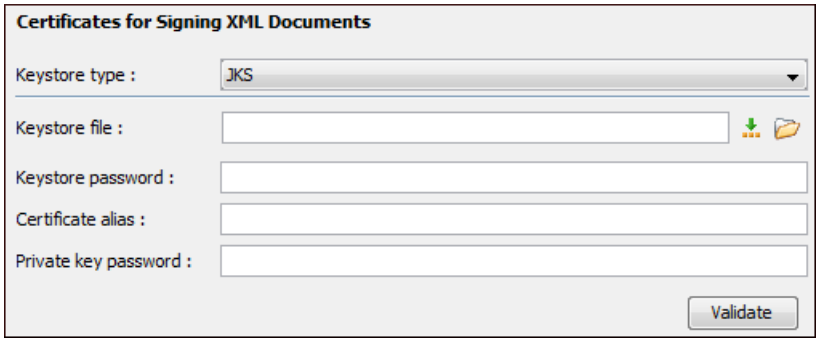


Figure 34: Certificates Preferences Panel

- **Keystore type** - The type of *keystore* that Oxygen XML Editor uses (**JKS** or **PKCS-12**).
- **Keystore file** - The location of the imported file.
- **Keystore password** - The password that is used for protecting the privacy of the stored keys.
- **Certificate alias** - The alias used for storing the key entry (the certificate or the private key) inside the *keystore*.
- **Private key password** - The private key password of the certificate (required only for JKS *keystores*).
- **Validate** - Press this button to verify the configured *keystore* and the validity of the certificate.

XML Refactoring Preferences

To specify a folder for loading the custom XML refactoring operations, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **XML** > **XML Refactoring**. The following option is available in this preferences page:

Load additional refactoring operations from

Use this text box to specify a folder for loading custom XML refactoring operations. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

DITA Preferences

To access the DITA Preferences page, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **DITA**. This preferences page includes the following sections and options:

DITA Open Toolkit section

This section allows you to specify the default directory of the DITA Open Toolkit distribution (bundled with the Oxygen XML Editor installation) to be used for validating and publishing DITA content. You can select from the following:

Built-in DITA-OT 1.8

If this is set, all defined DITA transformation scenarios will run with DITA-OT 1.8.5. The built-in DITA OT 1.8.5 directory is: `[OXYGEN_INSTALL_DIR]/frameworks/dita/DITA-OT`.

Built-in DITA-OT 2.x (with support for DITA 1.3 and Lightweight DITA)

Starting with Oxygen 18.0, this is the default setting. All defined DITA transformation scenarios will run with DITA-OT 2.4.4. This also gives you access to DITA 1.3 file templates when you *create new documents from templates*. The default DITA OT 2.4.4 directory is: `[OXYGEN_INSTALL_DIR]/frameworks/dita/DITA-OT2.x`.

Custom

Allows you to specify a custom directory for your DITA OT distribution.

Location

You can either provide a new file path for the specific DITA OT that you want to use or select a previously used one from the drop-down list. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

DITA Maps file patterns

Allows you to specify the extension types that will be handled as *DITA maps* when opened in Oxygen XML Editor.

When opening a map

Oxygen XML Editor can open a *DITA map* in the regular editor view or in the ***DITA Maps Manager***. This options allows you to specify how a map will be opened. You can choose one of the following options:

- **Always open in the DITA Maps Manager** - A *DITA map* file is always opened in the **DITA Maps Manager** view.
- **Always open as XML** - A *DITA map* file is always opened in the XML editor.
- **Always ask** - When opening a *DITA map*, you are prompted to choose between opening it in the XML editor panel or in the **DITA Maps Manager** view.

Prefer navigation title for topicref rendering

If selected and there is a `navtitle` attribute set on a `topicref`, then the `navtitle` is used to render the title of the topic in the ***DITA Maps Manager***.

Insert topic reference section

Allows you to specify that when inserting a topic reference (using the ***Insert Reference*** *dialog box* and ***Edit Properties*** *dialog box*), the values for certain attributes will always be automatically populated with a detected value (based on the specifications), even if it is the same as the default value. You can choose to always populate the values for the following attributes:

- **Format** - If selected, the attribute will always be automatically populated with a detected value.
- **Scope** - If selected, the `sformatcope` attribute will always be automatically populated with a detected value.
- **Type** - If selected, the `type` attribute will always be automatically populated with a detected value.
- **Navigation title** - If selected, the `navtitle` attribute will always be automatically populated with a detected value.

Insert link section

Allows you to specify that when a link reference is inserted (using actions in the **Link** drop-down menu), the values for certain attributes will always be automatically populated with a detected value (based on the specifications), even if it is the same as the default value. You can choose to always populate the values for the following attributes:

- **Format** - If selected, the `format` attribute will always be automatically populated with a detected value.
- **Scope** - If selected, the `scope` attribute will always be automatically populated with a detected value.
- **Type** - If selected, the `type` attribute will always be automatically populated with a detected value.

Use '.' instead of the ID of the parent topic (DITA 1.3)

When addressing a non-topic element within the topic that contains the URI reference, the URI reference can use an abbreviated fragment-identifier syntax that replaces the topic ID with "`.`" (`#./elementId`). For more information, see *https://www.oxygenxml.com/dita/1.3/specs/index.html#archSpec/base/uri-based-addressing.html*.

File name generation rules for new topics section

The options in this section pertain to the rules that will be used to generate file names in the *New DITA File dialog box* if the **Use the title to generate the file name** option is selected.

Replace spaces with

If selected, the file name generation mechanism will replace spaces in the title with the character entered in this option.

Lower case only

Select this option if you want the file name generation mechanism to only use lower case letters.

Use camel case

If selected, the file name generation mechanism will convert the title to a file name using the *camel case* convention where the first word starts with a lower case letter and all subsequent words begin with upper case (for example, `myFileName`).

Upper case first letter

Select this option if you want the file name generation mechanism to convert the title to a file name using the *camel case* convention but with an upper case letter for the first word (for example, `MyFileName`).

Show console output

Allows you to specify when to display the console output log. The following options are available:

- **When build fails** - displays the console output log if the build fails.
- **Always** - displays the console output log, regardless of whether or not the build fails.

Profiling Attributes link

Link to the ***Profiling Attributes*** *preferences page*, where you can configure how profiling and conditional text is displayed in **Author** mode.

Data Sources Preferences

To configure the **Data Sources** preferences, *open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Data Sources**. This preferences page allows you to configure data sources and connections to relational and native XML databases. For a list of drivers that are available for the major database servers, see *Download Links for Database Drivers* on page 141.

Connection Wizards Section

Create eXist-db XML connection

Click this link to open the dedicated ***Create eXist-db XML connection*** *dialog box* that provides a quick way to create an eXist connection.

Data Sources Section

This section allows you to add and configure data sources.

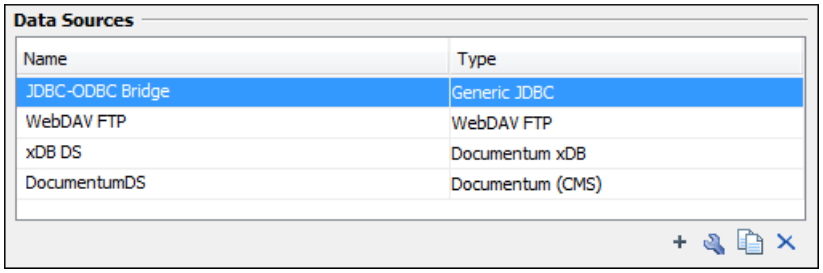


Figure 35: Data Sources Preferences Panel

The following buttons are available at the bottom of the **Data Sources** panel:

+ New

Opens the **Data Sources Drivers** dialog box that allows you to configure a new database driver.

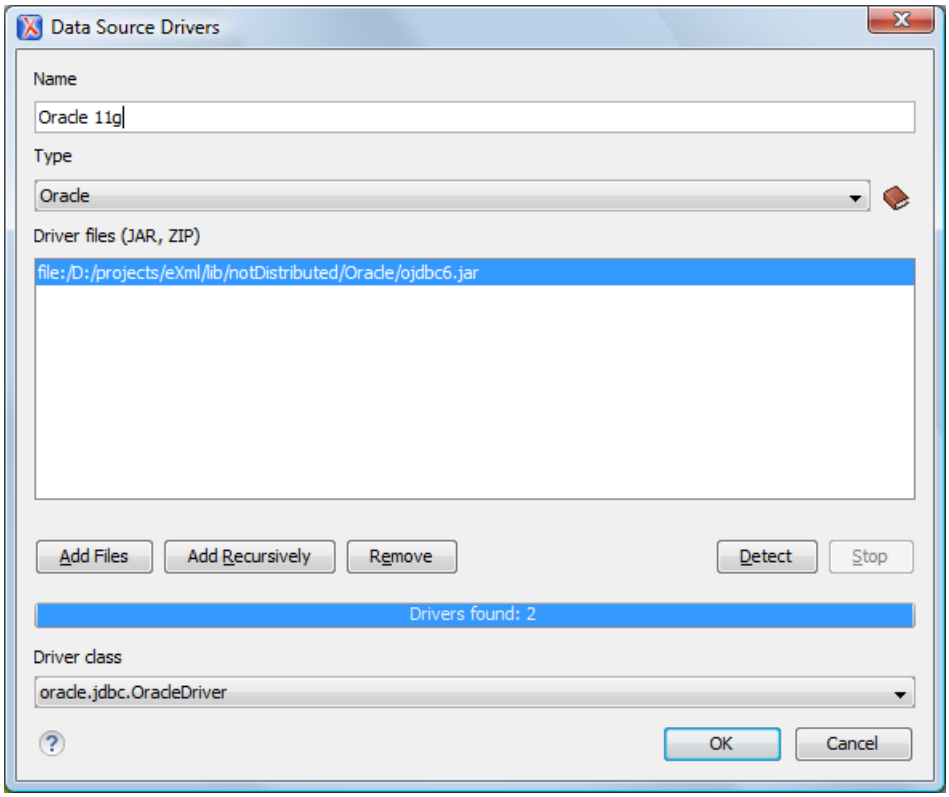


Figure 36: Data Sources Drivers Dialog Box

The following options are available in the **Data Source Drivers** dialog box:

- **Name** - The name of the new data source driver that will be used for creating connections to the database.
- **Type** - Selects the data source type from the supported driver types.
  - **Help button** - Opens the User Manual at *the list of the sections* where the configuration of supported data sources is explained and the URLs for downloading the database drivers are specified.
- **Driver files (JAR, ZIP)** - Lists *download links for database drivers* that are necessary for accessing databases in Oxygen XML Editor.
- **Add Files** - Adds the driver class library.
- **Add Recursively** - Adds driver files recursively.
- **Remove** - Removes the selected driver class library from the list.
- **Detect** - Detects driver file candidates.
- **Stop** - Stops the detection of the driver candidates.
- **Driver class** - Specifies the driver class for the data source driver.

Edit

Opens the **Data Sources Drivers** dialog box for editing the selected driver. See above the specifications for the **Data Sources Drivers** dialog box. To edit a data source, there must be no connections using that data source driver.

Duplicate

Creates a copy of the selected data source.

Delete

Deletes the selected driver. To delete a data source, there must be no connections using that data source driver.

Connections Section

This section allows you to add and configure data source connections.

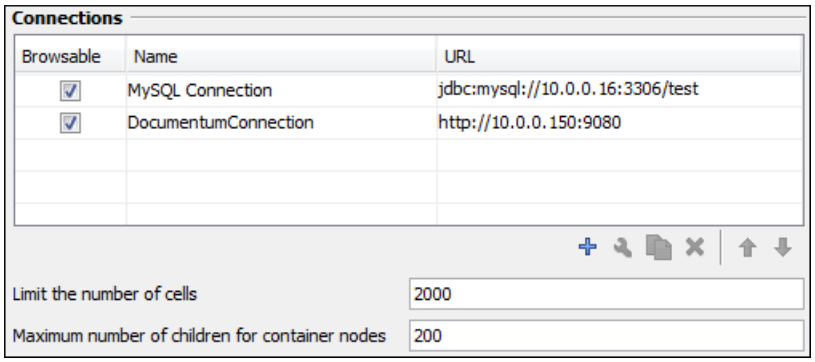


Figure 37: Connections Preferences Panel

The following buttons and options are available at the bottom of the **Connections** panel:

New

Opens the **Connection** dialog box that allows you to configure a new database connection.

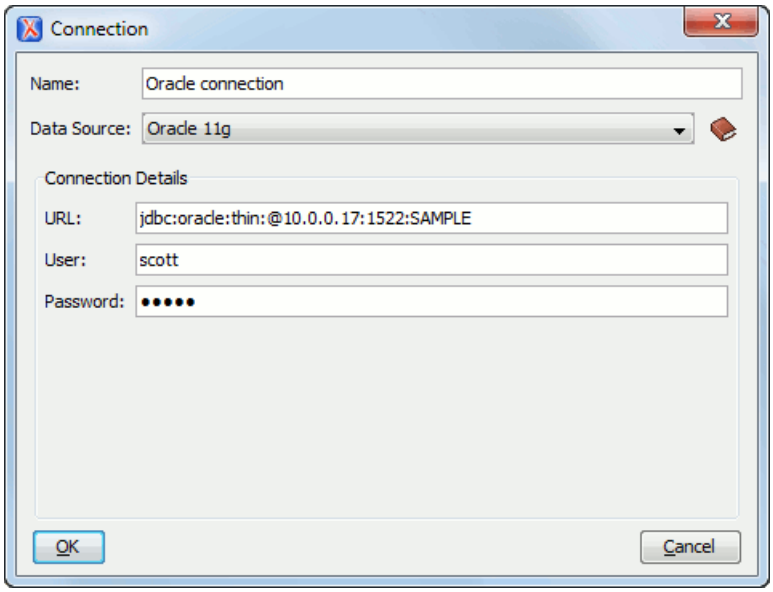


Figure 38: Connection Dialog Box

The following options are available in the **Connection** dialog box:

- **Name** - The name of the new connection that will be used in transformation scenarios and validation scenarios.
- **Data Source** - Allows selecting a data source defined in the **Data Source Drivers** dialog box.

Depending upon the selected data source, you can set some of the following parameters in the **Connection details** area:

- **URL** - The URL for connecting to the database server.
- **User** - The user name for connecting to the database server.
- **Password** - The password of the specified user name.
- **Host** - The host address of the server.
- **Port** - The port where the server accepts the connection.
- **XML DB URI** - The database URI.
- **Database** - The initial database name.
- **Collection** - One of the available collections for the specified data source.
- **Environment home directory** - Specifies the home directory (only for a Berkeley database).
- **Verbosity** - Sets the verbosity level for output messages (only for a Berkeley database).

- **Use a secure HTTPS connection (SSL)** - Allows you to establish a secure connection to an eXist database through the SSL protocol.

Edit

Opens the **Connection** dialog box, allowing you to edit the selected connection. See above the specifications for the **Connection** dialog box.

Duplicate

Creates a copy of the selected connection.

Delete

Deletes the selected connection.

Move Up

Moves the selected connection up one row in the list.

Move Down

Moves the selected connection down one row in the list.

Limit the number of cells

For performance issues, you can set the maximum number of cells that will be displayed in the ***Table Explorer*** *view* for a database table. Leave this field empty if you want the entire content of the table to be displayed. By default, this field is set to 2000. If a table that has more cells than the value set here is displayed in the ***Table Explorer*** *view*, a warning dialog box will inform you that the table is only partially shown.

Maximum number of children for container nodes

In Oracle XML, a container can hold millions of resources. If the node corresponding to such a container in the ***Data Source Explorer*** *view* would display all the contained resources at the same time, the performance of the view would be very slow. To prevent this, only a limited number of the contained resources is displayed as child nodes of the container node. You can navigate to other contained resources from the same container by using the *Up* and *Down* buttons in the ***Data Source Explorer*** *view*. This limited number is set in the field. The default value is 200 nodes.

Table Filters Preferences

The **Table Filters** preferences page allows you to choose the types of tables to be shown in the ***Data Source Explorer*** *view*. To open this preferences page, *open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Data Sources** > **Table Filters**.

You can choose to display the following types of tables:

- **Alias**
- **Global Temporary**
- **Local Temporary**
- **Synonym**
- **System Table**
- **Table**
- **View**

Download Links for Database Drivers

For a list of major relational databases and the drivers that are available for them, see *https://www.oxygenxml.com/database_drivers.html*.

In addition, the following is a list of other popular databases along with instructions for getting the drivers that are necessary to access the databases in Oxygen XML Editor:

- **Berkeley DB XML database** - Copy the *jar* files from the Berkeley database install directory into the Oxygen XML Editor install directory as described in *the procedure for configuring a Berkeley DB data source*.
- **IBM DB2 Pure XML database** - Go to the *IBM website* and in the *DB2 Clients and Development Tools* category select the *DB2 Driver for JDBC and SQLJ* download link. Fill out the download form and download the zip file. Unzip the zip file and use the `db2jcc.jar` and `db2jcc_license_cu.jar` files in Oxygen XML Editor for *configuring a DB2 data source*.

- **eXist database** - Copy the *jar* files from the eXist database install directory to the Oxygen XML Editor install directory as described in *the procedure for configuring an eXist data source*.
- **MarkLogic database** - Download the MarkLogic driver from *MarkLogic Community site*.
- **Microsoft SQL Server 2005 / 2008 database** - Download the appropriate MS SQL JDBC driver from the Microsoft website. For SQL Server 2008 R2 and older go to *http://www.microsoft.com/en-us/download/details.aspx?id=21599*. For SQL Server 2012 and 2014 go to *http://www.microsoft.com/en-us/download/details.aspx?id=11774*.
- **Oracle 11g database** - Go to *http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html* and download the Oracle 11g JDBC driver called `ojdbc6.jar`.
- **PostgreSQL 8.3 database** - Go to *http://jdbc.postgresql.org/download.html* and download the PostgreSQL 8.3 JDBC3 driver.
- **Documentum xDB (X-Hive/DB) 10 XML database** - Copy the *jar* files from the Documentum xDB (X-Hive/DB) 10 database install directory to the Oxygen XML Editor install directory as described in *the procedure* for configuring a Documentum xDB (X-Hive/DB) 10 data source.

SVN Preferences

To configure the options for the SVN client tool, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **SVN**. Some other preferences for the embedded SVN client tool can be set in the global files called `config` and `servers`. These files contain parameters that act as defaults applied to all the SVN client tools that are used by the same user on their computer login account. To open these files for editing, launch the embedded SVN client tool (**Tools** > **SVN Client**) and select **Global Runtime Configuration** > **Edit 'config' file** or **Global Runtime Configuration** > **Edit 'servers' file** from the SVN client **Options** menu.

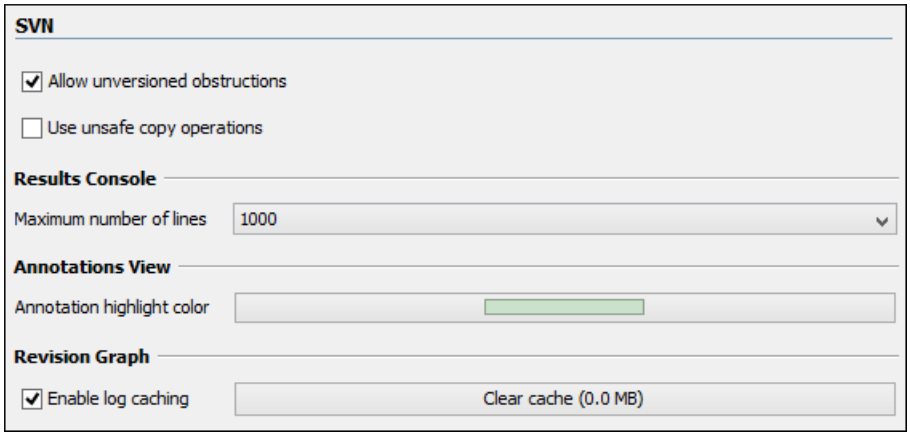


Figure 39: SVN Preferences Panel

The following SVN options can be configured in this preferences page:

Enable symbolic link support (*available only on Mac OS X and Linux*)

Apache Subversion™ has the ability to put a symbolic link under version control, via the usual SVN `add` command. The Subversion repository has no internal concept of a symbolic link. It stores a versioned symbolic link as an ordinary file with a `svn:special` property attached. On Unix/Linux, the SVN client sees the property and translates the file into a symbolic link in the working copy. If the symbolic link support is disabled, the versioned symbolic links appear as a text file instead of symbolic link.

Note: Windows file systems have no symbolic links, so a Windows client will not do any such translation and the object appears as a normal file.

Important: It is recommended to disable symbolic links support if you do not have versioned symbolic links in your repository, since the SVN operations will work faster. However, you should not disable this option when you do have versioned symbolic links in repository. In that case a workaround would be to reference the working copy by its real path, instead of a path that includes a symbolic link.

Allow unversioned obstructions

Controls how to handle a situation where working copy resources are ignored / unversioned when performing an update operation and incoming files (from the repository) with the same name and location intersect with

those being ignored / unversioned. If the option is selected, the incoming items will become BASE revisions of the ones already present in the working copy, and those present will be made versioned resources and will be marked as modified (exactly as if the user first made the update operation and then modified the files). If the option is not selected, the update operation will fail when encountering files in this situation, possibly leaving other files not updated. By default, this option is selected.

Use unsafe copy operations

Sometimes when the working copy is accessed through Samba and the SVN client cannot make a safe copy of the committed file due to a delay in getting a write permission, the result is that the committed file will be saved with zero length (the content is removed) and an error will be reported. In this case, this option should be selected so that the SVN client does not try to make the safe copy.

HTTPS encryption protocols *(available if you are using Java version 1.6 or older)*

Sets a specific encryption protocol to be used when the *application accesses a repository through HTTPS protocol*. You can choose one of the following values:

- **SSLv3, TLSv1** (default value)
- **SSLv3 only**
- **TLSv1 only**

Results Console

Specifies the maximum number of lines displayed in the **Console** view. The default value is 1000.

Annotations View

Sets the color used in the editor panel for highlighting all the changes contributed to a resource by the revision selected in *the **Annotations** view*.

Revision Graph

Enables caching for the action of computing a revision graph. When a new revision graph is requested, one of the caches from the previous actions may be used that will avoid running the whole query again on the SVN server. If a cache is used, it will finish the action much faster.

Working Copy Preferences

To configure the **Working Copy** preferences, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **SVN** > **Working Copy**. The options in this preferences page are specific to SVN working copies and they include the following:

Working copies location

Allows you to define a location where you keep your working copies. This location is automatically suggested when you checkout a new working copy.

Working copy administrative directory

Allows you to customize the directory name where the SVN entries are kept for each directory in the working copy.

When loading an old format working copy

You can instruct the SVN client to do one of the following:

- **Always ask** - You are notified when such a working copy is used and you are allowed to choose what action to be taken - to upgrade or not the format of the current working copy.
- **Never upgrade** - Older format working copies are left untouched. No attempt to upgrade the format is made.

  Note: SVN 1.6 and older working copies still need to be upgraded before loading them.

Enable working copy caching

If selected, the content of the working copies is cached for refresh operations.

Automatically refresh the working copy

If selected, the working copy is refreshed from cache. Only the new changes (modifications with a date/time that follows the last refresh operation) are refreshed from disk. This option is not selected by default.

Allow moving/renaming mixed revision directories

If selected, Oxygen XML Editor will allow you to move or rename a directory even if its child items have a different revision. Otherwise, an error message is displayed when there are multiple revisions to avoid unnecessary conflicts. It is recommended to leave this option deselected and to **Update** the subtree to a single revision before moving or renaming it.

When synchronizing with repository

The action that will be executed automatically after the **Synchronize** action. The possible actions are:

- **Always switch to 'Modified' mode** - The **Synchronize** action is followed automatically by a switch to **Modified** mode of **Working Copy** view, if **All Files** mode is currently selected.
- **Never switch to 'Modified' mode** - Keeps the currently selected view mode unchanged.
- **Always ask** - The user is always asked if they want to switch to **Modified** mode.

Application global ignores

Allows you to set file patterns that may include the `*` and ? wildcards for unversioned files and folders that must be ignored when displaying the working copy resources in *the **Working Copy** view*. These patterns are case-sensitive. For example,`*.txt` matches `file.txt`, but does not match `file.TXT`.

Diff Preferences

To configure the SVN Diff options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Diff**.

The following option is available:

Compare With External Application

Specifies an external application to be launched for compare operations in the following cases:

- When two history revisions are compared.
- When the working copy file is compared with a history revision.
- When a conflict is edited.

The parameters ${firstFile} and ${secondFile} specify the positions of the two compared files in the command line for the external diff application. The parameter ${ancestorFile} specifies the common ancestor (that is, the BASE revision of a file) in a three-way comparison. The working copy version of a file is compared with the repository version, with the BASE revision (the latest revision read from the repository by an Update or Synchronize operation) being the common ancestor of these two compared versions.

Important: If the path to the external compare application includes spaces (or any of the subsequent options or arguments), then each of these paths or *tokens* must be double-quoted for the Oxygen XML Editor to correctly parse and identify them. For example, `C:\Program Files\compareDir\app name.exe` must be written as `"C:\Program Files\compareDir\app name.exe"`.

Messages Preferences

The **Messages** preferences page allows you to disable certain warning messages that may appear in the application. To configure these options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **SVN** > **Messages**.

This preferences page allows you to disable the following warning messages:

Show confirmation dialog when using the "Update All" action

Allows you to avoid performing accidental update operations by requesting you to confirm them before execution.

Show confirmation dialog for drag and drop actions in Working Copy

This option avoids doing a drag and drop when you just want to select multiple files in the **Working Copy** view.

Show warning dialog when editing conflicts

When the **Edit Conflicts** action is executed, a warning dialog box notifies you that the action overwrites the conflicted version of the file created by an update operation. The conflicted file is overwritten with the version of the same file that existed in the working copy before the update operation and then *proceeds with the visual editing of the conflicting file*.

Show warning dialog when "svn:externals" definitions are ignored

A warning dialog box is displayed when "svn:externals" definitions are ignored before performing any operation that updates resources of the working copy (such as *Update* and *Override and Update*).

Diff Preferences

The Diff Preferences Page has sub-pages for configuring File Comparisons and Directory Comparisons.

Files Comparison Preferences

To configure the **Files Comparison** options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Diff** > **Files Comparison**.

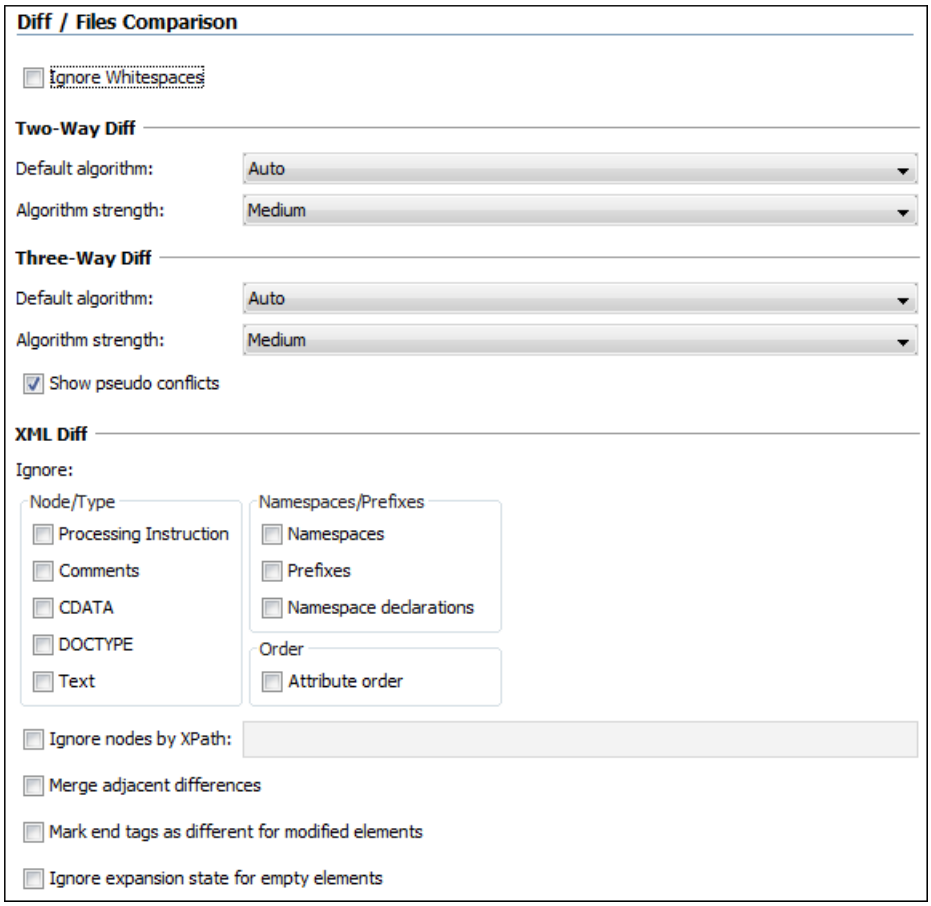


Figure 40: Files Comparison Preferences Page

This preferences page allows you to configure the following options:

Ignore Whitespaces

If selected, before performing the comparison, the application normalizes the content (collapses any sequence of whitespace characters into a single space) and trims its leading and trailing whitespaces.

Note: If the **Ignore Whitespaces** checkbox is selected, comparing the a b sequence with a b, Oxygen XML Editor finds no differences, because after normalization, all whitespaces from the first sequence are collapsed into a single space character. However, when comparing a b with ab (no whitespace between a and b), Oxygen XML Editor signals a difference.

Two-Way Diff section

Default algorithm

The default algorithm used for comparing two files. The following options are available:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).

- **Characters** - Computes the differences at character level, meaning that it compares two files or fragments looking for identical characters.
- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words.
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **Syntax Aware** - Computes differences for the file types or fragments known by Oxygen XML Editor, taking the syntax (the specific types of tokens) into consideration.
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Algorithm strength

Controls the amount of resources allocated to the application to perform the comparison. The algorithm stops searching more differences when reaches the maximum allowed resources. A dialog box is displayed when this limit is reached and partial results are displayed. Four settings are available: **Low**, **Medium** (default), **High** and **Very High**.

Three-Way Diff section

Default algorithm

The default algorithm used for performing a three-way comparison. The following options are available:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Algorithm strength

Controls the amount of resources allocated to the application to perform the comparison. The algorithm stops searching more differences when reaches the maximum allowed resources. A dialog box is displayed when this limit is reached and partial results are displayed. Four settings are available: **Low**, **Medium** (default), **High** and **Very High**.

Show pseudo conflicts

Specifies whether or not the file comparison displays pseudo-conflicts. A pseudo-conflict occurs when two users make the same change (for example, when they both add or remove the same line of code).

XML Diff section

Ignore

Allows you to specify the types of XML nodes that will be ignored in the file comparison for the **XML Fast** and **XML Accurate** algorithms.

Ignore nodes by XPath

If selected, you can enter an *XPath expression* to ignore certain nodes from the comparison. It will be processed as XPath version 2.0. The XPath expression specified in this option is used as the default ignore instructions **only** when the application is started. If you enter an XPath expression in the similar option on the **Diff Files** toolbar, that expression will be used instead.

Merge adjacent differences

If selected, the application considers two adjacent differences as one when the differences are painted in the side-by-side editors. If not selected, every difference is represented separately.

Mark end tags as different for modified elements

If selected, end tags of modified elements are also presented as differences. Otherwise, only the start tags are presented as differences.

Ignore expansion state for empty elements

If selected, empty elements in both expansion states are considered matched (that is `<element/>` and `<element></element>` are considered equal).

Appearance Preferences

To configure the appearance options for the Files Comparison tool, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Diff** > **Files Comparison** > **Appearance**. This preferences page offers the following options:

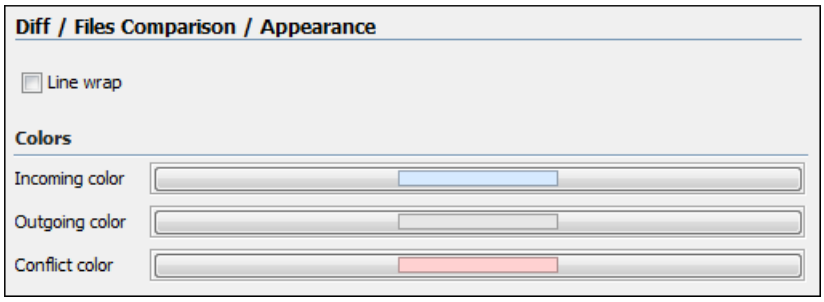


Figure 41: Files Comparison Appearance Preferences Panel

Line wrap

Wraps the lines presented in the two diff panels at the right margin of each panel, so no horizontal scrollbar is necessary.

Incoming color

Specifies the color used on the vertical bar for incoming changes.

Outgoing color

Specifies the color used on the vertical bar for outgoing changes.

Conflict color

Specifies the color used on the vertical bar for conflicts between the compared files.

Directories Comparison Preferences

To configure the **Directories Comparison** preferences, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Diff** > **Directories Comparison**.

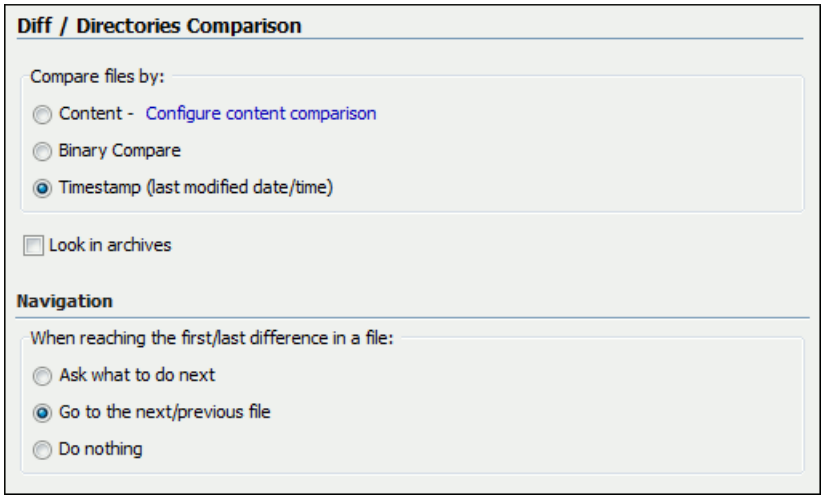


Figure 42: Diff Preferences Page

For the directories comparison, you can specify the following options:

Compare files by

Controls the method used for comparing two files:

- **Content** - The file content is compared using the current *diff algorithm*. This option is applied for a pair of files only if that file type is associated with a built-in editor type (either associated by default or associated by the user when prompted to do so on opening a file of that type for the first time).

  You can use the **Configure content comparison** link to open the ***Files Comparison*** *preferences page* where you can configure options for comparing files. However, the **Ignore nodes by XPath** option is ignored when using the **Compare Directories** tool.
- **Binary Compare** - The files are compared at byte level.
- **Timestamp (last modified date / time)** - The files are compared only by their last modified timestamp.

Look in archives

If selected, *known archive types* are considered directories and their content is compared just like regular files.

Navigation

This options control the behavior of the differences traversal actions (**Go to previous modification**, **Go to next modification**) when the first or last difference in a file is reached:

- **Ask what to do next** - A dialog box is displayed asking you to confirm that you want the application to display modifications from the previous or next file.
- **Go to the next/previous file** - The application opens the next or previous file without waiting for your confirmation.
- **Do nothing** - No further action is taken.

Appearance Preferences

To configure the appearance options for the Directories Comparison tool, *open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Diff** > **Directories Comparison** > **Appearance**.

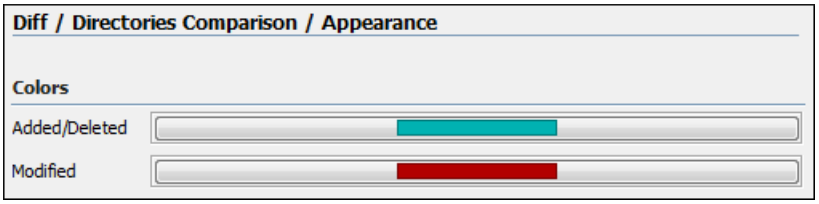


Figure 43: Diff Appearance Preferences Panel

- **Added/Deleted** - Color used for marking added or deleted files and folders.
- **Modified** - Color used for marking modified files.

Archive Preferences

To configure *Archive* options, *open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Archive**.

The following options are available in the **Archive** preferences page:

Archive backup options

Controls if the application makes backup copies of the modified archives. The following options are available:

- **Always create backup copies of modified archives** - When you modify an archive, its content is backed up.
- **Never create backup copies of modified archives** - No backup copy is created.
- **Ask for each archive once per session** - Once per application session for each modified archive, user confirmation is required to create the backup. This is the default setting.

  Note: Backup files have the name `originalArchiveFileName.bak` and are located in the same folder as the original archive.

Archive types

This table contains all known archive extensions mapped to known archive formats. Each row maps a list of extensions to an archive type supported in Oxygen XML Editor. You can use the **Edit** button at the bottom of the table to edit an existing mapping or the **New** button to create a new one and associate your own list of extensions to an archive format.

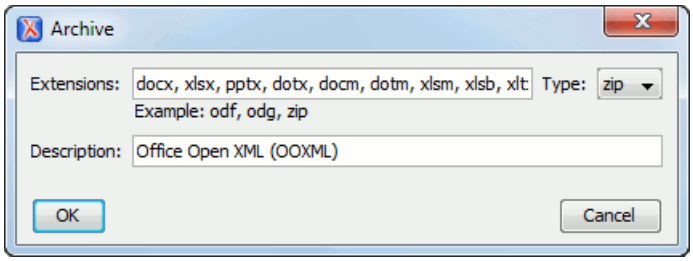


Figure 44: Edit Archive Extension Mappings

Important: You have to restart Oxygen XML Editor after removing an extension from the table for that extension to not be recognized as an archive extension.

Store Unicode file names in Zip archives

Use this option when you archive files that contain international (non-English) characters in file names or file comments. If this option is selected and an archive is modified in any way, UTF-8 characters are used in the names of all files in the archive.

Plugins Preferences

You can add *plugins* that extend the functionality of Oxygen XML Editor. The *plugins* are shipped as separate packages. To check for new *plugins*, go to *http://www.oxygenxml.com/oxygen_sdk.html*.

A *plugin* consists of a separate sub-folder in the `Plugins` folder of the Oxygen XML Editor installation folder. This sub-folder must contain a valid `plugin.xml` file in accordance with the `plugin.dtd` file located in the `Plugins` folder.

Oxygen XML Editor automatically detects and loads *plugins* installed correctly in the `Plugins` folder and displays them in the **Plugins** preferences page. To configure *plugins*, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Plugins**.

You can use the checkboxes in front of each *plugin* to enable or disable them. To display the properties of a *plugin* in the second section of the **Plugins** preferences page, click the name of the *plugin*.

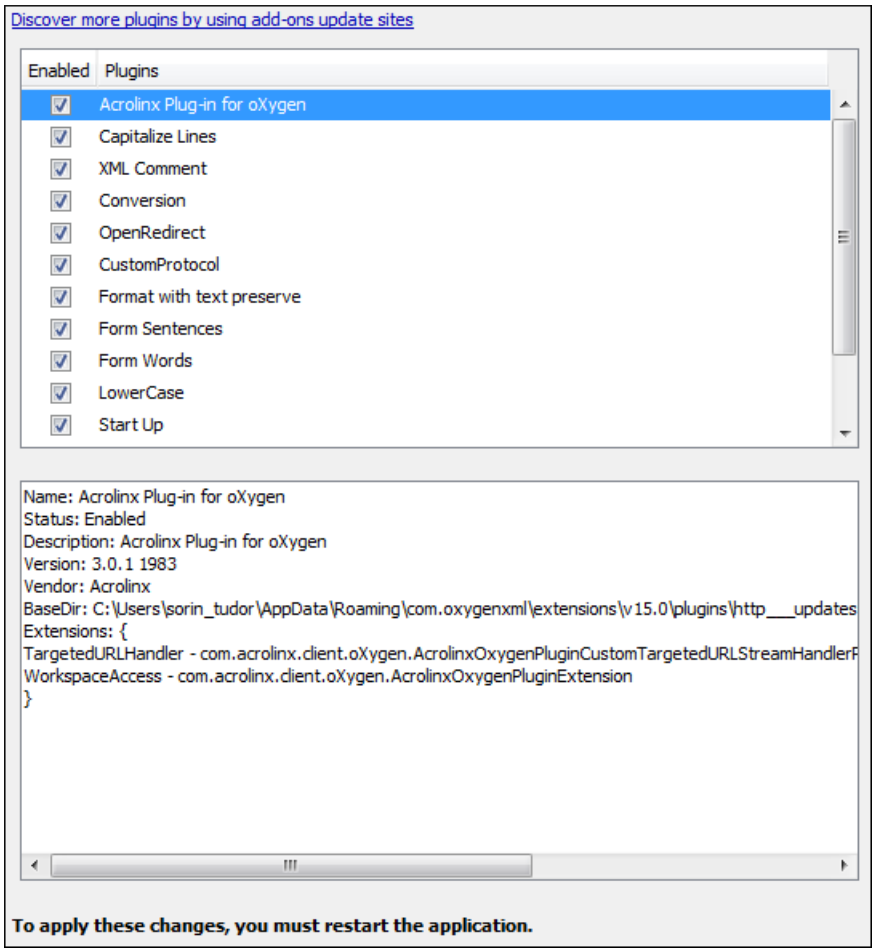


Figure 45: Plugins Preferences Panel

Also, you can install a *plugin* as an add-on. For further details about this, go to *Deploying Add-ons*

External Tools Preferences

A command-line tool can be started in the Oxygen XML Editor user interface as if from the command line of the operating system shell. The **External Tools** preferences page allows you to add and configure these external tools that could be used while working with Oxygen XML Editor. To access this preferences page, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **External Tools** (or select **Configure** from the **Tools** > **External Tools** menu).

This preferences page presents a list of the external tools that have been configured. You can use the buttons at the bottom of the page to configure the items in the list. Once a tool has been configured, you can open it by selecting it from the **Tools** > **External Tools** menu or from the **External Tools** drop-down menu on the toolbar (the **Tools** toolbar needs to be selected in the ***Configure Toolbars** dialog box*).

How to Configure an External Tool

To configure an external tool in the **External Tools** preferences page, use any of the following buttons at the bottom of the page:

- **New** - Adds a new external tool to the list.
- **Edit** - Allows you to configure an existing external tool, selected from the list.
- **Duplicate** - Duplicates an existing external tool, selected from the list, to use as a template for configuring a similar tool.

Any of those three buttons opens the **External Tools** configuration dialog box.

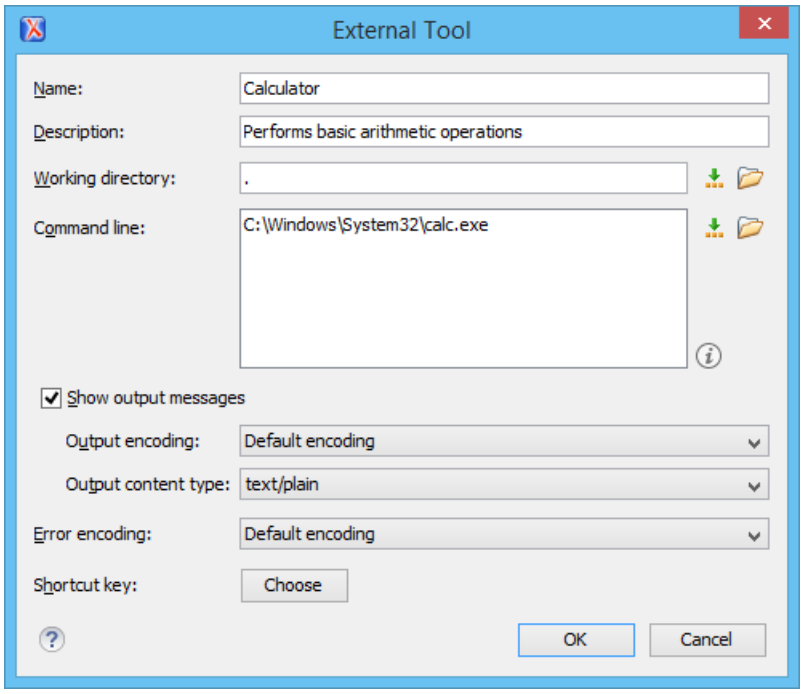


Figure 46: External Tools Configuration Dialog Box

This configuration dialog box includes the following options:

Name

The name of tool that will be displayed in the **Tools** > **External Tools** menu and in the **External Tools** drop-down menu on toolbar.

Description

A description of the tool displayed as a tooltip where the tool name is used.

Working directory

The directory that the external tool will use to store intermediate and final results. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button. You can use one of the following editor variables: *${cfd}*, *${pd}*, *${oxygenInstallDir}*, *${homeDir}*, *${system(var.name)}*, *${date(pattern)}*, *${xpath_eval(expression)}*. You can also use the browsing tools to select the directory.

Command line

The command line that will start the external tool. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button. You can use one of the following editor variables: *${homeDir}*, *${home}*, *${cfn}*, *${cfne}*, *${cf}*, *${currentFileURL}*, *${cfd}*, *${cfdu}*, *${tsf}*, *${pd}*, *${pdu}*, *${oxygenInstallDir}*, *${oxygenHome}*, *${frameworksDir}*, *${frameworks}*, *${ps}*, *${timeStamp}*, *${uuid}*, *${id}*, *${afn}*, *${afne}*, *${af}*, *${afu}*, *${afd}*, *${afdu}*, *${ask('message', type, 'default_value')}*, *${dbgXML}*, *${dbgXSL}*, *${env(VAR_NAME)}*, *${system(var.name)}*, *${date(pattern)}*, and *${xpath_eval(expression)}*. You can also use the browsing tools to select a file path.

Show output messages

When this option is selected, all the messages emitted by the external tool are displayed in the ***Results*** *view*. When this option is not selected, only the error messages are displayed. You can also choose the output encoding and content type:

- **Output encoding** - The encoding of the output stream of the external tool that will be used byOxygen XML Editor to read the output of the tool.
- **Output content type** - A list of predefined content type formats that instructOxygen XML Editor how to display the generated output. For example, setting the **Output content type** to `text/xml` enables the syntax coloring of XML output.

Error Encoding

The encoding of the error stream of the external tool that will be used by Oxygen XML Editor to read the error stream.

Shortcut key

You can choose a keyboard shortcut that can be used to launch the external tool.

Menu Shortcut Keys Preferences

You can use the **Menu Shortcut Keys** preferences page to configure shortcut keys for the actions available in Oxygen XML Editor. The shortcuts assigned to actions are displayed in a table in this preference page. To access the full list of shortcut keys, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Menu Shortcut Keys** (or simply go to **Options** > **Menu Shortcut Keys**).

For a list of the most commonly used shortcuts, see *Frequently Used Shortcut Keys* on page 18.

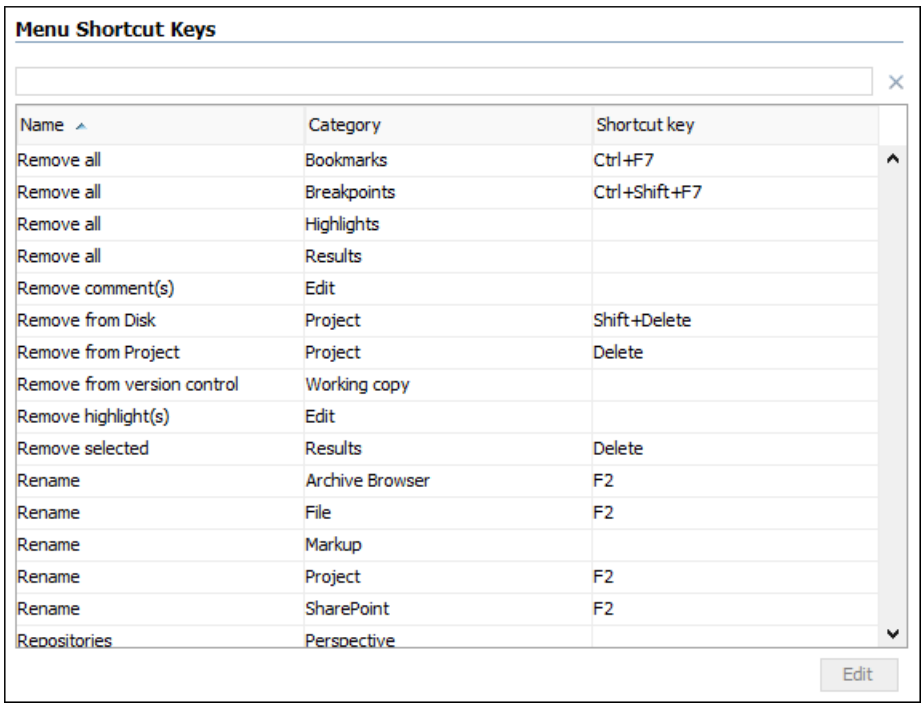


Figure 47: Menu Shortcut Keys Preferences Page

The **Menu Shortcut Keys** preferences page also contains the *shortcuts that you define at document type level*.

Note: A shortcut defined at *document type* level overwrites a default shortcut.

To find a specific action, you can use the filter text field to search through any of the columns in the table. You can also press shortcut key combinations on your keyboard to filter the list and click on a column header to sort that column.

The table includes the following columns or options:

- **Description** - A short description of the action.
- **Category** - A classification of the actions in categories for easier management and more flexibility in assigning multiple keys for the same action.
- **Shortcut key** - The combination of keyboard keys that can be used to launch the action. To add or change a shortcut key, you can either double-click a row or select the row and press the **Edit** button.
- **'Home' and 'End' keys are applied at line level** (available on Mac OS X only) - Controls the way the HOME and END keys are interpreted. If selected, the default behavior of these keys is overridden and the cursor only moves on the current line.

How to Assign a Shortcut Key or Edit an Existing Shortcut

To assign a shortcut key to an action or edit an existing shortcut configuration, follow these steps:

1. Select the action in the table.
2. Click the **Edit** button.

   Step Result: The **Shortcut key** configuration dialog box is displayed.

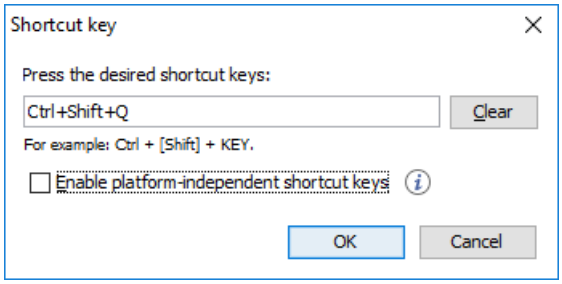


   Figure 48: Shortcut Key Configuration Dialog Box

3. Press the desired shortcut keys on your keyboard.
4. If you need the shortcut to work on multiple platforms, select the **Enable platform-independent shortcut keys** option. In this case, the following modifiers are used:
   - `M1` represents the **Command** key on MacOS X, and the **Ctrl** key on other platforms.
   - `M2` represents the **Shift** key.
   - `M3` represents the **Option** key on MacOS X, and the **Alt** key on other platforms.
   - `M4` represents the **Ctrl** key on MacOS X, and is undefined on other platforms.
5. Click **OK** to save your configuration.

Related Information:

Frequently Used Shortcut Keys on page 18

File Types Preferences

Oxygen XML Editor offers built-in editing support for a wide variety of file types, but you can also add new file extensions and associate them with whatever editor type fits your needs. The associations set here between a file extension and the type of editor will determine which editor will be opened for editing purposes when that type of file is created or opened.

To configure the **File Types** options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **File Types**.

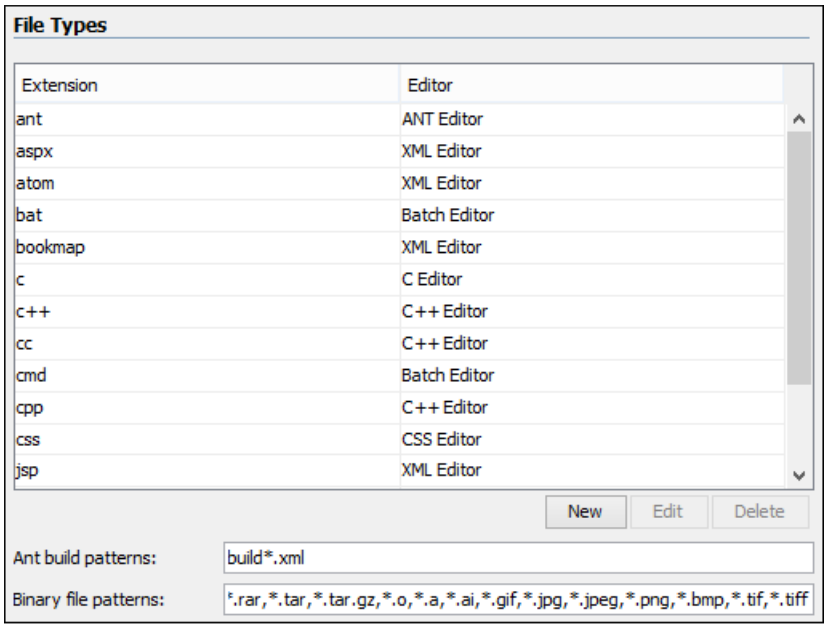


Figure 49: File Types Preferences Page

The table contains the following columns:

- **Extension** - The extensions of the files that will be associated with an editor type.
- **Editor** - The type of editor which the extensions will be associated with. Some editors provide easy access to frequent operations via toolbars (XML editor, XSL editor, DTD editor) while others provide just a syntax highlight scheme (Java editor, SQL editor, Shell editor, etc.).

  If the editor set here is not one of the XML editors (XML editor, XSL editor, XSD editor, RNG editor, WSDL editor) then the encoding set in the ***Encoding for non XML files*** *option* is used for opening and saving a file of this type.

The files that match the **Ant build patterns** will be associated with the Ant editor.

The files that match the **Binary file patterns** patterns are handled as binary and opened in the associated system application. Also, they are excluded from the following actions available in the ***Project*** *view*: **File/Replace in Files**, **Check Spelling in Files**, **Validate**.

Open/Find Resource Preferences Page

You can configure various options that pertain to the ***Open/Find Resource*** *dialog box* and ***Open/Find Resource*** *view*. To access these options, *open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Open/Find Resource**.

The following options are available in this **Open/Find Resource** preferences page:

Limit search results to

Specifies the maximum number of results that are displayed in the ***Open/Find Resource*** *dialog box/view*.

Enable searching in content

This option is selected by default and it allows you to use the ***Open/Find Resource*** *dialog box/view* to search in content or reviews, as well as in file paths. If this option is not selected, you can only use the **Open/Find Resource** feature to search in file paths.

Content search scope section

Ignore content of these files

Allows you to select specific directories, files, or file types that are ignored when you perform a search. For example, `*.txt` ignores all the `.txt` files, `*/topics/*` ignores all the files from the `topics` directory, regardless of their depth, and `file:/C:/tmp/*` ignores everything from the `tmp` directory.

Note: The specified pattern must begin with the desired protocol (in our case *file*) and also contain forward slash (/).

Index the content of remote resources

Controls the indexing of resources that are not local. For example, the resources referenced in a *DITA map* opened from a remote server (from a CMS or from a WebDAV location) are not indexed by default. To index the content of these resources, select this option.

Note: Selecting this option may lead to delays when the indexing is computed.

Content search options section

Content language

Use this option to specify a language for the search engine to use for the current document. This is helpful if you have multiple languages within the content of a document. The search engine will use a set of *stop words* and analyzers tuned specifically for that specific language. By default, it is mapped to the *UI language specified in the **Global** preferences page*. Therefore, you need to change this option only if the language of the text you want to perform the search in differs from the UI language.

Tip: If you select **<Generic language (no stemming)>** from the drop-down list, no word stemming is performed when creating the index. This might be useful if your content has many technical terms that should be indexed as they are.

Stop words

A list of *stop words* that will be filtered out of the search processing. The list is automatically populated based upon the specified **Content language**, but you can add or remove words from the list.

When searching in content, return

This option specifies how matches are returned when doing searches in content. You can choose between two options:

- **Exact matches** - The search results match the exact whole words that you enter in the search field of the **Open/Find Resource** dialog box/view.
- **Prefix matches** (default) - The search results match documents that contain words starting with the search terms. For instance, searching for "pref page" will also find documents containing "preference page".

Automatically join search terms using:

Allows you to select the default boolean operator that Oxygen XML Editor applies when you perform a search. For example, if the AND operator is selected and you search for "car assembly", the matches must contain both of the words. If you choose OR, the matches must contain one of the selected search terms and results that contain both words are promoted to the top of the list.

Enable XML-aware searching

When selected, you can perform *XML-specific searches* for XML elements and attributes.

Note: Selecting this option may slow down the indexing of your documents and increase the index size on the disk.

Index files with size less than (KB)

Since indexing can be slowed down when the ***Enable XML-aware searching** option* is active, you can use this option to set a maximum file size to be indexed.

Stop Words

A list of comma separated *stop words*, meaning that the words added in this list are filtered out prior to processing a search query.

Related Information:

Open/Find Resource View on page 194

Open/Find Resource Dialog Box on page 271

Custom Editor Variables Preferences

An *editor variable* is useful for making a transformation scenario, validation scenario, or other tool independent of its file path. An editor variable is specified as a parameter in a transformation scenario, validation scenario, or command line of an external tool. Such a variable is defined by a name, a string value, and a text description. A custom editor variable is defined by the user and can be used in the same expressions as the *built-in editor variables*.

Custom editor variables are created and configured in the **Custom Editor Variables** preferences page. To access this page, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Custom Editor Variables**.

This preferences page displays a table of all the custom editor variables that have been defined. The table includes three columns for the editor variable **Name**, its **Value**, and its **Description**. The create a new variable, click the **New** button at the bottom of the table and define your custom editor variable in the subsequent dialog box. To edit an existing custom editor variable, click the **Edit** button and configure the variable in the subsequent dialog box. You can also use the **Delete** button to remove custom editor variables that are no longer needed.

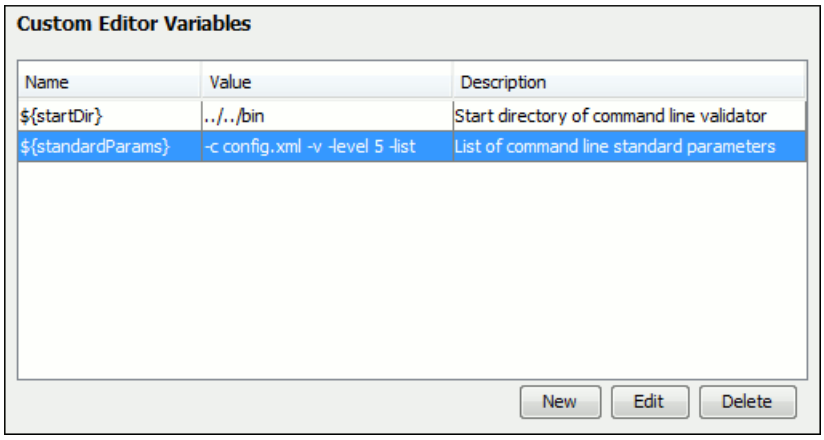


Figure 50: Custom Editor Variables Table

Network Connection Settings Preferences

This section presents the options available in the **Network Connection Settings** preferences pages.

Proxy Preferences

Some networks use proxy servers to provide internet services to LAN clients. Therefore, clients behind the proxy may only connect to the Internet via the proxy service. If you are not sure if your computer is required to use a proxy server to connect to the Internet or you do not know the proxy parameters, consult your network administrator.

To configure the **Proxy** options, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Network Connection Settings** > **Proxy**. The following options are available:

Proxy section

Specifies how HTTP(S) connections go through the proxy server. You can choose between the following three options:

- **Direct connection** - HTTP(S) connections will go directly to the target host without going through a proxy server.
- **Use system settings** (default setting) - HTTP(S) connections will go through the proxy server set in the operating system.

  Attention: The system settings for the proxy cannot be read correctly from the operating system on some Linux systems. The system settings option should work properly on Gnome based Linux systems, but it does not work on KDE based ones as the Java virtual machine does not offer the necessary support yet.

- **Manual proxy configuration** - HTTP(S) connections will go through the proxy server specified in the **Web Proxy (HTTP/HTTPS)** section.

Web Proxy (HTTP/HTTPS) section

Address

The address of the proxy server used for manual configurations.

Port

The port of the proxy server used for manual configurations.

No proxy for

Specifies the hosts that the connections must not go through a proxy server. A host needs to be written as a fully qualified domain name (for example, `myhost.example.com`) or as a domain name (for example, `example.com`). Use a comma to separate multiple hosts.

User

The user name for authentication with the proxy server.

Password

The password for authentication with the proxy server.

SOCKS Proxy section

Address

The address of a SOCKS proxy that all connections will pass through. If this field is empty, the connections do not use a SOCKS proxy.

Port

The port of a SOCKS proxy that all connections will pass through.

Using a Proxy Auto-Configuration Script (PAC)

If you have set up the path to a Proxy auto-configuration script in your system, Oxygen XML Editor cannot detect this setting out of the box.

You can create a new folder (*[OXYGEN_INSTALL_DIR]*`\lib\endorsed`) in which you should copy two additional Java libraries: `deploy.jar` and `plugin.jar`. These libraries can be found in the *[OXYGEN_INSTALL_DIR]*`\jre\lib` folder if the application came with a bundled Java VM (otherwise, in the Java VM installation used to run the application).

HTTP(S)/WebDAV Preferences

To set the **HTTP(S)/WebDAV** preferences, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Network Connection Settings** > **HTTP(S)/WebDAV**. The following options are available:

Internal Apache HttpClient Version

Oxygen XML Editor uses the Apache HttpClient to establish connections to HTTP servers. For Oxygen XML Editor to benefit from particular sets of features provided by different versions, you may choose between v3 and v4.

Note: For a full list of features, go to *http://hc.apache.org/httpclient-3.x/* and *http://hc.apache.org/httpcomponents-client-ga/*.

Maximum number of simultaneous connections per host

Defines the maximum number of simultaneous connections established by the application with a distinct host. Servers might consider multiple connections opened from the same source to be a **Denial of Service** attack. You can avoid that by lowering the value of this option.

Note: The minimum value that can be set in this option is 5.

Read Timeout (seconds)

The period (in seconds) after which the application considers that an HTTP server is unreachable if it does not receive any response from that server.

Enable HTTP 'Expect: 100-continue ' handshake (for HTTP/1.1 protocol)

Activates *Expect: 100-Continue* handshake. The purpose of the *Expect: 100-Continue* handshake is to allow a client that is sending a request message with a request body to determine if the origin server is willing to accept the request (based on the request headers) before the client sends the request body. The use of the *Expect: 100-continue* handshake can result in noticeable performance improvement when working with databases. The *Expect: 100-continue* handshake should be used with caution, as it may cause problems with HTTP servers and proxies that do not support the HTTP/1.1 protocol.

Use the 'Content-Type' header field to determine the content type

When selected, Oxygen XML Editor tries to determine a resource type using the **Content-Type** header field. This header indicates the *Internet media type* of the message content, consisting of a type and subtype. For example:

```
Content-Type: text/xml
```

When unchecked, the resource type is determined by analyzing its extension. For example, a file ending in `.xml` is considered to be an XML file.

Automatic retry on recoverable error

When selected, if an HTTP error occurs when Oxygen XML Editor communicates with a server via HTTP (for example, sending or receiving a SOAP request to or from a Web services server) and the error is recoverable, Oxygen XML Editor tries to re-send the request to the server.

Automatically accept a security certificate, even if invalid

When selected, the HTTPS connections that Oxygen XML Editor attempts to establish with will accept all security certificates, even if they are invalid.

Important: By accepting an invalid certificate, you accept (at your own risk) a potential security threat, since you cannot verify the integrity of the certificate's issuer.

Lock WebDAV files on open

If selected, the files opened through WebDAV are locked on the server so that they cannot be edited by other users while the lock placed by the current user still exists on the server.

(S)FTP Preferences

To configure the **(S)FTP** options, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Network Connection Settings** > **(S)FTP**. You can customize the following options:

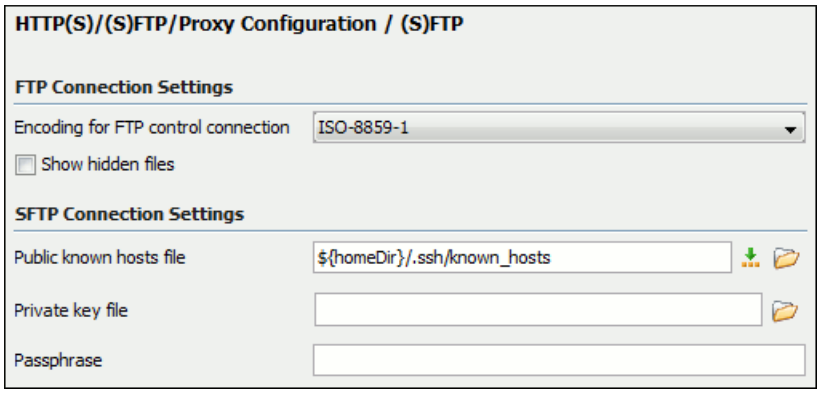


Figure 51: (S)FTP Configuration Preferences Panel

- **Encoding for FTP control connection** - The encoding used to communicate with FTP servers: either ISO-8859-1 or UTF-8. If the server supports the UTF-8 encoding Oxygen XML Editor will use it for communication. Otherwise, it will use ISO-8859-1.
- **Public known hosts file** - File containing the list of all SSH server host keys that you have determined are accurate. The default value is `${homeDir}/.ssh/known_hosts`.
- **Private key file** - The path to the file containing the private key used for the private key method of authentication of the secure FTP (SFTP) protocol. The user / password method of authentication has precedence if it is used in *the Open URL dialog box*.

- **Passphrase** - The passphrase used for the private key method of authentication of the secure FTP (SFTP) protocol. The user / password method of authentication has precedence if it is used in *the Open URL dialog box*.

Trusted Hosts Preferences

This preferences page contains a list of domains that have been identified as trusted. You can add or remove domains from the list and Oxygen XML Editor will allow connections to the listed hosts without requesting user confirmation.

To configure the **Trusted Hosts** options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Network Connection Settings** > **Trusted Hosts**. The following options are available:

- **New** - Allows you to manually add a new entry to the list of trusted hosts.

  Tip: You can specify a specific port at the end of the URL (for instance, `www.example.com:8080`). Otherwise, if no port is specified,connections will be allowed on all ports for the particular host.
- **Delete** - Allows you to remove an entry from the list of trusted hosts.

SSH Preferences

To configure the **SSH** options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Connection settings** > **SSH**. The following options are available:

- **SSH** - Specifies the command line for an external SSH client that will be used when connecting to a SVN +SSH repository. Absolute paths are recommended for the SSH client executable and the file paths given as arguments (if any). Depending on the SSH client used and your SSH server configuration, you may need to specify the user name and/or private key/passphrase in the command line . You can also choose whether to use the **Default SVN user** (the same user name as the SSH client user) or **Prompt for a SVN user** for SVN repository operations whenever SVN authentication is required. For example, on Windows the following command line uses the `plink.exe` tool as the external SSH client for connecting to the SVN repository with SVN+SSH:

  ```
  C:\plink-install-folder\plink.exe -l username -pw password -ssh -batch
  host_name_or_IP_address_of_SVN_server
  ```

XML Structure Outline Preferences

To configure options in regards to the ***Outline** view*, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML Structure Outline**. It contains the following options:

Preferred attribute names for display

The preferred attribute names when displaying the attributes of an element in the **Outline** view. If there is no preferred attribute name specified, the first attribute of an element is displayed.

Enable outline drag and drop

Drag and drop is disabled for the tree displayed in the **Outline** view only if there is a possibility to accidentally change the structure of the document by such operations.

Views Preferences

The **Views** preferences page allows you to configure some options in regards to certain views. To edit these options, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Views**.

The following options are available:

Project view section

Enable drag-and-drop in Project view

Enables drag and drop support in the ***Project** view*. It should be disabled only if there is a possibility of accidentally changing the structure of the project by drag and drop actions.

Information view section

Maximum number of lines

Specifies the maximum number of lines that can be written in the ***Information*** *view*.

Elements view section

Show only items allowed at cursor position

If selected, when editing in **Author** mode, only the elements that are allowed at the current cursor position will be listed in the ***Elements*** *view*. If not selected, two additional tabs (**Before** and **After**) will be displayed at the bottom of the view (in **Author** mode only). These tabs list the elements that are allowed before or after the element at the current cursor position.

Messages Preferences

The **Messages** preference page allows you to specify whether or not certain messages are displayed. To configure these options, *open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Messages**.

The following warning messages can be enabled or disabled:

Show confirmation dialog when moving resources

Specifies whether or not to display a confirmation dialog box when you move a resource in the ***Project*** *view*, ***Data Source Explorer*** *view*, and ***Archive Browser***. In the confirmation dialog box, there is an option to choose to not show this dialog box in the future. To reset that behavior, simply select **Restore Defaults** at the bottom of this preferences page.

Show warning when adding resources already included in the project

Specifies whether or not to display a dialog box that warns you if you try to add files that already exist in your project.

Show warning for document size limit for bidirectional text, Asian languages, and other special characters

Specifies whether or not to display a warning message when an opened file that contains bidirectional characters is too large and bidirectional support is disabled.

Show warning message when changing the text orientation in the editor

Specifies whether or not to display a warning message when you change the text orientation in the editor.

Show warning when editing long expressions in the XPath toolbar

Specifies whether or not to display an information dialog box that allows you to specify if you want to use the ***XPath/XQuery Builder*** view when editing long XPath expressions.

Show MathFlow recommendation

Specifies whether or not to display an information dialog box that recommends using the ***MathFlow Editor*** to edit MathML equations.

Show SFTP certificate warning dialog

Specifies whether or not to display a warning dialog box each time the authenticity of the SFTP server host cannot be established.

Show Enterprise license related message when trying to connect to a Microsoft SharePoint server

Specifies whether or not to display an error message if you try to connect to a Microsoft SharePoint server without having the proper license.

Convert DB Structure to XML Schema

When tables from a database schema are selected in the **Select database table** section of the ***Convert DB Structure to XML Schema*** *dialog box* and another database schema is expanded, a confirmation is needed since the previous selection will be discarded. This option specifies whether or not you are always asked if you want the other database schema to always be expanded without asking you, or it is never expanded.

Show the dialog box for choosing the encoding for Base64, Base 32, Hex conversions

Specifies whether or not to display a dialog box that allows you to choose a specific encoding whenever you use the **Encode Selection** or **Decode Selection** actions for *Base64*, *Base32*, or *Hex conversions*. In the dialog box, there is an option to choose to not show this dialog box in the future. To reset that behavior, simply select **Restore Defaults** at the bottom of this preferences page.

Configuring Options

A set of options controls the behavior of Oxygen XML Editor, allowing you to configure most of the features. To offer you the highest degree of flexibility in customizing the application to fit the needs of your organization, Oxygen XML Editor includes several distinct layers of option values.

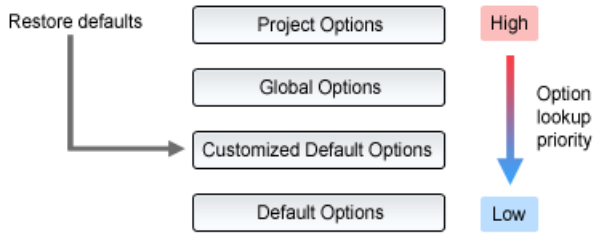


Figure 52: Option Lookup Priority

The option layers are as follows (sorted from high priority to low):

- ***Project Options***

  Allows project managers to establish a set of rules for a specific project. These rules standardize the information exchanged by the team members (for example, if the project is stored in a repository, a common set of formatting rules avoid conflicts that may appear when documents modified by various team members are committed to the repository).
- *Global Options*

  Allows individual users to personalize Oxygen XML Editor according to their specific needs.
- ***Customized Default Options***

  Designed to customize the initial option values for a group of users, this layer allows an administrator to deploy the application preconfigured with a standardized set of option values.

  Note: Once this layer is set, it represents the initial state of Oxygen XML Editor when an end-user uses the ***Restore defaults*** or ***Reset Global Options*** actions.
- **Default Options**

  The predefined default or built-in values, tuned so that Oxygen XML Editor behaves optimally in most working environments.

Important: If you set a specific option in one of the layers, but it is not applied in the application, make sure that one of the higher priority layers does not overwrite it.

Customizing Default Options

Oxygen XML Editor has an extensive set of options that you can configure. When Oxygen XML Editor in installed, these options are set to default values. You can provide a different set of default values for an installation using an XML *options file*.

Creating an XML *Options File*

To create an *options file*, follow these steps:

1. It is recommended that you use a fresh install for this procedure, to make sure that you do not copy personal or local preferences.
2. Open Oxygen XML Editor and *open the **Preferences** dialog box **(Options > Preferences)***.
3. Go through the options and set them to the desired defaults. Make sure that ***Global Options*** is selected in each page.

4. Click **OK** and close the **Preferences** dialog box.
5. Go to **Options** > **Export Global Options** to create an XML options file.

Using Customized Default Options

There are two methods that you can use to configure an Oxygen XML Editor installation to use the customized default options from the created XML options file:

- **Copy the XML Options File to the Installation Directory**

  In the *[OXYGEN_INSTALL_DIR]*, create a folder called `preferences` and copy the created XML options file into it (for example, *[OXYGEN_INSTALL_DIR]*`/preferences/default.xml`).
- **Specify a Path to the XML Options File in a Startup Parameter**

  Set the path to the XML options file as the value of the `com.oxygenxml.default.options` system property in the *startup parameters*. The path can be specified with any of the following:

  - A URL or file path relative to the application installation folder. For example:

    ```
    -Dcom.oxygenxml.default.options=options/default.xml
    ```
  - A system variable that specifies the file path. For example:

    ```
    com.oxygenxml.default.options=${system(CONFIG)}/default.xml
    ```
  - An environmental variable that specifies the file path. For example:

    ```
    com.oxygenxml.default.options=${env(CONFIG)}/default.xml
    ```

  Note: If you are using the *Java Webstart distribution*, use the *`optionsDir` property* to specify the path of the options file (in this case, the file must be named `default.xml`), or you can edit the *`.jnlp` file* that launches the application and set the `com.oxygenxml.default.options` parameter using a `property` element, as in the following example:

  ```
  <property name="oxy:com.oxygenxml.default.options"
            value="http://host/path/to/default.xml"/>
  ```

Storing Global and Project Level Options

When you configure the Oxygen XML Editor options, you can store them globally or bind them to a specific project by choosing the appropriate setting in the preferences pages. They can then be *shared with others by exporting the global options* or by *sharing the stored project-level files*. The same is true with transformation and validation scenarios.

For each preferences page, you can choose between ***Global Options*** and ***Project Options*** depending upon how you want to store the options in that particular preferences page.

Notice: Some pages do not have the **Project Options** button, since the options they host might contain sensitive data (passwords, for example), unsuitable for sharing with other users.

If changes have been made to the options in a preferences page and you switch between **Project Options** and **Global Options**, a dialog box will be displayed that allows you to select one of the following:

- **Overwrite** - The existing options from the current preferences page will be overwritten.
- **Preserve** - The existing options from the current preferences page will be preserved.

Figure 53: Controlling the Storage Options for the Preferences

Global Options

By default, **Global Options** is selected in the preferences pages, meaning that the options are stored locally on your computer and are not accessible to other users (unless you *export them into an XML options file that can then be shared*).

Global options are stored locally in option files (for example, `oxyOptionsSa18.1.xml` for a standalone distribution of Oxygen XML Editor version 18.1) located in the following directories:

- **Windows (Vista, 7, 8, 10)** - `[user_home_directory]\AppData\Roaming\com.oxygenxml`
- **Windows XP** - `[user_home_directory]\Application Data\com.oxygenxml`
- **Mac OS X** - `[user_home_directory]/Library/Preferences/com.oxygenxml`
- **Linux/Unix** - `[user_home_directory]/.com.oxygenxml`

Project Options

If you select **Project Options**, the preferences are stored in the project file (`.xpr`), which can easily be *shared with other users*.

Notice: Some pages do not have the **Project Options** button, since the options they host might contain sensitive data (passwords, for example), unsuitable for sharing with other users.

Related Information:

Sharing Application Settings on page 163
Customizing Default Options on page 161
Importing/Exporting/Resetting Global Options on page 164

Sharing Application Settings

There are a variety of ways that you can share the settings in Oxygen XML Editor with other members of your team so that you all use a common set of options. This topic describes various possibilities.

Share Settings Through a Project File

Most of the preference pages in Oxygen XML Editor include a ***Project Options*** button that allows you to pass changes to the settings to the current project file that is opened in the ***Project*** *view*. That project file can then be shared with other users. For instance, if your project file is saved on a version control system (such as SVN, CVS, or Source Safe) or in a shared folder, your team will have access to the same option configuration that you stored in the project file.

For more information about sharing projects, see *Sharing a Project - Team Collaboration* on page 284.

Share Settings by Exporting/Importing Global Options

Oxygen XML Editor includes actions in the **Options** menu that allow you to export and import the *global settings*. The **Export Global Options** action will save the global settings as an XML properties file. You can then share those settings with others by using the **Import Global Options** action to import that properties file on their computer.

For more information about global options, see *Importing/Exporting/Resetting Global Options* on page 164.

Share Settings with a Custom Options File During Installation

When Oxygen XML Editor in installed, all the settings are set to default values. You can customize the set of default values by creating an XML *options file* that you will use when installing Oxygen XML Editor on each computer. You can then copy the XML options file to the installation directory or specify its path in a startup parameter.

For more information about creating and referencing a custom options file, see *Customizing Default Options* on page 161.

Share Settings by Imposing Fixed Options with an API

The Maven-based *Oxygen XML SDK* includes a sample *plugin* called **ImposeOptions** that imposes a fixed set of options when the application starts. This can be achieved by using the `PluginWorkspaceProvider.getPluginWorkspace().setGlobalObjectProperty(key, value)` API method.

For more information about this API, see *PluginWorkspaceProvider Class*.

Related Information:

Sharing a Project - Team Collaboration on page 284
Sharing Transformation Scenarios on page 929
Sharing Validation Scenarios on page 460
Customizing Default Options on page 161
Importing/Exporting/Resetting Global Options on page 164
Sharing a Framework (Document Type) on page 1239

Importing/Exporting/Resetting Global Options

Actions for importing, exporting, and resetting global options are available in the **Options** menu. The export operation allow you to save *global preferences* as an XML properties file and the import operation allows you to load the property file. You can use this file to reload saved options on your computer or to *share with others*.

The following actions are available in the **Options** menu:

Reset Global Options

Restores the preference to the factory defaults or to *customized defaults*. This action also resets the transformation and validation scenarios to the default scenarios and clears recently used file templates.

Import Global Options

Allows you to import a set of *Global Options* from an exported XML properties file. You can also select a *project-level options file* (`.xpr`) to import all the *Global Options* that are set in that project file. After you select a file, the **Import Global Options** dialog box is displayed, and it informs you that the operation will only override the options that are included in the imported file. You can select the **Reset all other options to their default values** option to reset all options to the default values before the file is imported.

Export Global Options

Allows you to export *Global Options* to an XML properties file. Some user-specific options that are private are not included. For example, passwords and the name of the *Review Author* is not included in the export operation.

Oxygen XML Editor automatically stores your global options in an XML properties file. Depending on the platform you are using, this file is located in the following directories:

- `[user-home-folder]\AppData\Roaming\com.oxygenxml` for Windows Vista/7/8/10
- `[user-home-folder]\Application Data\com.oxygenxml` for Windows XP
- `[user-home-folder]/Library/Preferences/com.oxygenxml` for OS X
- `[user-home-folder]/.com.oxygenxml` for Linux

The name of the `options` file of Oxygen XML Editor 19.0 is oxyOptionsSa19.0.xml.

Associating a File Extension with Oxygen XML Editor

To associate a file extension with Oxygen XML Editor on Windows:

1. Go to the Windows **Start** menu and open **Control Panel**.
2. Go to **Default Programs**.
3. Click **Associate a file type or protocol with a program**.
4. Click the file extension you want to associate with Oxygen XML Editor, then click the **Change program** button.
5. In the subsequent dialog box, browse for and choose Oxygen XML Editor.
6. Click **OK**.

To associate a file extension with Oxygen XML Editor on Mac OS:

1. In **Finder**, select a file and from the contextual menu select **Get Info**.
2. In the **Open With** subsection, select **Other** from the application combo box.
3. Browse to and select Oxygen XML Editor.
4. Select the **Always Open With** option, then click **Add**.

5. If you want all files of that type to be opened in Oxygen XML Editor, click the **Change All** button and then the **Continue** button in the subsequent confirmation dialog box.

Configuring the Layout of the Views and Editors

All the Oxygen XML Editor views available in the ***Editor***, **XSLT Debugger**, and ***XQuery Debugger*** *perspectives* are *dockable*. To open a view, select it from the **Window** > **Show View** menu. You can hide a view by closing it with the ✕ button at the top-right corner of the view, or with the **Window** > **Hide current view** action.

Arranging the Layout

You can drag any view to any margin of another view or editor inside the Oxygen XML Editor window. Once you create a layout that suites your needs, you can save it from **Window** > **Export Layout**. Oxygen XML Editor creates a layout file containing the preferences of the saved layout. To load a layout, go to **Window** > **Load Layout**. To reset it, select **Window** > **Reset Layout**.

Note: The **Load Layout** menu lets you select between the default layout, a predefined layout, or a custom layout. The changes you make using the **Load Layout** menu are also reflected in the **Application Layout** preferences page.

The changes you make to any layout are preserved between working sessions. The predefined layout files are saved in the *preferences directory* of Oxygen XML Editor.

To gain more editing space in the Oxygen XML Editor window, click 📌 **Toggle auto-hide** in any view. This button sets the view in the *auto-hide* state, making it visible only as a vertical tab, at the margins of the Oxygen XML Editor window. To display a view in the *auto-hide* state, hover its side-tab with your cursor, or click it to keep the view visible until you click elsewhere. A view can also be set to a floating state by using the **Toggle floating** action, making it independent from the rest of the Oxygen XML Editor window.

You can drag the editors and arrange them in any order, both horizontally and vertically.

The following image presents two editors arranged as horizontal tiles. To arrange them vertically, drag one of them on top of the other. In this example, the `personal.xml` file was dragged over the `personal-schema.xml` file. When doing this, a dark gray rectangle marks the rearranged layout.

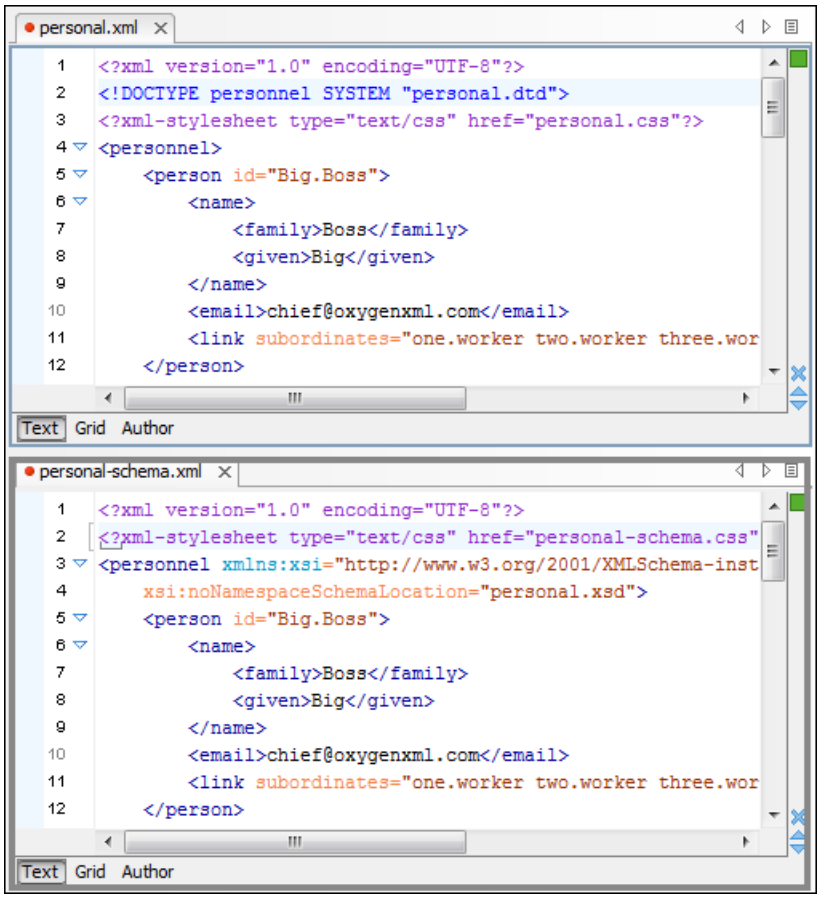


Figure 54: Drag and Drop Editors

Tile/Stack Editor Actions

You can also tile or stack all open editors, both in the *Editor perspective* or in the *Database perspective*, using the following actions from the **Editor** toolbar or **Window** menu:

Tile Editors Horizontally

Splits the editing area into horizontal tiles, one for each open file.

Tile Editors Vertically

Splits the editing area into vertical tiles, one for each open file.

Stack Editors

The reverse of the **Tile Editors Horizontally/Vertically** actions. Stacks all open editors.

Synchronous Scrolling

Select this action to scroll through the tiled editors at the same time.

Note: When tiled, you can still drag and drop the editors, but note that they are docked in the same way as a window/view (instead of just tabs). You are actually rearranging the editor windows, so drag the editor tab and drop it to one of the sides of an editor (left/right/top/bottom). While dragging, you will see the dark gray rectangle aligned to one of the sides of the editor, or around the entire editor window. If you drop it to one of the sides it will dock to that side of the editor. If you drop it when the rectangle is around the entire window of the editor it will get stacked on top of that editor. You can also grab one of the stacked editors and tile it to one of the sides.

Split Editor Actions

You can divide the editing area vertically and horizontally using the following actions available in the **Editor** toolbar and **Window** menu:

- **Split Editor Horizontally** - Splits the editor horizontally. This is useful for comparing and merging content between two documents.
- **Split Editor Vertically** - Splits the editor vertically. This is useful for comparing and merging content between two documents.
- **Unsplit Editor** - Removes a split action on the editing area.

To maximize or restore the editors, go to **Window** > **Maximize/Restore Editor Area**.

Scroll Wheel Actions

When the opened documents titles do not fit in the tab strip, the scroll wheel can be used to scroll the editor title tabs to the left or right, the same as the two arrows on the top-right. You can also switch between edited files by using the **Next editor** and **Previous editor** actions from the **Window** menu, or by using their respective keyboard shortcuts (**Ctrl + F6 (Command + F6 on OS X)** and **Ctrl + Shift + F6 (Command + Shift + F6 on OS X)**). These two actions display a small pop-up window that allows you to cycle through all opened files.

To watch our video demonstration about *dockable* and floating views and editors in Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/Dockable_Views.html*.

Configure Toolbars

You can configure the toolbars in Oxygen XML Editor to personalize the interface for your specific needs. You can choose which toolbars to show or hide in the current editor mode (**Text**, **Author**,**Design**, or **Grid**) and in the current *perspective* (**Editor**, **XSLT Debugger**, **XQuery Debugger**, or **Databse**). You can also choose which actions to display in each toolbar, add actions to toolbars, and customize the layout of the toolbars.

To configure the toolbars, open the **Configure Toolbars** dialog box by doing one of the following:

- Right-click any toolbar and select **Configure Toolbars**.
- Select **Configure Toolbars** from the **Window** menu.

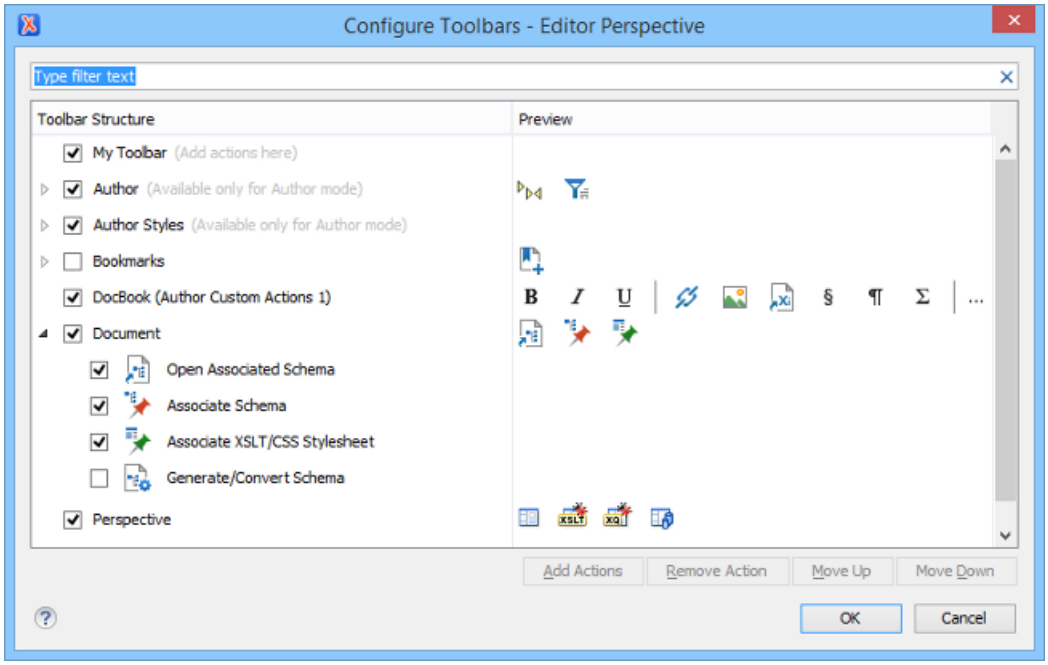


Figure 55: Configure Toolbars Dialog Box

The **Configure Toolbars** dialog box provides the following features:

Filter Text Box

You can use the filter text box at the top of the dialog box to search for a specific toolbar or action.

Show or Hide Toolbars

You can choose whether to show or hide a toolbar by using the checkbox next to the toolbar name. This checkbox is only available for toolbars that are available for the current *perspective* and editing mode.

Show or Hide Actions in a Toolbar

To show or hide actions in a toolbar, expand it by clicking the arrow next to the toolbar name, then use the checkbox to select or deselect the appropriate actions. The toolbar configuration changes in the **Preview** column according to your changes.

Add Actions to a Toolbar

Use the **Add Actions** button to open the **Add Actions** dialog box that displays all the actions that can be added to any of the toolbars, with the exception of those that are contributed from *frameworks* or 3rd party *plugins*.

Remove Actions from a Toolbar

You can remove actions that you have previously added to toolbars by using the **Remove Action** button.

Move Actions in a Toolbar

Use the **Move Up** and **Move Down** actions to change the order of the actions in a toolbar.

The **Configure Toolbars** dialog box also provides a variety of other ways to customize the layout in Oxygen XML Editor.

Customize My Toolbar

You can customize the **My Toolbar** to include your most commonly used actions. By default, this toolbar is listed first. Also, it is hidden until you add actions to it and you can easily hide it with the **Hide "My Toolbar" Toolbar** action that is available when you right-click anywhere in the toolbar area.

Drop-down Menu Actions

Composite actions that are usually displayed as a drop-down menu can only be selected in one toolbar at a time. These actions are displayed in the **Configure Toolbars** dialog box with the name in brackets.

Configure External Tools Action

There is a **Configure external tools** composite action that appears in the toolbar called **Tools**. It is a drop-down menu that contains any external tools that are configured in the **External Tools** preferences page.

Note: If no external tools are configured, this drop-down menu is not shown in the toolbar.

Additional actions are available from the **Window** menu or contextual menu when invoked from a toolbar that allows you to further customize your layout. These actions include:

Reset Toolbars

To reset the layout of toolbars to the default setting, select the **Reset Toolbars** action from the contextual menu or **Window** menu.

Reset Layout

To reset the entire layout (including toolbars, editing modes, views, etc.) to the default setting, select **Reset Layout** from the contextual menu or **Window** menu.

Export Layout

You can use the **Export Layout** action that is available in the **Window** menu to export the entire layout of the application to share it with other users.

Hide Toolbars

You can use the **Hide Toolbar** action from the contextual menu to easily hide a displayed toolbar. When you right-click a toolbar it will be highlighted to show you which actions are included in that toolbar.

Import/Export Transformation or Validation Scenarios

You can export global transformation and validation scenarios into specialized *scenarios* files. You can import transformation and validation scenarios from various sources (such as project files, *framework* option files, or exported scenario files). The import and export scenario actions are available in the **Options** menu. The following actions are available:

Import Transformation Scenarios

Loads a set of transformation scenarios from a project file, *framework* options file, or exported scenarios file.

Export Global Transformation Scenarios

Stores a set of global transformation scenarios in a specialized *scenarios* file.

Import Validation Scenarios

Loads a set of validation scenarios from a project file, *framework* options file, or exported scenarios file.

Export Global Validation Scenarios

Stores a set of global validation scenarios in a specialized *scenarios* file.

The **Export Global Transformation Scenarios** and **Export Global Validation Scenarios** options are used to store all the scenarios in a separate file. Associations between document URLs and scenarios are also saved in this file. You can load the saved scenarios using the **Import Transformation Scenarios** and **Import Validation Scenarios** actions. To distinguish the existing scenarios and the imported ones, the names of the imported scenarios contain the word *import*.

Editor Variables

An editor variable is a shorthand notation for context-dependent information, such as a file or folder path, a time-stamp, or a date. It is used in the definition of a command (for example, the input URL of a transformation, the output file path of a transformation, or the command line of an external tool) to make a command or a parameter generic and re-usable with other input files. When the same command is applied to multiple files, the notation is expanded at the execution of the command so that the same command has different effects depending on the actual file.

Oxygen XML Editor includes a variety of built-in editor variables. You can also create your own *custom editor variables* by using the ***Custom Editor Variables*** *preferences page*.

You can use the following editor variables in Oxygen XML Editor commands of external engines or other external tools, in transformation scenarios, **Author** mode operations, and in validation scenarios:

- *${af}* - The local file path of the ZIP archive that includes the current edited document.
- *${afd}* - The local directory path of the ZIP archive that includes the current edited document.
- *${afdu}* - The URL path of the directory of the ZIP archive that includes the current edited document.
- *${afn}* - The file name (without parent directory and without file extension) of the zip archive that includes the current edited file.
- *${afne}* - The file name (with file extension, for example `.zip` or `.epub`, but without parent directory) of the zip archive that includes the current edited file.
- *${afu}* - The URL path of the ZIP archive that includes the current edited document.
- *${ask('message', type, ('real_value1':'rendered_value1'; 'real_value2':'rendered_value2'; ...), 'default_value')}* - To prompt for values at runtime, use the *ask('message', type, ('real_value1':'rendered_value1'; 'real_value2':'rendered_value2'; ...), 'default-value')* editor variable. You can set the following parameters:
  - *'message'* - The displayed message. Note the quotes that enclose the message.
  - *type* - Optional parameter, with one of the following values:

    Note: The title of the dialog box will be determined by the type of parameter and as follows:

    - For *url* and *relative_url* parameters, the title will be the name of the parameter and the value of the *'message'*.
    - For the other parameters listed below, the title will be the name of that respective parameter.
    - If no parameter is used, the title will be "Input".

Parameter	
url	**Format:** *${ask('message', url, 'default_value')}*
	Description: Input is considered a URL. Oxygen XML Editor checks that the provided URL is valid.
	Example: • *${ask('Input URL', url)}* - The displayed dialog box has the name *Input URL*. The expected input type is URL. • *${ask('Input URL', url, 'http://www.example.com')}* - The displayed dialog box has the name *Input URL*. The expected input type is URL. The input field displays the default value *http://www.example.com*.
password	**Format**: *${ask('message', password, 'default')}*
	Description: The input is hidden with bullet characters.
	Example: • *${ask('Input password', password)}* - The displayed dialog box has the name *'Input password'* and the input is hidden with bullet symbols. • *${ask('Input password', password, 'abcd')}* - The displayed dialog box has the name *'Input password'* and the input hidden with bullet symbols. The input field already contains the default abcd value.
generic	**Format**: *${ask('message', generic, 'default')}*
	Description: The input is considered to be generic text that requires no special handling.
	Example: • *${ask('Hello world!')}* - The dialog box has a *Hello world!* message displayed. • *${ask('Hello world!', generic, 'Hello again!')}* - The dialog box has a *Hello world!* message displayed and the value displayed in the input box is *'Hello again!'*.
relative_url	**Format:** *${ask('message', relative_url, 'default')}*
	Description: Input is considered a URL. Oxygen XML Editor tries to make the URL relative to that of the document you are editing. **Note:** If the *$ask* editor variable is expanded in content that is not yet saved (such as an *untitled* file, whose path cannot be determined), then Oxygen XML Editor will transform it into an absolute URL.
	Example: • *${ask('File location', relative_url, 'C:/example.txt')}* - The dialog box has the name *'File location'*. The URL inserted in the input box is made relative to the current edited document location.
combobox	**Format:** *${ask('message', combobox, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default')}*

Parameter	
	Description: Displays a dialog box that offers a drop-down menu. The drop-down menu is populated with the given *rendered_value* values. Choosing such a value will return its associated value (*real_value*). **Note:** The *'default'* parameter specifies the default selected value and can match either a key or a value.
	Example: • *${ask('Operating System', combobox, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'osx')}* - The dialog box has the name *'Operating System'*. The drop-down menu displays the three given operating systems. The associated value will be returned based upon your selection. **Note:** In this example, the default value is indicated by the *osx* key. However, the same result could be obtained if the default value is indicated by *Mac OS X*, as in the following example: *${ask('Operating System', combobox, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'Mac OS X')}* • *${ask('Mobile OS', combobox, ('win':'Windows Mobile';'ios':'iOS';'and':'Android'), 'Android')}*
editable_combobox	**Format:** *${ask('message', editable_combobox, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default')}*
	Description: Displays a dialog box that offers a drop-down menu with editable elements. The drop-down menu is populated with the given *rendered_value* values. Choosing such a value will return its associated real value (*real_value*) or the value inserted when you edit a list entry. **Note:** The *'default'* parameter specifies the default selected value and can match either a key or a value.
	Example: • *${ask('Operating System', editable_combobox, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'osx')}* - The dialog box has the name *'Operating System'*. The drop-down menu displays the three given operating systems and also allows you to edit the entry. The associated value will be returned based upon your selection or the text you input.
radio	**Format:** *${ask('message', radio, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default')}*
	Description: Displays a dialog box that offers a series of radio buttons. Each radio button displays a *'rendered_value* and will return an associated *real_value*. **Note:** The *'default'* parameter specifies the default selected value and can match either a key or a value.

Parameter	
	Example: • *${ask('Operating System', radio, ('win':'Microsoft Windows','osx':'Mac OS X','lnx':'Linux/UNIX'), 'osx')}* - The dialog box has the name *'Operating System'*. The radio button group allows you to choose between the three operating systems. **Note:** In this example `Mac OS X` is the default selected value and if selected it would return *osx* for the output.

- 'default-value' - optional parameter. Provides a default value.

- *${author.name}* - Expands to the current author name that is set in the ***Review*** *preferences page*.
- *${caret}* - The position where the cursor is located. This variable can be used in a code template, in **Author** mode operations, or in a **selection** *plugin*.
- *${cf}* - Current file as file path, that is the absolute file path of the current edited document.
- *${cfd}* - Current file folder as file path, that is the path of the current edited document up to the name of the parent folder.
- *${cfdu}* - Current file folder as URL, that is the path of the current edited document up to the name of the parent folder, represented as a URL.
- *${cfn}* - Current file name without extension and without parent folder. The current file is the one currently opened and selected.
- *${cfne}* - Current file name with extension. The current file is the one currently opened and selected.
- *${configured.ditaot.dir}* - The default directory of the DITA Open Toolkit distribution, as configured in the ***DITA*** *preferences page*.
- *${cp}* - Current page number. Used to display the current page number on each printed page in the **Editor / Print** Preferences page.
- *${currentFileURL}* - Current file as URL, that is the absolute file path of the current edited document represented as URL.
- *${date(pattern)}* - Current date. The allowed patterns are equivalent to the ones in the *Java SimpleDateFormat class*. **Example:** *yyyy-MM-dd*;

  Note: This editor variable supports both the xs:date and xs:datetime parameters. For details about xs:date, go to *http://www.w3.org/TR/xmlschema-2/#date*. For details about xs:datetime, go to *http://www.w3.org/TR/xmlschema-2/#dateTime*.
- *${dbgXML}* - The local file path to the XML document that is current selected in the Debugger source combo box (for tools started from the XSLT/XQuery Debugger).
- *${dbgXSL}* - The local file path to the XSL/XQuery document that is current selected in the Debugger stylesheet combo box (for tools started from the XSLT/XQuery Debugger).
- *${dita.dir.url}* - A special local contextual editor variable that gets expanded only in the **Libraries** dialog box that is accessible from the **Advanced** tab of DITA transformation scenarios. The **Libraries** dialog box allows you to specify additional libraries (*JAR* files or additional class paths) to be used by the transformer. This *${dita.dir.url}* editor variable gets expanded to the value of the *dita.dir* parameter from the **Parameters** tab of the DITA transformation scenario.
- *${ds}* - The path of the detected schema as a local file path for the current validated XML document.
- *${dsu}* - The path of the detected schema as a URL for the current validated XML document.
- *${env(VAR_NAME)}* - Value of the *VAR_NAME* environment variable. The environment variables are managed by the operating system. If you are looking for Java System Properties, use the *${system(var.name)}* editor variable.
- *${framework(fr_name)}* - The path (as URL) of the *fr_name framework*.
- *${framework}* - The path (as URL) of the current *framework*, as part of the `[OXYGEN_INSTALL_DIR]/frameworks` directory.
- *${frameworkDir(fr_name)}* - The path (as file path) of the *fr_name framework*.

  Note: Since multiple *frameworks* might have the same name (although it is not recommended), for both *${framework(fr_name)}* and *${frameworkDir(fr_name)}* editor variables Oxygen XML Editor employs the following algorithm when searching for a given *framework* name:

- All *frameworks* are sorted, from high to low, according to their ***Priority*** setting from the ***Document Type*** *configuration dialog box*. Only *frameworks* that have the **Enabled** checkbox selected are taken into account.
- Next, if the two or more *frameworks* have the same name and priority, a further sorting based on the **Storage** setting is made, in the exact following order:
  - *Frameworks* stored in the internal Oxygen XML Editor options.
  - Additional *frameworks* added in the ***Locations*** *preferences page*.
  - *Frameworks* installed using the add-ons support.
  - *Frameworks* found in the *main framework location* (**Default** or **Custom**).

- *${frameworkDir}* - The path (as file path) of the current *framework*, as part of the `[OXYGEN_INSTALL_DIR]/frameworks` directory.
- *${frameworks}* - The path (as URL) of the `[OXYGEN_INSTALL_DIR]` directory.
- *${frameworksDir}* - The path (as file path) of the `[OXYGEN_INSTALL_DIR]/frameworks` directory.
- *${home}* - The path (as URL) of the user home folder.
- *${homeDir}* - The path (as file path) of the user home folder.
- *${i18n(key)}* - Editor variable used only at *framework*-level to allow translating names and descriptions of **Author** mode actions in multiple actions. For more details, see *Localizing Frameworks* on page 1203.
- *${id}* - Application-level unique identifier. It is a short sequence of 10-12 letters and digits that is not guaranteed to be universally unique.
- *${makeRelative(base,location)}* - Takes two URL-like paths as parameters and tries to return a relative path. A use-case would be to insert content references to a certain reusable component when defining code templates.

  Example:

  ```
  ${makeRelative(${currentFileURL}, ${dictionaryURL}#gogu)}
  ```

- *${oxygenHome}* - Oxygen XML Editor installation folder as URL.
- *${oxygenInstallDir}* - Oxygen XML Editor installation folder as file path.
- *${pd}* - The file path for the parent folder of the current project selected in the **Project** view.
- *${pdu}* - The URL for the parent folder of the current project selected in the **Project** view.
- *${pn}* - Current project name.
- *${ps}* - Path separator, which is the separator that can be used on the current platform (Windows, OS X, Linux) between library files specified in the class path.
- *${selection}* - The current selected text content in the current edited document. This variable can be used in a code template, in **Author** mode operations, or in a **selection** *plugin*.
- *${system(var.name)}* - Value of the *var.name* Java System Property. The Java system properties can be specified in the command line arguments of the Java runtime as *-Dvar.name=var.value*. If you are looking for operating system environment variables, use the *${env(VAR_NAME)}* editor variable instead.
- *${timeStamp}* - Time stamp, that is the current time in Unix format. For example, it can be used to save transformation results in multiple output files on each transformation.
- *${tp}* - Total number of pages in the document. Used to display the total number of pages on each printed page in the **Editor / Print** Preferences page.
- *${tsf}* - The transformation result file path. If the current opened file has an associated scenario that specifies a transformation output file, this variable expands to it.
- *${uuid}* - Universally unique identifier, a unique sequence of 32 hexadecimal digits generated by the Java *UUID* class.
- *${xpath_eval(expression)}* - Evaluates an XPath 3.0 expression. Depending on the context, the expression can be:
  - *static* - When executed in a non-XML context. For example, you can use such static expressions to perform string operations on other editor variables for composing the name of the output file in a transformation scenario's **Output** tab.

    Example:

    ```
    ${xpath_eval(upper-case(substring('${cfn}', 1, 4)))}
    ```

- *dynamic* - When executed in an XML context. For example, you can use such dynamic expression in a code template or as a value of a parameter of an **Author** mode operation.

  Example:

  ```
  ${ask('Set new ID attribute', generic, '${xpath_eval(@id)}')}
  ```

Related Information:

Code Templates on page 308

Selection Plugin Extension on page 1346

This type of *plugin* allows you to manage selections of text.

Installing and Updating Add-ons in Oxygen XML Editor on page 45

Custom Editor Variables

An *editor variable* can be created and included in any user-defined expression where a built-in editor variable is also allowed. For example, a custom editor variable may be necessary for configuring the command line of an external tool, the working directory of a custom validator, the command line of a custom XSLT engine, or a custom FO processor.

You can create or configure custom editor variables in the ***Custom Editor Variables*** *preferences page*. To create a custom editor variable, follow these steps:

1. *Open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Custom Editor Variables**.
2. Click the **New** button at the bottom of the table.
3. Use the subsequent dialog box to specify the **Name**, **Value**, and **Description** for the new editor variable.
4. Click **OK** to save your configuration.

Related Information:

Editor Variables on page 169

Custom System Properties

A variety of Java system properties can be set in the application to influence its behavior. For information about how to do this, see *Setting a system property* on page 179.

Table 4: Custom System Properties

Property	Allowed values	Default value	Purpose
com.oxygenxml.disable.http.protocol.handlers	*true* or *false*	*false*	By default, Oxygen XML Editor uses the open source Apache HTTP Client software for HTTP(S) connections. If set to *True*, the default Java Sun HTTP(S) will be used instead. You will also lose **WebDAV** support and possibly other related features.
com.oxygenxml.present.license.reminders	*true* or *false*	*true*	When set to *false*, Oxygen XML Editor will not display the messages that remind you to renew your Support and Maintenance Pack that covers your current license.

Property	Allowed values	Default value	Purpose
com.oxygenxml.enable.content.reference.caching	*true* or *false*	*true*	Enables content reference caching.
com.oxygenxml.default.java.accessibility	*true* or *false*	*false*	System property that can be set to "true" to force the default detection of java accessibility. If *com.sun.java.accessibility.AccessBridge* cannot be loaded, Oxygen XML Editor forces the Java accessibility to be disabled.
com.oxygenxml.floating.license.timeout			Stores the time interval (in minutes) before floating licenses are released in case of application's inactivity.
com.oxygenxml.language	Language code (for example, *en-us*)	N/A	Property that holds the language code set during installation.
com.oxygenxml.default.options	A URL-type relative or absolute path.	N/A	Provides the path to an XML file containing default application options. For more details, see *Customizing Default Options* on page 161.
com.oxygenxml.customOptionsDir	A file system absolute path pointing to a folder.	N/A	Sets a folder to be used by the application to load and save preference files. The default location where the options are saved varies according to the operating system. See *Importing/Exporting/Resetting Global Options* on page 164.
com.oxygenxml.ApplicationDataFolder (Windows only)	A file system absolute path pointing to a folder.	*%APPDATA%*	When the application runs on Windows, you can set this property to change the location where the application considers that the *APPDATA* folder is located.
com.oxygenxml.editor.frameworks.url	A URL-type absolute path.	*OXYGEN_DIR\frameworks*	Changes the folder where the application considers that the main *frameworks* are installed. It has the same effect as changing the custom *frameworks* directory value in the ***Location*** *preferences page*.

Property	Allowed values	Default value	Purpose
com.oxygenxml.MultipleInstances	*true* or *false*	*false*	If set to *true*, multiple instances of the application are allowed to be started.
com.oxygenxml.xep.location	A file system absolute path pointing to a folder.	N/A	Points to a folder where RenderX XEP is installed. Has the same effect as configuring XEP in the ***FO Processors*** *preferences page*.
com.oxygenxml.additional.classpath	A list of *JAR*-type resources separated by a classpath separator.	N/A	An additional list of libraries to be used in the application's internal class loader in addition to the libraries specified in the *lib* folder.
com.oxygenxml.user.home (Windows only)	A file system absolute path pointing to a folder.	*USERPROFILE* Folder	Overwrites the user home directory that was implicitly detected for the application.
com.oxygenxml.use.late.delegation.for.author.extensions	*true* or *false*	*true*	All Java extensions in a *framework* configuration are instantiated in a separate class loader. When **true**, the *JAR* libraries used in a certain document type will have priority to resolve classes before delegating to the parent class loader. When **false**, the parent class loader will take precedence.
com.oxygenxml.stack.size.validation.threads	The number of bytes used for validation threads.	5*1024*1024	Some parts of the application (validation, content completion) that use the Relax NG parser sometimes require a larger *Thread* stack size to parse complex schemas. The default value should be more than enough.
com.oxygenxml.jing.skip.validation.xhtml.data.attrs	*true* or *false*	*true*	By default, the Relax NG validation was configured to skip validation for XHTML attributes that start with "data-", which should be skipped from validation according to the XHTML 5 specification.

Property	Allowed values	Default value	Purpose
com.oxygenxml.report.problems.url	User defined URL	N/A	The URL where a problem reported through the **Report Problem** dialog box is sent. The report is sent in XML format using the *report* parameter with the POST HTTP method.
com.oxygenxml.parallel.title.computing.threads	Integers	4	The number of parallel threads that will be used to compute referenced topic titles. Increasing this value reduces the amount of time it takes to compute topic titles in the **DITA Maps Manager** view.

Related Information:

Setting a system property on page 179

Localizing the User Interface

You can select the language of the Oxygen XML Editor user interface. Oxygen XML Editor ships with the following languages: English, French, German, Japanese, and Dutch. To change the interface language, go to **Options** > **Preferences** > **Global** preferences page, then choose the appropriate language from the **Language** drop-down menu.

You can also localize the interface in another language by *creating an interface localization file*.

Creating an Interface Localization File

You can change the language of the Oxygen XML Editor user interface by creating an interface localization file. The example in this procedure is based on the Spanish language, and a standard Oxygen XML Editor Windows distribution.

1. Identify the code for the new language you want to translate the interface. It is composed from a language code (two or three lowercase letters that conform to the *ISO 639 standard*), followed by an underscore character, and a region code (two or three uppercase letters that conform to the *ISO 3166 standard*).
2. Write an email to the Oxygen XML Editor support team and ask them to send you the `translation.xml` sample file.
3. Open `translation.xml` in Oxygen XML Editor. The file contains all the interface messages that can be translated and is updated at every new release with the latest additions. Here is a sample of its content:

```
<translation>
    <languageList>
        <language description="English" lang="en_US"/>
    </languageList>
    <key value="New">
        <comment>The File/New action. Creates a new document.</comment>
        <val lang="en_US">New</val>
    </key>
    <key value="New_folder">
        <comment>Creates a folder in the Project View.</comment>
        <val lang="en_US">New Folder</val>
    </key>
    .....
</translation>
```

4. Update the `language` element to reflect the new language. Set the `description` attribute to `Spanish` and the `lang` attribute to `es_ES`.
5. For each `key` element translate the `comment` (optional) and `val` elements. Set the `lang` attribute to `es_ES`.

Note: After you are finished, the file should look like this:

```
<translation>
    <languageList>
        <language description="Español" lang="es_ES"/>
    </languageList>
    <key value="New">
        <comment>El Archivo / Nueva acción. Crea un nuevo documento.</comment>
        <val lang="es_ES">Nuevo</val>
    </key>
    <key value="New_folder">
        <comment>Crea una carpeta en la vista del proyecto.</comment>
        <val lang="es_ES">Nueva carpeta</val>
    </key>
    .....
</translation>
```

6. *Open the **Preferences** dialog box **(Options** > **Preferences)***, go to **Global**, and select the ***Other language** option*. Browse for the `translation.xml` file.
7. Restart the application.

Setting a Java Virtual Machine Parameter when Launching Oxygen XML Editor

You can set Java Virtual Machine parameters (for example, if you want to increase the maximum amount of memory available) for the Oxygen XML Editor *application launchers* or *command-line scripts*. You can also create a *custom startup parameters file*.

Setting Parameters for the Application Launchers

Increasing the amount of memory that Oxygen XML Editor uses on Windows

To increase the memory available to Oxygen XML Editor on Windows:

- Browse to the installation directory of Oxygen XML Editor.
- Locate the -Xmx parameter in the `oxygen19.0.vmoptions` file.

  Note: For 32-bit Windows modify the parameter to -Xmx1024m or larger, but not over -Xmx1200m. Make sure you do not exceed your physical RAM. For 64-bit Windows modify the parameter to a larger value (for example, -Xmx2048m). We recommended you to not use more than half of your existing physical RAM.

Restart Oxygen XML Editor. Go to **Help** > **About** and verify the amount of memory that is actually available (see the **JVM Memory Used** in the last row in the **Copyright** tab). If Oxygen XML Editor does not start and you receive and error message saying that it could not start the JVM, decrease the -Xmx parameter and try again.

For Windows Vista/7/8/10, copy the `oxygen19.0.vmoptions` to your desktop (or to any other folder with write access), modify it there, then copy it back to the Oxygen XML Editor installation folder.

Note: The parameters from the `.vmoptions` file are used when you start Oxygen XML Editor with the *oxygen* launcher (or with the desktop shortcut). In case you use the command line script `oxygen.bat/oxygen.sh`, modify the -Xmx parameter in the script file.

Increasing the amount of memory that Oxygen XML Editor uses on OS X

To increase the memory available to Oxygen XML Editor on OS X:

- **Ctrl + Single-Click (Command + Single-Click on OS X)** (or right-click) the Oxygen XML Editor icon in **Finder**.
- From the contextual menu, select **Show Package Contents**.
- Go to the `contents` directory and edit the `Info.plist` file.

  Note: You can open this file either with the **Property List Editor**, or the **TextEdit**.
- Look for the **VMOptions** key and adjust the -Xmx parameter to a larger value (for example, -Xmx1500m)

  Note: For a Mac kit bundled with Java 8, look for the **VMOptionArray** key and add the -Xmx parameter in a new string `element` from the `array` element. For example, for 1500 MB use the following:

```
<string>-Xmx1500m</string>
```

Tip: Try not to use more than half of your existing physical RAM if possible.

Setting a system property

To set a system property, edit the application launcher and add a parameter after the `%OXYGEN_JAVA%` token, using the following form:

```
-Dproperty.name=value
```

You can also set a system property through a parameter prefixed with `-Doxy` in the command line used to start the application:

```
oxygen19.0.exe "-Doxyproperty.name=value"
```

All system properties are displayed in the **System properties** tab of the **About** dialog box.

To view the list of Oxygen XML Editor system properties, go to *Custom System Properties* on page 174.

Disabling DPI Scaling

Some users may prefer the look of smaller icons in an HiDPI display. To achieve this, display scaling needs to be disabled for high DPI settings. To disable the DPI scaling, set the following property in `.vmoptions` (or in the `.bat` script):

```
sun.java2d.dpiaware=false
```

Setting Parameters in the Command Line Scripts

If you start Oxygen XML Editor with the `oxygen.bat` command line script, you have to add or modify the **-Xmx** parameter to the java command at the end of the script.

For example, to set the maximum amount of Java memory to 600 MB in **Windows**, modify the **-Xmx** parameter like this:

```
java -Xmx600m -Dsun.java2d.noddraw=true ...
```

On **Mac OS X**, the java command should look like this:

```
java "-Xdock:name=Oxygen"\
 -Dcom.oxygenxml.editor.plugins.dir="$OXYGEN_HOME/plugins"\
 -Xmx600m\
 ...
```

On **Linux**, the Java command should look like this:

```
java -Xmx600m\
 "-Dcom.oxygenxml.editor.plugins.dir=$OXYGEN_HOME/plugins"\
```

Creating Custom Startup Parameters File

The startup launchers for Oxygen XML Editor and its executable internal tools (**Tree Editor**, **XML Schema Regular Expressions Builder**, **Large File Viewer**, **SVN Client**, **Compare Directories**, and **Compare Files**) include a default `.vmoptions` file that contain some startup parameters (such the -Xmx parameter, which is used for allocating memory for that particular application). You can edit the parameters in these `.vmoptions` files so that the applications will launch with your desired values. However, if you re-install the application, install an update for the application, or deploy it to other users or machines, those parameters will be reset to their default values.

To avoid resetting user-defined startup parameters, you can create custom `.vmoptions` files and the application and the executable tools will automatically include your custom parameters at startup. The following custom files are recognized by the application and the executable tools:

- `custom_commons.vmoptions` - The parameters and their values of this file will be included in all the startup launchers.

- `custom_<app name>.vmoptions` - The *<app name>* is the name of the executable application or tool (for example, `custom_diffFiles.vmoptions` for the **Compare Files** tool). The parameters and their values of this file will be included in the startup launcher for this particular executable.

To be recognized and included, these custom startup parameter files must be saved in the installation directory of Oxygen XML Editor.

Perspectives

5

Topics:

- *Editor Perspective*
- *XSLT Debugger Perspective*
- *XQuery Debugger Perspective*
- *Database Perspective*

An Oxygen XML Editor *perspective* is an interface layout geared towards a specific use. The Oxygen XML Editor interface uses standard interface conventions and components to provide a familiar and intuitive editing environment across all operating systems. There are several *perspectives* that you can use to work with documents in Oxygen XML Editor. You can change the *perspective* by selecting the respective icon ( ) in the top-right corner of Oxygen XML Editor or by selecting the *perspective* from the **Window** > **Open Perspective** menu.

Editor Perspective

The **Editor** *perspective* is the most commonly used *perspective* and it is the default *perspective* when you start Oxygen XML Editor for the first time. It is the *perspective* that you will use to edit the content of your XML documents.

To switch the focus to this *perspective*, select the **Editor** button in the top-right corner of Oxygen XML Editor (or select **Editor** from the **Window** > **Open perspective** menu)

The layout of this *perspective* is composed of the following components:

Menus

Provides menu driven access to all the features and functions available in Oxygen XML Editor. Most of the menus are common for all types of documents. However, Oxygen XML Editor also includes some context-sensitive and *framework*-specific menus that are only available for a specific context or type of document.

Toolbars

Provides easy access to common and frequently used functions. Each icon is a button that acts as a shortcut to a related function. Most of the toolbars are common for all types of documents. However, **Author** mode also includes *framework*-specific toolbars, depending on the type of document that is being edited (for example, if you are editing a DITA document, a *DITA Author Custom Actions* toolbar is available that includes operations that are specific to DITA documents). The *toolbars can be configured* to suit your specific needs.

Editor Pane

The main editing pane where you spend most of your time reading, editing, applying markup, and validating your documents.

Views

Oxygen XML Editor includes a large variety of *dockable* views to assist you with editing, viewing, searching, validating, transforming, and organizing your documents. The most commonly used views are displayed by default and you can choose to display others by selecting them from the **Window** > **Show View** menu. The *layout of the views can also be configured* according to your preferences.

When two or more views are displayed, the application provides divider bars. Divider bars can be dragged to a new position increasing the space occupied by one panel while decreasing it for the other.

As the majority of the work process centers around the Editor area, other views can be hidden using the toggle controls located on the top corner of the view ([on Mac OS X]).

Some of the most helpful views in the **Editor** *perspective* include the following:

- ***Project*** *view* - Enables the definition of projects and logical management of the documents they contain.
- ***DITA Maps Manager*** *view* - For DITA document types, this view helps you organize, manage, and edit DITA topics and maps.
- ***Open/Find Resource*** *view* - Designed to offer advanced search capabilities in various scopes.
- ***Outline*** *view* - It provides an XML tag overview and offers a variety of functions, such as modifications follow-up, document structure change, document tag selection, and elements filtering.
- ***Results view*** - Displays the messages generated as a result of user actions such as *validations*, *transformation scenarios*, *spell checking in multiple files*, search operations, and others. Each message is a link to the location related to the event that triggered the message.
- ***Attributes*** *view* - Presents all possible attributes of the current element and allows you to edit attribute values. You can also use this view to insert attributes in **Text** mode. **Author** mode also includes an *in-place attribute editor*.
- ***Model*** *view* - Presents the current edited element structure model and additional documentation as defined in the schema.
- ***Elements view*** - Presents a list of all defined elements that you can insert at the current cursor position according to the document's schema. In ***Author*** *mode this view* includes tabs that present additional information relative to the cursor location.
- ***Entities*** *view* - Displays a list with all entities declared in the current document as well as built-in ones.
- ***Transformation Scenarios*** *view* - Displays a list with all currently configured transformation scenarios.
- ***XPath/XQuery Builder*** *view* - Displays the results from running an XPath expression.
- ***WSDL SOAP Analyzer*** *view* - Provides a tool that helps you test if the messages defined in a Web Service Descriptor (WSDL) are accepted by a Web Services server.

Related Information:

Editing Documents on page 248
Editing Modes on page 186
Configuring the Layout of the Views and Editors on page 165

XSLT Debugger Perspective

The **XSLT Debugger** *perspective* allows you to detect problems in an XSLT transformation by executing the process step by step in a controlled environment. To switch the focus to this *perspective*, select the **XSLT Debugger** button in the top-right corner of the interface or **Window** > **Open perspective** > **XSLT Debugger**.

The workspace in this *perspective* is organized as an editing area assisted by special helper views. The editing area contains editor panels that you can *split horizontally or vertically* in a stack of XML editor panels and a stack of XSLT editor panels. The XML files and XSL files can be edited in ***Text*** *mode* only.

The layout of this *perspective* is composed of the following components:

Menus

Provides menu driven access to all the features and functions available in the **XSLT Debugger**.

Toolbars

Contains all actions needed to configure and control the debugging process. The *toolbars can be configured* to suit your specific needs.

XML Source Pane

The editing pane where you can display and edit data or document-oriented XML documents.

XSL Source Pane

The editing pane where you can display and edit XSL stylesheets.

Output View

Displays the transformed output that results from the input of a selected document (XML) and selected stylesheet (XSL) to the transformer. The result of transformation is dynamically written as the transformation is processed. There are three types of views for the output: a text view (with XML syntax highlight), an XHTML view, and one text view for each `xsl:result-document` element used in the stylesheet (if it is an XSLT 2.0 / 3.0 stylesheet).

Debugging Information Views

Presented in two panes, they display various types of information that can be used to understand the transformation process. For each information type there is a corresponding tab. While running a transformation, relevant events are displayed in the various information views. This allows you to obtain a clear view of the transformation progress. See the *Debugging Information Views* topic for a list of all the information views (and links to more details on each view).

Note: You can add XPath expression automatically in the **XWatch** view using the **Watch expression** action from the contextual menu. In case you select an expression or a fragment of it and then click **Watch expression** in the contextual menu, the entire selection is presented in the **XWatch** view. Using **Watch expression** without selecting an expression displays the value of the attribute from the cursor position in the **XWatch** view. Variables detected at the cursor position are also displayed. Expressions displayed in the **XWatch** view are *normalized* (unnecessary white spaces are removed from the expression).

To watch our video demonstration about the XSLT debugging capabilities in Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/XSLT_Debugger.html*.

Related Information:

Debugging XSLT Stylesheets and XQuery Documents on page 1142

XQuery Debugger Perspective on page 183

XQuery Debugger Perspective

The **XQuery Debugger** *perspective* allows you to detect problems in an XQuery transformation process by executing the process step by step in a controlled environment and inspecting the information provided in the special views. To switch the focus to this *perspective*, select the **XQuery Debugger** button in the top-right corner of the interface or **Window** > **Open perspective** > **XQuery Debugger**.

The workspace in this *perspective* is organized as an editing area assisted by special helper views. The editing area contains editor panels that you can *split horizontally or vertically* in a stack of XML editor panels and a stack of XQuery editor panels. The XML files and XQuery files can be edited in ***Text** mode* only.

The layout of this *perspective* is composed of the following components:

Menus

Provides menu driven access to all the features and functions available in the **XQuery Debugger**.

Toolbars

Contains all actions needed to configure and control the debugging process. The *toolbars can be configured* to suit your specific needs.

XML Source Pane

The editing pane where you can display and edit data or document-oriented XML documents.

XQuery Source Pane

The editing pane where you can display and edit XQuery files.

Output View

Displays the transformed output that results from the input of a selected document (XML) and selected XQuery document to the XQuery transformer. The result of transformation is dynamically written as the transformation is processed. There are two types of views for the output: a text view (with XML syntax highlight) and an XHTML view.

Debugging Information Views

Presented in two panes, they display various types of information that can be used to understand the transformation process. For each information type there is a corresponding tab. While running a transformation, relevant events are displayed in the various information views. This allows you to obtain a clear view of the transformation progress. See the *Debugging Information Views* topic for a list of all the information views (and links to more details on each view).

Note: You can add XPath expression automatically in the **XWatch** view using the **Watch expression** action from the contextual menu. If you select an expression, or a fragment of it, and then click **Watch expression**

in the contextual menu, the entire selection is presented in the **XWatch** view. Expressions displayed in the **XWatch** view are normalized (unnecessary white spaces are removed from the expression).

To watch our video demonstration about the XQuery debugging capabilities in Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/XQuery_Debugger.html*.

Related Information:

Debugging XSLT Stylesheets and XQuery Documents on page 1142

XSLT Debugger Perspective on page 182

Database Perspective

The **Database** *perspective* allows you to manage databases. To switch the focus to this *perspective*, select the **Database** button in the top-right corner of Oxygen XML Editor or **Window** > **Open perspective** > **Database** from the **Window** > **Open perspective** menu.

The **Database** *perspective* offers various helpful features, including:

- Support for browsing multiple connections at the same time.
- Support for both *Relational* and *Native XML* databases.
- Browsing the structure of databases.
- Viewing tables from databases.
- Inspecting or modifying data.
- Specifying XML Schemas for XML fields.
- SQL execution.
- XQuery execution.
- Data export to XML.

Supported Databases

Oxygen XML Editor supports numerous types of databases, including:

- Oracle Berkeley DB XML Database
- eXist XML Database
- IBM DB2 (Enterprise edition only)
- JDBC-ODBC Bridge
- MarkLogic (Enterprise edition only)
- Microsoft SQL Server 2005 and Microsoft SQL Server 2008 (Enterprise edition only)
- MySQL
- Oracle 11g (Enterprise edition only)
- PostgreSQL 8.3 (Enterprise edition only)
- Documentum xDB (X-Hive/DB) 10 XML Database (Enterprise edition only)
- Documentum (CMS) 6.5 (Enterprise edition only)
- SharePoint (CMS)

Note: For the databases marked with "Enterprise edition only", the XML capabilities are only available in the Enterprise edition of Oxygen XML Editor. For a detailed feature matrix that compares the Academic, Professional, and Enterprise editions of Oxygen XML Editor *go to the Oxygen XML Editor website*.

Supported Capabilities

The supported non-XML capabilities are as follows:

- Browsing the structure of the database instance.
- Opening a database table in the ***Table Explorer** view*.
- Handling the values from **XML Type** columns as String values.

Note: The non-XML capabilities are available in the Enterprise, Academic, and Professional editions of Oxygen XML Editor by registering the database driver as a *Generic JDBC* type driver when defining the data source for accessing the database. For more information, see *Database Connection Support* on page 1072.

The supported XML capabilities are as follows:

- Displaying an XML Schema node in the tree of the database structure (for databases with an XML-specific structure) with actions for opening, editing, and validating the schemas in an Oxygen XML Editor editor panel.
- Handling the values from **XML Type** columns as XML instance documents that can be opened and edited in an Oxygen XML Editor editor panel.
- Validating an XML instance document added to an XML Type (column of a table, etc.)

Tip: Connections configured on relational data sources can be used to import data to XML or to generate XML schemas.

Layout of the Database Perspective

The layout of this *perspective* is composed of the following components:

Menus

Provides menu driven access to all the features and functions available in the **XQuery Debugger**.

Toolbars

Contains all actions needed to configure and control the debugging process. The *toolbars can be configured* to suit your specific needs.

Editor Pane

The main editing pane where you spend most of your time reading, editing, applying markup, and validating your documents.

Data Source Explorer View

Provides browsing support for the configured connections.

Table Explorer View

Provides table content editing support for inserting new rows, deleting table rows, editing cell values, exporting to an XML file, and more.

Related Information:

Working with Databases on page 1068

Data Source Explorer View on page 1068

Table Explorer View on page 1069

Editing Modes

6

Topics:

- *Text Editing Mode*
- *Grid Editing Mode*
- *Author Editing Mode*
- *Design Editing Mode (XML Schema Diagram Editor)*

The main editing area in Oxygen XML Editor includes several editing modes to suit the type of editing that you want to perform. You can easily switch between modes by clicking on the desired mode at the bottom of the main editing pane. Oxygen XML Editor offers the following editing modes:

- ***Text*** - This mode presents the source of an XML document.
- ***Grid*** - This mode displays an XML document as a structured grid of nested tables.
- ***Author*** - This mode enables you to edit in a WYSIWYG-like editor.
- ***Design*** - This mode is found in the schema editor and represents the schema as a diagram.

The default editing mode that will be initially opened for each type of document can be set in two ways:

- If the ***Allow Document Type specific edit mode setting to override the general mode setting*** *option* is selected in the **Edit Modes** preferences page, then the edit mode specified in the ***Document Type*** *configuration dialog box* is used when that particular type of document is initially opened.
- If the ***Allow Document Type specific edit mode setting to override the general mode setting*** *option* is not selected, then the edit mode specified in the *table in the* ***Edit Modes*** *preferences page* is used when that particular type of document is initially opened.

Text Editing Mode

The **Text** mode editor in Oxygen XML Editor is designed to be a simple, yet powerful, XML source editor. It provides support to help you edit, transform, and debug XML-based documents. It is similar to other common text editors, but Oxygen XML Editor also includes specialized editing actions, a powerful *Content Completion Assistant*, a helpful ***Outline*** *view*, and many other unique features.

To switch to this mode, select **Text** at the bottom of the editing area.

Related Information:

Editing XML Documents in Text Mode on page 290

Navigating the Document Content in Text Mode

Oxygen XML Editor includes some useful features to help you navigate XML documents in **Text** mode.

Using the Keyboard

Oxygen XML Editor allows you to quickly navigate through a document using the **Ctrl + CloseBracket (Command + CloseBracket on OS X)** key to go to the next XML node and **Ctrl + OpenBracket (Command + OpenBracket on OS X)** to go to the previous one.

To navigate one word forward or backwards, use **Ctrl + RightArrow (Command + RightArrow on OS X)**, and **Ctrl + LeftArrow (Command + LeftArrow on OS X)**, respectively. To position the cursor at the beginning or end of the document you can use **Ctrl + Home (Command + Home on OS X)**, and **Ctrl + End (Command + End on OS X)**, respectively.

Navigation Buttons

Oxygen XML Editor includes some actions that help you to quickly navigate to a particular modification. These navigation buttons are available in the main toolbar and the actions can also be accessed from the **Find** menu. The three actions include:

- **Last Modification** - Moves the cursor to the last modification in any opened document.
- **Back** - Moves the cursor to the previous position.
- **Forward** - Moves the cursor to the next position. Available after you use the **Back** button at least once.

Navigating with the Outline View

Oxygen XML Editor includes a very useful ***Outline*** *view* that displays a hierarchical tag overview of the currently edited XML Document.

You can use this view to quickly navigate through the current document by selecting nodes in the outline tree. It is synchronized with the editor area, so when you make a selection in the **Outline** view, the corresponding nodes are highlighted in the editor area.

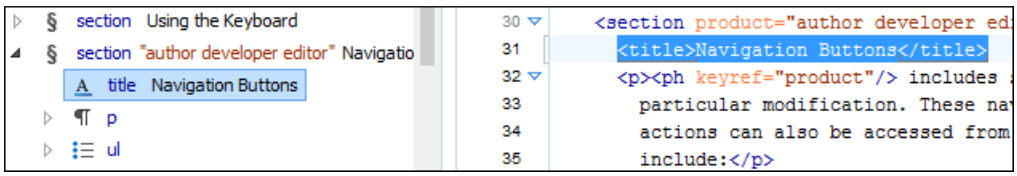


Figure 56: Outline View Navigation in Text Mode

Using the Breadcrumb to Navigate

A *breadcrumb* on the top stripe indicates the path from the document root element to the current element. It can also be used as a helpful tool to navigate to specific elements throughout the structure of the document.

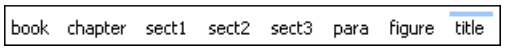


Figure 57: Breadcrumb in Text Mode

The last element listed in the *breadcrumb* is the element at the current cursor position. The current element is also highlighted by a thin light blue bar for easy identification. Clicking an element from the *breadcrumb* selects the entire element and navigates to it in the editor area.

Navigating with the Go To Dialog Box

In **Text** mode, you can navigate precisely to a location in the document you are editing by using the **Go to** dialog box. To open this dialog box, go to **Find** > **Go to** **(Ctrl+L (Command+L on OS X))**.

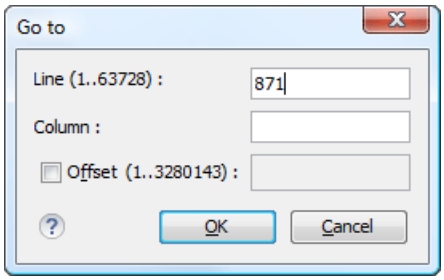


Figure 58: Go to Dialog Box

The dialog box includes the following fields for specifying a specific navigation location:

- **Line** - Destination line in the current document.
- **Column** - Destination column in the current document.
- **Offset** - Destination offset relative to the beginning of document.

Navigating with Bookmarks

By using *bookmarks*, you can mark positions in an edited document so that you can return to it later. This is especially helpful for navigating through large documents or while editing multiple documents. You can place up to nine distinct *bookmarks* in any document. Shortcut keys are available to place the *bookmarks* or to return to any of the marked positions. You can configure these shortcut keys in the ***Options > Menu Shortcut Keys*** menu.

```
1  86  <xsl:template name="customizePageTop">
   87    <xsl:param name="page"/>
   88    <xsl:param name="chapter"/>
   89    <xsl:param name="linksection"/>
   90    <xsl:param name="siteElement"/>
2  91    <xsl:param name="element"/>
   92    <xsl:param name="product"/>
```

Figure 59: Editor Bookmarks

A *bookmark* can be inserted in **Text** mode by doing one of the following:

- Click in the vertical stripe on the left side of the editor (to the left of the line number).
- Select the **Create Bookmark (F9)** action from the **Edit** > **Bookmarks** menu.

A *bookmark* can be removed by right-clicking its icon on the vertical stripe and selecting **Remove** or **Remove all (Ctrl+F7 (Command+F7 on OS X))**.

You can navigate the *bookmarks* by using one of the actions available on the **Edit** > **Bookmarks** > **Go to** menu or by using the shortcut keys that are listed in that menu.

Text Mode Views

There is a variety of *dockable* helper views that are displayed by default in **Text** mode. There are also a large selection of additional views available in the **Window** > **Show View** menu. This section presents some of the most helpful views for editing in **Text** mode.

Project View

The **Project** view is designed to assist you with organizing and managing related files grouped in the same XML project. The actions available in the contextual menu and on the toolbar associated to this panel allow you to create XML projects and provide shortcuts to various operations for the project documents.

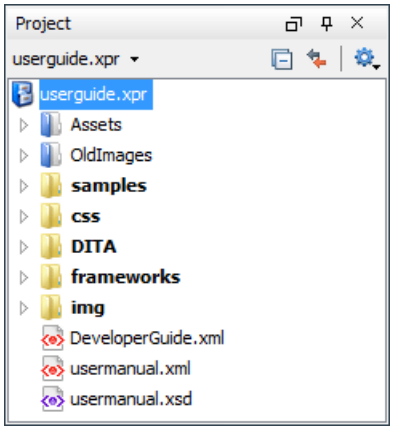


Figure 60: Project View

By default, the view is positioned on the left side of the Oxygen XML Editor window, above the ***Outline*** *view*. If the view has been closed, it can be reopened at any time from the **Window** > **Show View** menu (or using the **Show Project View** action from the **Project** menu).

The tree structure occupies most of the view area. In the upper left side of the view, there is a drop-down menu that contains all recently used projects and project management actions:

Open Project (Ctrl + F2 (Command + F2 on OS X))

Opens an existing project. Alternatively, you can open a project by dropping an Oxygen XML Editor XPR project file from the file explorer into the **Project** panel.

Notice: When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

New Project

Creates a new, empty project.

The following actions are grouped in the upper right corner:

Collapse All

Collapses all project tree folders. You can also collapse/expand a project tree folder if you select it and press the **Enter** key or **Left Arrow** to collapse and **Right Arrow** to expand.

Link with Editor

When selected, the project tree highlights the currently edited file, if it is found in the project files.

Note: This button is disabled automatically when you move to the **Debugger** *perspective*.

Settings

A submenu that contains the following actions:

Filters

Allows you to filter the information displayed in the **Project** view. Click the toolbar button to set filter patterns for the files you want to show or hide. Also, you can set filter patterns for the linked directories that are hidden.

Show Full Path

When selected, linked files and folders are presented with a full file path.

Enable Master Files Support

Select this option to enable the ***Master Files*** *support*.

Change Search and Refactor operations scope

Allows you to change the collection of documents that define the context of the *search and refactor operations*.

- **Use only Master Files, if enabled** - Restricts Oxygen XML Editor to perform the search and refactor operations starting from the *master files* that are defined for the current resource. This option is available when you select **Project** in the **Select the scope for Search and Refactor operations** dialog box and the **Master Files** support is enabled.
- **Working sets** - Allows you to specify the set of files that will be used for the scope of the search and refactor operations.

The files are usually organized in an XML project as a collection of folders. There are three types of resources displayed in the **Project** view:

- **Logical folders** - marked with a blue icon on Windows and Unix/Linux ( ) and a magenta icon on Mac OS X ( ). They help you group files within the project. This folder type has no correspondent on the physical disk, since they are used as containers for related items. Creating and deleting them does not affect the file system on disk. They are created on the project root or inside other logical folders by using the contextual action **New** > **Logical Folder**. The contextual menu action **Remove from Project** can be used to remove them from the project.
- **Physical folders and files** - marked with the operating system-specific icon for folders (usually a yellow icon on Windows and a blue icon on Mac OS X). These folders and files are mirrors of real folders or files that exist in the local file system. They are created or added to the project by using contextual menu actions (such as **New** > **File**, **New** > **Folder**, **Add Folder**, etc.) Also, the contextual menu action **Remove from Disk (Shift +Delete)** can be used to remove them from the project and local file system.

- **Shortcut folders and files** - the icons for file shortcuts include a shortcut symbol and names of folder shortcuts are displayed in bold text. All files and folders that appear as direct descendants of a logical folder are considered shortcuts. They are created and added with the contextual actions **Add Files** and **Add Folder** from the project root. Both contextual menu actions **Remove from Project** and **Remove from Disk (Shift +Delete)** are available for shortcuts. **Remove from Project** just removes the shortcut from the project, while **Remove from Disk (Shift+Delete)** removes the shortcut and the physical resource from the local file system.

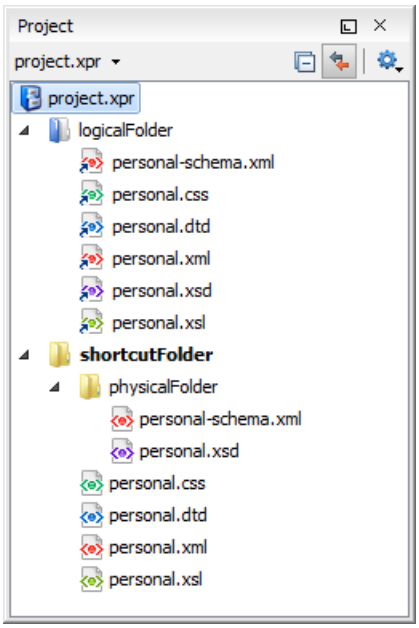


Figure 61: Project View with Examples of all Three Types of Resources

Creating New Projects

The following action is available in the **Project** menu, the **New** menu in the contextual menu, or from the drop-down menu in the top-left of the **Project** view:

New Project

Creates a new, empty project.

Creating New Project Items

The following actions are available by selecting **New** from the contextual menu, when invoked from the **Project** view:

New > File

Opens a ***New*** *file dialog box* that helps you create a new file and adds it to the project structure.

New > Folder

Opens a **New Folder** dialog box that allows you to specify a name for a new folder and adds it to the structure of the project.

New > Logical Folder

Available when invoked from the *project root*, this action creates a logical folder in the tree structure (the icon is a magenta folder on Mac OS X -).

New > Logical Folders from Web

Available when invoked from the *project root*, this action replicates the structure of a remote folder accessible over FTP/SFTP/WebDAV, as a structure of logical folders. The newly created logical folders contain the file structure of the folder it points to.

Managing Physical Folders and Files

You can create physical folders by selecting **New** > **Folder** from the contextual menu.

When adding files to a project, the default target is the project root. To change a target, select a new folder. Files may have multiple instances within the folder system, but cannot appear twice within the same folder.

Removing Files and Folders

To remove one or more linked files or folders, select them in the project tree and press the **Delete** key, or select the contextual menu action **Remove from Project**. To remove a linked file or folder from both project and local file system, select the contextual menu action **Remove from Disk (Shift+Delete)**. The **Remove from Disk (Shift+Delete)** action is also used to remove physical files or folders.

CAUTION: In most cases this action is irreversible, deleting the file permanently. Under particular circumstances (if you are running a Windows installation of Oxygen XML Editor and the *Recycle Bin* is active) the file is moved to *Recycle Bin*.

Moving Files and Folders

You can *move the resources of the project* with drag and drop operations on the files and folders of the tree (the **Enable drag-and-drop in Project view** option must be selected in the ***View*** *preferences page*).

You can also use the usual **Cut**, **Copy**, and **Paste** actions to move resources in the **Project** view.

Renaming Files and Folders

There are three ways you can *rename an item in the* ***Project*** *view*. Select the item in the **Project** view and do one of the following:

- Invoke the **Rename** action from the contextual menu.
- Press **F2** and type the new name.
- Click the selected item and type the new name.

  To finish editing the item name, press **Enter**.

Note: The **Rename** action is also available on *logical* files.

Locating and Opening Files

If a project folder contains a lot of documents, a certain document can be located quickly in the project tree by selecting the folder containing the desired document and typing the first few characters of the document name. The desired document is automatically selected as soon as the typed characters uniquely identify its name in the folder.

The selected document can be opened by pressing the **Enter** key, by double-clicking it, or with one of the **Open** actions from the contextual menu. The files with known document types are opened in the associated editor, while binary files are opened with the associated system application. To open a file with a known document type in an editor other than the default one, use the **Open with** action. Also, dragging and dropping files from the project tree to the editor area results in the files being opened.

Saving the Project

The project file is automatically saved every time the content of the **Project** view is saved or modified by actions such as adding or removing files and drag and drop.

Linked Folders

You can create linked folders (shortcuts) by dragging and dropping folders from a system explorer to the project tree (the **Enable drag-and-drop in Project view** option must be selected in the ***Views*** *preferences page*), or by selecting **Add Folder** in the contextual menu from the project root.

To create a file inside a linked folder, select the **New** > **File** action from the contextual menu. This opens the ***New Document*** *wizard*.

Note: The linked files presented in the **Project** view are marked with a special icon. Linked folders are displayed in bold text.

Logical Folders

The project itself is considered a logical folder. You can add content to a logical folder using one of the actions available in the contextual menu, when invoked from the *project root*:

Add Folder

Adds a link to a physical folder, whose name and content mirror a real folder that exists in the local file system (the icon of this action is different on Mac OS X).

Add Files

Adds links to files on the local file system.

Add Edited File

Adds a link to the currently edited file in the project.

Validate Files

The currently selected files in the **Project** view can be checked to be XML well-formed or validated against a schema (DTD, XML Schema, Relax NG, Schematron or NVDL) with one of the following contextual menu actions found in the **Validate** submenu:

Check Well-Formedness

Checks if the selected file or files are well-formed.

Validate

Validates the selected file or files against their associated schema. EPUB files make an exception, because this action triggers a ***Validate and Check for Completeness*** operation.

Validate with Schema

Validates the selected file of files against a specified schema.

Configure Validation Scenario(s)

Allows you to configure and run a *validation scenario*.

Applying Transformation Scenarios

The currently selected files in the **Project** view can be transformed in one step with one of the following actions available from contextual menu in the **Transform** submenu:

Apply Transformation Scenario(s)

Obtains the output with one of the *built-in scenarios*.

Configure Transformation Scenario(s)

Opens a *dialog box* that allows you to configure pre-defined transformation scenarios.

Transform with

Allows you to select a transformation scenario to be applied to the currently selected files.

Along with the logical folder support, this allows you to group your files and transform them very easily.

Refactoring Actions (Available for certain document types (such as XML, XSD, and XSL)

Oxygen XML Editor includes some refactoring operations that help you manage the structure of your documents. The following actions are available from the contextual menu in the **Refactoring** submenu:

Rename resource

Allows you to change the name of a resource.

Move resource

Allows you to change the location on disk of a resource.

XML Refactoring

Opens *the **XML Refactoring** tool wizard* that presents refactoring operations to assist you with managing the structure of your XML documents.

Other XML Refactoring Actions

For your convenience, the last 5 *XML Refactoring tool operations* that were finished or previewed will also appear in this submenu.

Other Contextual Menu Actions

Other actions that are available in the contextual menu from the project tree include:

Open

Opens the selected files in the corresponding editor.

Open with submenu

This submenu allows you to open the selected file with the internal editor, a system application, or other internal tools: ***DITA Maps Manager***, ***Archive Browser***, *MathML Editor*, *Generate/Convert Schema*, ***WSDL/SOAP Analyzer***, *Large File Viewer*, *Hex Viewer*, *SVG Viewer*.

Show in Explorer (or Show in Finder on OS X)

In Windows, the content of the selected folder or file is presented in a specific explorer window. On MAC OS X, the parent folder is opened and the selected folder is highlighted in a specific finder window.

Copy Location

Copies an application-specific URL for the selected resource to the clipboard.

Refresh

Refreshes the content and the dependencies between the resources in *the **Master Files** directory*.

Find/Replace in Files

Opens the ***Find/Replace in Files** dialog box* that allows you to find and replace text in multiple files.

XPath in Files

Opens the ***XPath/XQuery Builder** view* that allows you to compose XPath and XQuery expressions and execute them over the currently edited XML document.

Open/Find Resource

Opens the ***Open/Find Resource** dialog box*.

Check Spelling in Files

Allows you to *check the spelling of multiple files.*

Format and Indent Files

Opens the ***Format and Indent Files** dialog box* that allows you to configure the format and indent (*pretty-print*) action that will be applied on the selected documents.

Open in SVN Client

Syncro SVN Client tool is opened and it highlights the selected resource in its corresponding working copy.

Compare

Allows you to compare multiple files or directories and the order of your selection determines where they are opened in the ***Compare Files*** or ***Compare Directories*** tool. If you select two files or folders, your first selection will be opened in the left panel and the other one in the right panel.

You can also select 3 files and the tool will automatically be opened in the *three-way comparison mode*. If you select three files, your first selection will be opened in the left panel, the second in the right panel, and the third selection will be the base (ancestor) file.

Generate Documentation > XML Schema Documentation

Opens *the XML Schema Documentation Dialog Box*.

Generate Documentation > XSLT Stylesheet Documentation

Opens *the XSLT Stylesheet Documentation Dialog Box*.

Generate Documentation > XQuery Documentation

Opens *the XQuery Documentation Dialog Box*.

Generate Documentation > WSDL Documentation

Opens *the WSDL Documentation Dialog Box*.

Properties

Displays the properties of the current file in a **Properties** dialog box.

Menu Level Actions

The following actions are available in the **Project** menu:

New Project

Creates a new, empty project.

Open Project (Ctrl + F2 (Command + F2 on OS X))

Opens an existing project. Alternatively, you can open a project by dropping an Oxygen XML Editor XPR project file from the file explorer into the **Project** panel.

Notice: When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

Save Project As

Allows you to save the current project under a different name.

Validate all project files

Checks if the project files are well-formed and their mark-up conforms with the specified DTD, XML Schema, or Relax NG schema rules. It returns an error list in the message panel.

Filters

Opens the **Project filters** dialog box that allows you to decide which files and directories will be shown or hidden.

Enable Master Files Support

Allows you to enable the ***Master Files Support*** for each project you are working on.

Change Search and Refactor operations scope

Opens a dialog box that allows you to define the context of search and refactor operations.

Show Project View

Displays the **Project** view.

Reopen Project

Contains a list of links of previously used projects. This list can be emptied by invoking the **Clear history** action.

Open/Find Resource View

The **Open/Find Resource** view is designed to offer advanced search capabilities either by using a simple text search or by using the *Apache Lucene - Query Parser Syntax*. By default, the view is presented in the left side of the Oxygen XML Editor layout, next to the ***Project** view* or ***DITA Maps Manager***. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

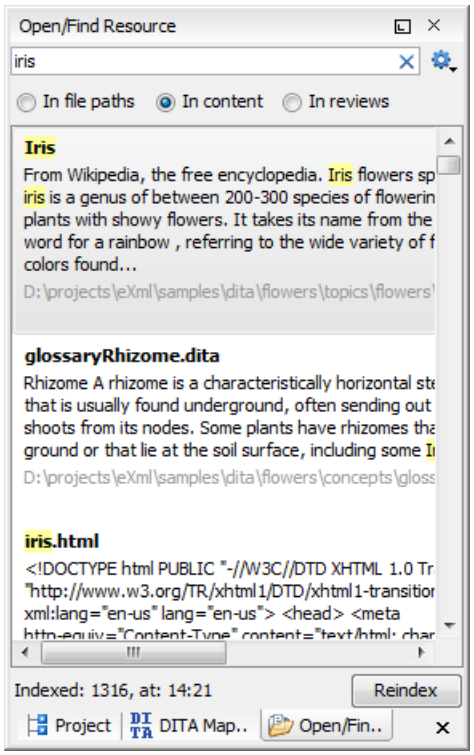


Figure 62: Open/Find Resource View

You can use this view to find a file in the current Oxygen XML Editor project or in one of the *DITA maps* opened in *the **DITA Maps Manager** view* by typing only a few letters of the file name of a document or a fragment of the content you are searching for. The **Open/Find Resource** view also supports searching in document edits (comments, tracked change insertions/deletions, and highlighted content) by selecting the ***In reviews** option*.

Note: Full support for searching in document edits (the **In reviews** option) is available only in the Enterprise edition of Oxygen XML Editor. The Professional edition offers limited support to search through a maximum of 10 edits.

Search Results

Search results are presented instantly, after you finish typing the content. The matching fragments of text are highlighted in the results list displayed in the dialog box. When you open one of the documents from the results list, the matching fragments of text are highlighted in the editing area. To remove the highlighting from your document, close the corresponding tab in the ***Results** view* at the bottom of the editor. To display the search history, position the cursor in the search field and press **Ctrl + DownArrow (Command + DownArrow on OS X)** or **Ctrl + UpArrow (Command + UpArrow on OS X)** on your keyboard. Pressing only the **DownArrow** key moves the selection to the list of results.

Note: Searches are not case sensitive. For example, if you search for `car` you get the same results as when you search for `Car`.

Tip: Suffix searches are also supported, both for searching in the content of your resources and in their name. For this, you can use wildcards. If you search for `*ing` with the **in content** option selected, you will find documents that contain the word *presenting*. If you search for `*/samples/*.gif` with the **in file paths** option selected, you will find all the *gif* images from the `samples` directory.

Options Available in the View

The **Open/Find Resource** view offers the following options:

- **Settings** - Drop-down menu that includes the following settings for the view:
  - **Clear Index** - Clears the index.

- **Show description** - Presents the search results in a more compact form, displaying only the title and the location of the resources.
- **Options** - Opens the ***Open/Find Resource*** *preferences page* where you can configure various search options. For example, you can specify a **Content language** that differs from the default UI language in case your document contains multiple languages.

- ***In file paths*** - Select this option to search for resources by their name or by its path (or a fragment of its path).
- ***In content*** - Select this option to search through the content of your resources.
- ***In reviews*** - Select this option to search through the comments, *tracked change* insertions/deletions, or highlights in your resources.
- **Reindex** - Use this option to reindex your resources.

Contextual Menu Actions

A contextual menu is available on each search result and provides actions applicable to that particular document. These actions include:

- **Open** - Opens the document in one of Oxygen XML Editor internal editors.
- **Open with** - Allows you to choose to open the document in the **Internal editor** or an external **System application**.
- **Show in Explorer** - Identifies the document in the system file explorer.
- **Copy Location** - Copies the file path and places it in the clipboard.

Indexing Process

The content of the resources used to search in is parsed from an index. The indexing is performed both automatically and on request. Automatic indexing is performed when you modify, add, or remove resources in the currently indexed project. If the index was never initialized, the index in not updated on project changes.

To improve performance, the indexing process skips the following set of common English words (the so-called **stop words**): *a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with*. This means that if you are searching for any of these words, the indexing process will not be able to match any of them. However, you can configure the list of **stop words** in the ***Open/Find Resource*** *preferences page*.

Caching Mechanism

When you perform a search, a caching mechanism is used to gather the paths of all files linked in the current project. When the first search is performed, all project files are indexed and added to the cache. The next search operation uses the information extracted from the cache, thus improving the processing time. The cache is kept for the currently loaded project only, so when you perform a search in a new project, the cache is rewritten. Also, the cache is reset when you press the **Reindex** button.

Important: Files larger than 2GB are not indexed.

If there is no file found that matches your file pattern or text search, a possible cause is that the file you are searching for was added to the Oxygen XML Editor project after the last caching operation. In this case, re-indexing the project files with the **Reindex** button enables the file to be found. The date and time of the last index operation are displayed below the file list.

Opening the Results

Once you find the files that you want to open, select them in the list and press the **Open** button (or double-click them). Each of the selected files is opened in *the editor associated with the type of the file*.

Note: You can drag a resource from the **Open/Find Resource** view and drop it in a DocBook, DITA, TEI or XHTML document to create a link to that resource.

To watch our video demonstration about the **Open/Find Resource** feature and its search capabilities, go to *https://www.oxygenxml.com/demo/Open_Find_Resource.html*.

Related Information:

Open/Find Resource Dialog Box on page 271

Outline View in Text Mode

The **Outline** view in **Text** mode displays a general tag overview of the currently edited XML Document. When you edit a document, the **Outline** view dynamically follows the changes that you make, displaying the node that you modify. This functionality gives you great insight on the location of your modifications in the current document. It also shows the correct hierarchical dependencies between elements. This makes it easy for you to be aware of the document structure and the way element tags are nested.

Outline View Features

The **Outline** view allows you to:

- Quickly navigate through the document by selecting nodes in the **Outline** tree.
- Insert or delete nodes using contextual menu actions.
- Move elements by dragging them to a new position in the tree structure.
- Highlight elements in the editor area. It is synchronized with the editor area, so when you make a selection in the editor area, the corresponding nodes are highlighted in the **Outline** view, and vice versa.
- View document errors, as they are highlighted in the **Outline** view. A tooltip also provides more information about the nature of the error when you hover over the faulted element.

Outline View Interface

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (*, ?) and separate multiple patterns with commas.

It also includes a **Settings** menu in the top-right corner that presents a variety of options to help you filter the view even further.

Drop and Drop Actions in the Outline View

Entire XML elements can be moved or copied in the edited document using only the mouse in the **Outline** view with drag-and-drop operations. Several drag and drop actions are possible:

- If you drag an XML element in the **Outline** view and drop it on another node, then the dragged element will be moved after the drop target element.
- If you hold the mouse pointer over the drop target for a short time before the drop then the drop target element will be expanded first and the dragged element will be moved inside the drop target element after its opening tag.
- You can also drop an element before or after another element if you hold the mouse pointer towards the upper or lower part of the targeted element. A marker will indicate whether the drop will be performed before or after the target element.
- If you hold down the **(Ctrl (Command on OS X))** key after dragging, a copy operation will be performed instead of a move.

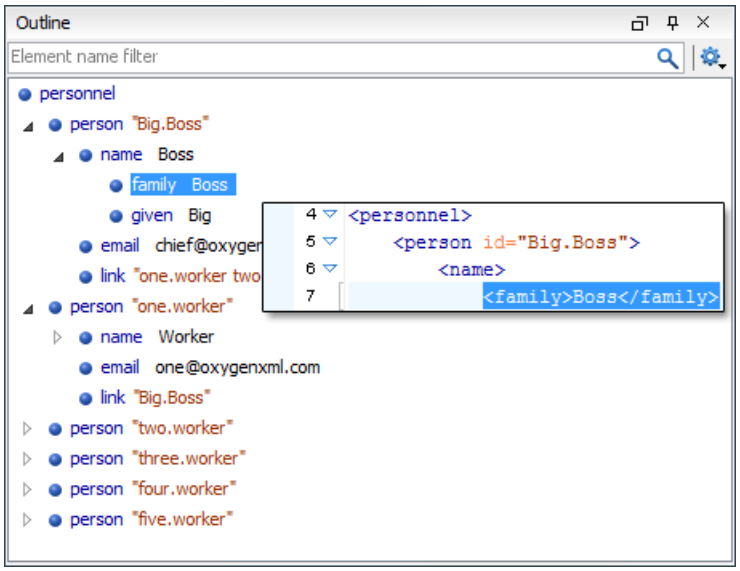


Figure 63: Outline View in Text Mode

Related Information:

Outline View in Author Mode on page 222

Outline View Filters in Text Mode

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (*, ?) and separate multiple patterns with commas.

The following actions are available in the **Settings** menu of the **Outline** view when editing in **Text** mode:

Filter returns exact matches

The text filter of the **Outline** view returns only exact matches.

Selection update on cursor move

Controls the synchronization between **Outline** view and source document. The selection in the **Outline** view can be synchronized with the cursor moves or the changes in the editor. Selecting one of the components from the **Outline** view also selects the corresponding item in the source document.

Flat presentation mode of the filtered results

When active, the application flattens the filtered result elements to a single level.

Show comments and processing instructions

Show/hide comments and processing instructions in the **Outline** view.

Show element name

Show/hide element name.

Show text

Show/hide additional text content for the displayed elements.

Show attributes

Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from *the Outline preferences panel*.

Configure displayed attributes

Displays the ***XML Structured Outline*** *preferences page*.

Outline View Contextual Menu Actions in Text Mode

The following actions are available from the contextual menu in the **Outline** view in **Text** mode:

Append Child

Allows you to select an element (from a drop-down list) that is allowed by the associated schema and inserts it as a child of the current element.

Insert Before

Allows you to select an element (from a drop-down list) that is allowed by the associated schema and inserts it immediately before the current element, as a sibling.

Insert After

Allows you to select an element (from a drop-down list) that is allowed by the associated schema and inserts it immediately after the current element, as a sibling.

Edit Attributes

Opens a dialog box that allows you to edit the attributes of the currently selected component.

Toggle Comment

Encloses the currently selected element in an XML comment, if the element is not already commented. If it is already commented, this action will remove the comment.

Cut

Cuts the currently selected component.

Copy

Copies the currently selected component.

Delete

Deletes the currently selected component.

Expand More

Expands the structure of a component in the **Outline** view.

Collapse All

Collapses the structure of all the component in the **Outline** view.

Attributes View in Text Mode

The **Attributes** view presents all the attributes of the current element determined by the schema of the document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened from the **Window** > **Show View** menu.

You can use the **Attributes** view to insert attributes, edit their values, or add values to existing attributes.

The attributes are rendered differently depending on their state:

- The names of the attributes are rendered with a bold font, and their values with a plain font.
- Default values are rendered with a plain font, painted gray.
- Empty values display the text "[empty]", painted gray.
- Invalid attributes and values are painted red.

To edit the value of the corresponding attribute, double-click a cell in the **Value** column . If the possible values of the attribute are specified as `list` in the schema of the edited document, the **Value** column acts as a combo box that allows you to either select the value from a list or manually enter it.

You can sort the attributes table by clicking the **Attribute** column header. The table contents can be sorted as follows:

- By attribute name in ascending order.
- By attribute name in descending order.
- Custom order, where the used attributes are displayed at the beginning of the table sorted in ascending order, followed by the rest of the allowed elements sorted in ascending order.

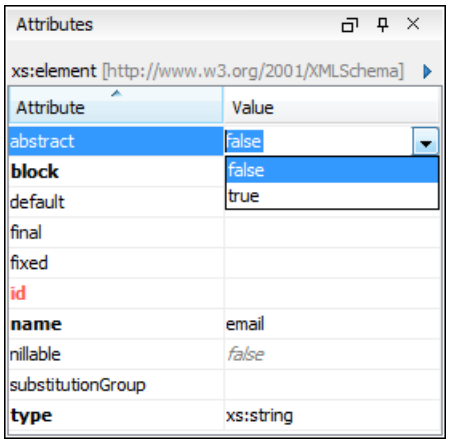


Figure 64: Attributes View

Expand/Collapse Button

There is an **Expand/** **Collapse** button at the top-right of the view. When expanded, this presents the following additional combo boxes:

Name Combo Box

Use this combo box to select an attribute. The drop-down list displays the list of possible attributes allowed by the schema of the document, as in the **Attributes** view. You can use the **Remove** button to delete an attribute and its value from the selected element.

Value Combo Box

Use this combo box to add, edit, or select the value of an attribute. If the selected attribute has predefined values in the schema, the drop-down list displays those possible values. You can use the **Browse** button to select a URL for the value of an attribute. After you have entered or selected a value, use the **Update** button (or press **Enter**) to add the value to the attribute.

Contextual Menu Actions in the Attributes View

The following actions are available in the contextual menu of the **Attributes** view when editing in **Text** mode:

Add

Allows you to insert a new attribute. Adding an attribute that is not in the list of all defined attributes is not possible when the *Allow only insertion of valid elements and attributes* schema-aware option is selected.

Set empty value

Specifies the current attribute value as empty.

Remove

Removes the attribute (action available only if the attribute is specified). You can invoke this action by pressing the **(Delete)** or **(Backspace)** keys.

Copy

Copies the `attrName="attrValue"` pair to the clipboard. The `attrValue` can be:

- The value of the attribute.
- The value of the default attribute, if the attribute does not appear in the edited document.
- Empty, if the attribute does not appear in the edited document and has no default value set.

Paste

Depending on the content of the clipboard, the following cases are possible:

- If the clipboard contains an attribute and its value, both of them are introduced in the **Attributes** view. The attribute is selected and its value is changed if they exist in the **Attributes** view.

- If the clipboard contains an attribute name with an empty value, the attribute is introduced in the **Attributes** view and you can start editing it. The attribute is selected and you can start editing it if it exists in the **Attributes** view.
- If the clipboard only contains text, the value of the selected attribute is modified.

Model View

The **Model** view presents the structure of the currently selected tag, and its documentation, defined as annotation in the schema of the current document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

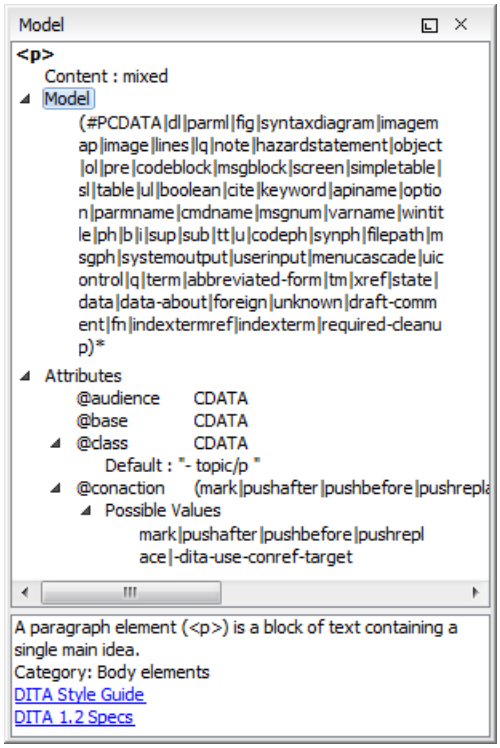


Figure 65: Model View

The **Model** view is comprised of two sections, an element structure panel and an annotations panel.

Element Structure Panel

The element structure panel displays the structure of the currently edited or selected tag in a tree-like format. The information includes the name, model, and attributes of the current tag. The allowed attributes are shown along with imposed restrictions, if any.

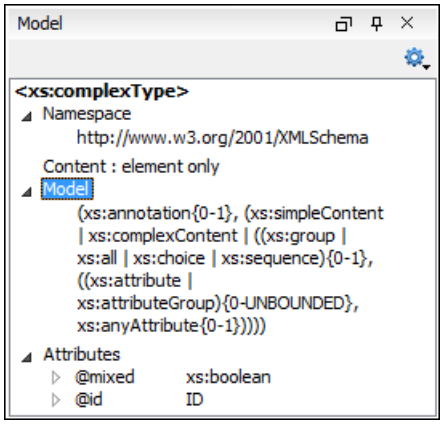


Figure 66: Element Structure Panel

Annotation Panel

The **Annotation** panel displays the annotation information for the currently selected element. This information is collected from the XML schema.

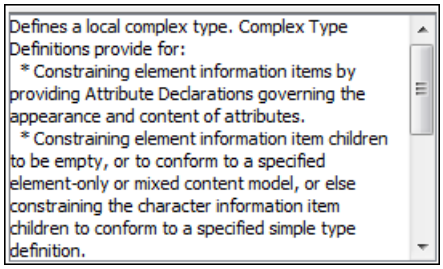


Figure 67: Annotation panel

Elements View in Text Mode

The **Elements** view presents a list of all defined elements that are valid at the current cursor position according to the schema associated to the document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Double-clicking any of the listed elements inserts that element into the edited document, at the current cursor position.

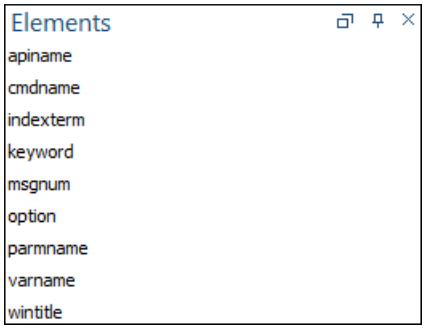


Figure 68: Elements View in Text Mode

Entities View

Entities provide abbreviated entries that can be used in XML files when there is a need of repeatedly inserting certain characters or large blocks of information. An *entity* is defined using the `ENTITY` statement either in the DOCTYPE declaration or in a DTD file associated with the current XML file.

There are three types of entities:

- **Built-in** or **Predefined** - Entities that are part of the predefined XML markup (`<`, `>`, `&`, `'`, `"`).
- **Internal** - Defined in the DOCTYPE declaration header of the current XML.
- **External** - Defined in an external DTD module included in the DTD referenced in the XML DOCTYPE declaration.

Note: If you want to add internal entities, you would need to switch to the Text editing mode and manually modify the DOCTYPE declaration. If you want to add external entities, you need to open the DTD module file and modify it directly.

The **Entities** view displays a list with all entities declared in the current document, as well as built-in ones. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Double-clicking one of the entities will insert it at the current cursor position in the XML document. You can also sort entities by name and value by clicking the column headers.

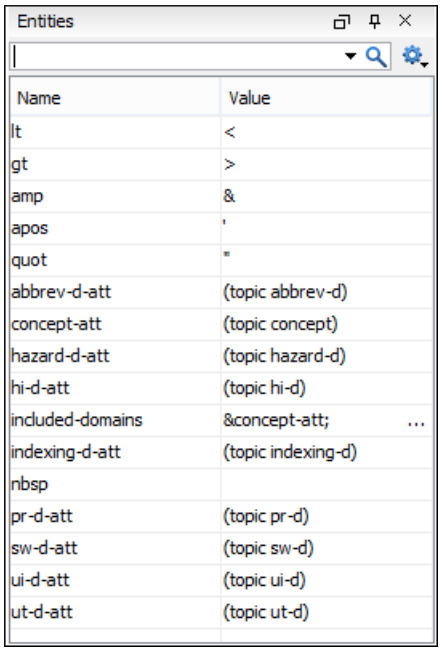


Figure 69: Entities View

The view features a filtering capability that allows you to search an entity by name, value, or both. Also, you can choose to display the internal or external entities.

Note: When entering filters, you can use the ? and * wildcards. Also, you can enter multiple filters by separating them with a comma.

Results View

The **Results** view displays the messages generated as a result of user actions such as validations, transformations, search operations, and others. Each message is a link to the location related to the event that triggered the message. Double-clicking a message opens the file containing the location and positions the cursor at the location offset. The **Results** view is automatically opened when certain actions generate result messages. Those actions include the following:

- ***Validation*** *actions*
- ***Transformation*** *actions*
- ***Check Spelling in Files*** *action*
- **Find All** action from the ***Find/Replace*** *dialog box*
- ***Find/Replace in Files*** *dialog box*
- ***Search References*** *action*
- *XPath expression results*
- *SQL results*

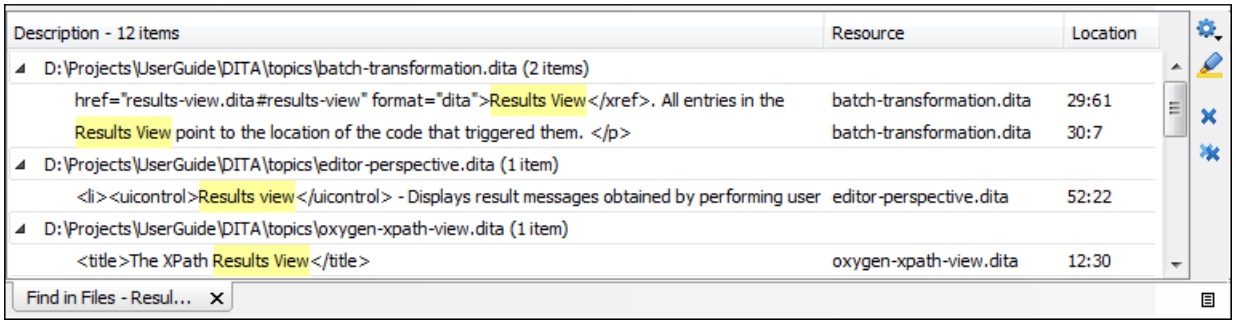


Figure 70: Results View

Results View Toolbar Actions

The view includes a toolbar with the following actions:

Grouping options drop-down menu

A set of **Group by** toggle actions that allow you to group the messages according to a selected criteria so that they can be presented in a hierarchical layout. The criteria used for grouping can be the severity of the errors (error, warning, info message, etc.), the resource name, the description of the message, and so on.

This drop-down menu also includes the following additional grouping actions:

Ungroup all

Removes the grouping rules so that the messages are presented in a continuous list.

Show group columns

If any of the **Group by** options are selected, you can use this option to show or hide grouping columns.

Restore Defaults

Restores the column size for each column and the grouping rules that were saved in the user preferences the last time when this view was used. If it is the first time this view is used, the action sets an initial default column size for each column and a grouping rule that is appropriate for the type of messages. For example:

- Group the messages by the path of the validated file if there are error messages from a validation action or spelling errors reported by the ***Check Spelling in Files*** *action*.
- No grouping rule for the results of *applying an XPath expression*.

Highlight all results in editor

Oxygen XML Editor highlights all matches obtained after executing an XPath expression, or performing one of the following operations: **Find All**, **Find in Files**, **Search References**, and **Search Declarations**. Click **Highlight all results in editor** again to turn off highlighting.

Note: To customize highlighting behavior, *open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Editor** > **Highlights category**. You can do the following customizations:

- Set a specific color of the highlights depending on the type of action you make.
- Set a maximum number of highlights that the application displays at any given time.

Remove selected

Removes the current selection from the view. This can be helpful if you want to reduce the number of messages or remove those that have already been addressed or not relevant to your task.

Remove all

Removes all messages from the view.

Results View Contextual Menu Actions

The following actions are available when the contextual menu is invoked in this view:

Show message

Displays a dialog box with the full error message, which is useful for a long message that does not have enough room to be displayed completely in the view.

Previous message

Navigates to the message above the current selection.

Next message

Navigates to the message below the current selection.

Remove selected

Removes selected messages from the view.

Remove all

Removes all messages from the view.

Copy

Copies information associated with the selected messages. For example:

- The file path of the document that triggered the output message.
- The path of the *master file* (in the case of a *validation scenario*, it is the path of the file from which the validation starts and can be different from the validated file).
- Error severity (error, warning, info message, etc.)
- Name of validating processor.
- Name of *validation scenario*.
- The line and column in the file that triggered the message.

Select All

Extends the selection to all the messages from the view.

Print Results

Sends the complete list of messages to a printer. For each message, the included details are the same as the ones for the ***Copy*** *action*.

Save Results

Saves the complete list of messages in a file in text format. For each message, the included details are the same as the ones for the ***Copy*** *action*.

Save Results as XML

Saves the complete list of messages in a file in XML format. For each message, the included details are the same as the ones for the ***Copy*** *action*.

Save Results as HTML

Saves the complete list of messages in a file in HTML format. For each message, the included details are the same as the ones for the ***Copy*** *action*.

Group by

A set of **Group by** toggle actions that allow you to group the messages according to a selected criteria so that they can be presented in a hierarchical layout. The criteria used for grouping can be the severity of the errors (error, warning, info message, etc.), the resource name, the description of the message, and so on.

Ungroup all

Removes the grouping rules so that the messages are presented in a continuous list.

Show group columns

If any of the **Group by** options are selected, you can use this option to show or hide grouping columns.

Restore Defaults

Restores the column size for each column and the grouping rules that were saved in the user preferences the last time when this view was used. If it is the first time this view is used, the action sets an initial default column size for each column and a grouping rule that is appropriate for the type of messages. For example:

- Group the messages by the path of the validated file if there are error messages from a validation action or spelling errors reported by the ***Check Spelling in Files*** *action*.
- No grouping rule for the results of *applying an XPath expression*.

Expand All

Expands all the nodes of the tree, which is useful when the messages are presented in a hierarchical mode.

Collapse All

Collapses all the nodes of the tree, which is useful when the messages are presented in a hierarchical mode.

Syntax Highlight Depending on Namespace Prefix

The *syntax highlight scheme of an XML file type* allows the configuration of a color per each type of token that can appear in an XML file. Distinguishing between the XML tag tokens based on the namespace prefix brings additional visual help in editing some XML file types. For example, in XSLT stylesheets, elements from various namespaces (such as XSLT, XHTML, XSL:FO, or XForms) are inserted in the same document and the editor panel can become cluttered. *Marking tags with different colors based on the namespace prefix* allows easier identification of the tags.

```
3   <xsl:template match="name">
4     <fo:list-item>
5       <fo:list-item-label end-indent="label-end()">
6         <fo:block text-align="end" font-weight="bold">Full Name</fo:block>
7       </fo:list-item-label>
8       <fo:list-item-body start-indent="body-start()">
9         <xsl:apply-templates select="*"/>
10      </fo:list-item-body>
11    </fo:list-item>
12  </xsl:template>
```

Figure 71: Example of Coloring XML Tags by Prefix

Related Information:

Changing the colors displayed in the Text Mode Editor on page 104

Presenting Validation Errors in Text Mode

Oxygen XML Editor can be configured to *automatically validate documents* while editing in the **Text** mode, and actions are also available to *manually validate documents* on-request.

In **Text** mode, validation issues are marked in the following locations:

- In the main editing pane, with the issue underlined in a color according to the type of issue.
- In the right-side vertical stripe, with a marker that is colored according to the type of issue.
- For attributes with detected issues, in the ***Attributes** view*, with the attribute and its value colored according to the type of issue.

The colors for each type of issue are as follows:

- **Validation Errors** [Red] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the **Attributes** view are colored in red.
- **Validation Warnings** [Yellow] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the **Attributes** view are colored in yellow.
- **Validation Info** [Blue] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the **Attributes** view are colored in blue.

You can configure the color for each type in the ***Document Checking** preferences page*.

Hovering over a validation issue presents a tooltip message with more details about the problem and *possible quick fixes* (if available for that issue).

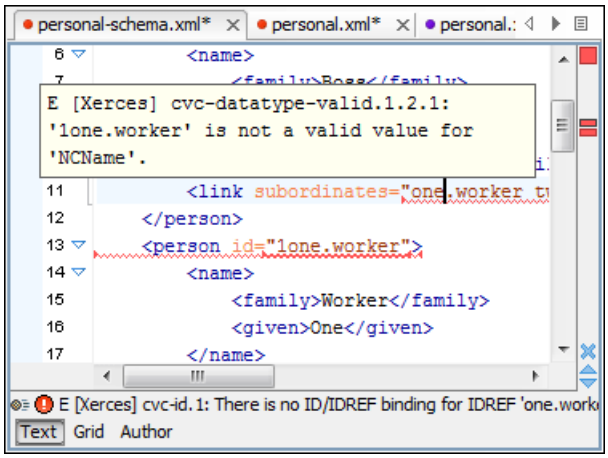


Figure 72: Presenting Validation Errors in Text Mode

Also, the stripe on the right side of the editor panel is designed to display the issues found during the validation process and to help you locate them in the document. The stripe contains the following:

Upper Part of the Stripe

A success indicator square will turn green if the validation is successful or only info messages are found, red if validation errors are found, or yellow if only validation warnings are found. More details about the issues are displayed in a tool tip when you hover over indicator square. If there are numerous problems, only the first three are presented in the tool tip.

Middle Part of the Stripe

Errors are depicted with red markers, warnings with yellow markers, and info message with blue markers. If you want to limit the number of markers that are displayed, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Editor** > **Document checking**, and specify the desired limit in the ***Maximum number of validation highlights** option*.

Clicking a marker will highlight the corresponding text area in the editor. The validation message is also displayed both in a tool tip (when hovering over the marker) and in the message area on the bottom of the editor panel (clicking the **Document checking options** button opens the ***Document Checking** preferences page*.

Bottom Part of the Stripe

Two navigation arrows ( ) allow you to jump to the next or previous issue. The same actions can be triggered from **Document** > **Automatic validation** > **Next error** (**Ctrl + Period (Command + Period on OS X)**) and **Document** > **Automatic validation** > **Previous error** (**Ctrl + Comma (Command + Comma on OS X)**). Also, the button can be used to clear all the validation markers.

Status messages from every validation action are also logged in the ***Information** view*.

If you want to see all the validation messages grouped in the ***Results** panel*, you should use the **Validate** action from the toolbar or **Document** > **Validate** menu..

Related Information:

Validating XML Documents Against a Schema on page 446

Presenting Schematron Validation Issues

Bidirectional Text Support in Text Mode

Bidirectional documents contain text in both directions, usually involving characters from unique types of alphabets.

If bidirectional text (such as Arabic or Hebrew languages), certain Asian languages (such as Devanagari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telugu, Kannada, Malayalam, Sinhala, Thai, Khmer), or other special characters (such as combining characters) are detected in a document, Oxygen XML Editor displays a **Special Characters Detected** dialog box that prompts you to **Enable** or **Disable** support for these special characters.

You can also configure the support for bidirectional text in the **Open/Save** preferences page. To enable or disable this support, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Editor** > **Open/Save**, and choose the appropriate setting in the ***Support for Special Characters** section.*

Note: Disabling this support may affect text rendering, cursor positioning and navigation, as well as text selection and management operations. If you need to open *very large documents*, the bidirectional editing support can also be disabled to enhance performance while editing.

Restriction: Bidirectional content in the **Text** mode cannot be rendered using **Bold** or **Italic**.

Related Information:

Bidirectional Text Support in Grid Mode on page 210

Inserting Symbols on page 250

Grid Editing Mode

The Oxygen XML Editor **Grid** editing mode displays the XML document as a structured grid of nested tables where the text content can be modified without directly interacting with the XML markup. This is helpful for non-technical users who want to edit text content without modifying the XML markup. You can easily expand or collapse elements within the table and the document structure can be changed with simple drag/drop or copy/paste operations.

To switch to this mode, select **Grid** at the bottom of the editing area.

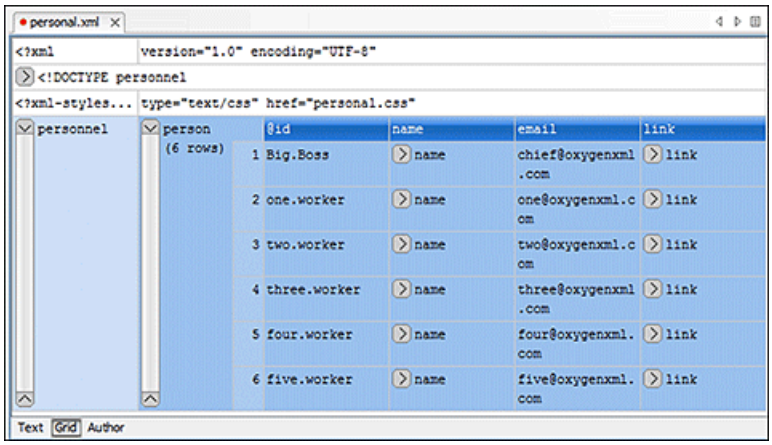


Figure 73: Grid Editing Mode

To watch our video demonstration about some of the features available in the **Grid** editor, go to *https://www.oxygenxml.com/demo/Grid_Editor.html*.

Related Information:

Editing XML Documents in Grid Mode on page 325

Layouts: Grid and Tree

The **Grid** editor offers two layout modes. The default one is the grid layout. This smart layout detects the recurring elements in the XML document and creates tables having the children (including the attributes) of these elements as columns. This way, it is possible to have tables nested in other tables, reflecting the structure of your document.

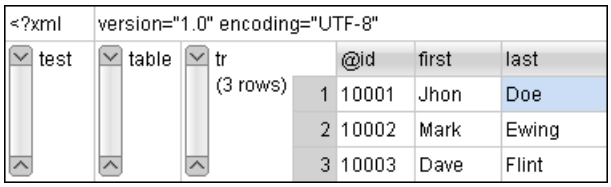


Figure 74: Grid Layout

The other layout mode is tree-like. It does not create any tables and it only presents the structure of the document.

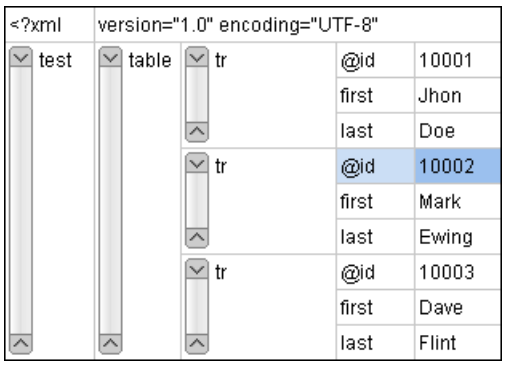


Figure 75: Tree Layout

To switch between the two modes, go to **Document** > **Grid Layout** > **Grid mode/Tree mode**.

Grid Mode Navigation

At first, the content of a document opened in the **Grid** mode is collapsed. Only the root element and its attributes are displayed. The grid disposition of the node names and values is similar to a web form or dialog box. The same set of key shortcuts used to select dialog box components is also available in the **Grid** mode:

Table 5: Shortcuts in the Grid Mode

Key	Action
Tab	Moves the cursor to the next editable value in a table row.
Shift + Tab	Moves the cursor to the previous editable value in a table row.
Enter	Begins editing and lets you insert a new value. Also commits the changes after you finish editing.
UpArrow/PageUp	Navigates toward the beginning of the document.
DownArrow/PageDown	Navigates toward the end of the document.
Shift	Used in conjunction with the navigation keys to create a continuous selection area.
Ctrl (Command on OS X) key	Used in conjunction with the mouse cursor to create discontinuous selection areas.

The following key combinations can be used to scroll the grid:

- **Ctrl + UpArrow (Command + UpArrow on OS X)** - scrolls the grid upwards.
- **Ctrl + DownArrow (Command + DownArrow on OS X)** - scrolls the grid downwards.
- **Ctrl + LeftArrow (Command + LeftArrow on OS X)** scrolls the grid to the left.
- **Ctrl + RightArrow (Command + RightArrow on OS X)** scrolls the grid to the right.

An arrow sign displayed at the left of the node name indicates that this node has child nodes. To display the children, click this sign. The expand/collapse actions can be invoked either with the **NumPad+** and **NumPad-** keys, or from the **Expand/Collapse** submenu of the contextual menu or from **Document** > **Grid Expand/Collapse**.

The following actions are available on the **Expand/Collapse** menu:

Expand All

Expands the selection and all its children.

Collapse All

Collapses the selection and all its children.

Expand Children

Expands all the children of the selection but not the selection.

Collapse Children

Collapses all the children of the selection but not the selection.

Collapse Others

Collapses all the siblings of the current selection but not the selection.

Bidirectional Text Support in Grid Mode

If you are editing documents with a bidirectional text orientation, you can change the way the text is rendered and edited in the grid cells by using the **Change Text Orientation**(**Ctrl + Shift + O (Command + Shift + O on OS X)**) action that is available from the **Edit** menu in the **Grid** editing mode. Use this action to switch from the default left to right text orientation to the right to left orientation, and vice versa.

Note: This change applies only to the text from the cells, and not to the layout of the grid editor.

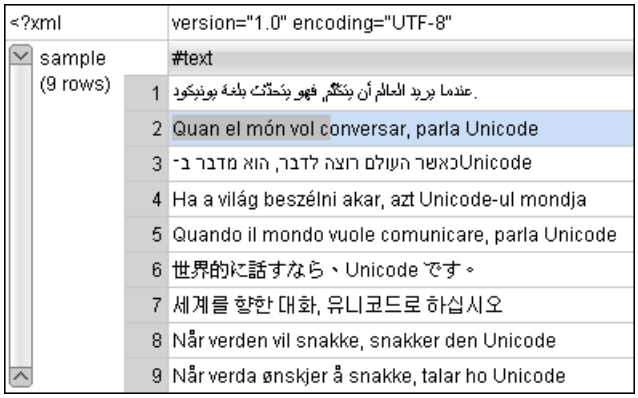


Figure 76: Default left to right text orientation

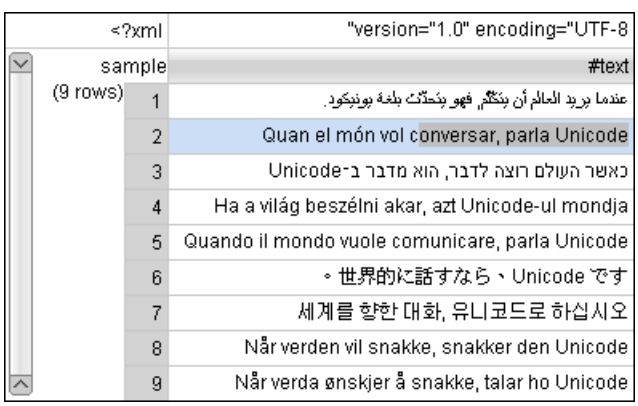


Figure 77: Right to left text orientation

Related Information:

Bidirectional Text Support in Text Mode on page 207

Bidirectional Text Support in Author Mode on page 238
Inserting Symbols

Author Editing Mode

The **Author** editing mode in Oxygen XML Editor allows you to visually edit XML documents in a user-friendly interface that is similar to a WYSIWYG word processor. Oxygen XML Editor provides support for visually editing the most commonly used XML vocabularies in **Author** mode, including DITA, DocBook, TEI, and XHTML. Adding text content is as simple as doing so in a standard text editor but the content is rendered similar to how you will see it in the output. Tables, images, and media objects (such as videos) are also rendered comparable to the output.

To switch to this mode, click the **Author** button at the bottom of the editing area.

To watch our video demonstration about some of the features available in the visual **Author** editing mode, go to *https://www.oxygenxml.com/demo/WYSIWYG_XML_Editing.html*.

Related Information:

Editing XML Documents in Author Mode on page 329

Navigating the Document Content in Author Mode

Oxygen XML Editor includes some useful features to help you navigate XML documents.

Using the Keyboard

Oxygen XML Editor allows you to quickly navigate through a document using the **Tab** key to move the cursor to the next XML node and **Shift + Tab** to go to the previous one. If you encounter a *space-preserved element* when you navigate through a document and you do not press another key, pressing the **Tab** key will continue the navigation. However, if the cursor is positioned in a *space-preserved element* and you press another key or you position the cursor inside such an element using the mouse, the **Tab** key can be used to arrange the text.

To navigate one word forward or backwards, use **Ctrl + RightArrow (Command + RightArrow on OS X)**, and **Ctrl + LeftArrow (Command + LeftArrow on OS X)**, respectively. Entities and hidden elements are skipped. To position the cursor at the beginning or at the end of the document you can use **Ctrl + Home (Command + Home on OS X)**, and **Ctrl + End (Command + End on OS X)**, respectively.

Navigation Buttons

Oxygen XML Editor includes some actions that help you to quickly navigate to a particular modification. These navigation buttons are available in the main toolbar and the actions can also be accessed from the **Find** menu. The three actions include:

- **Last Modification** - Moves the cursor to the last modification in any opened document.
- **Back** - Moves the cursor to the previous position.
- **Forward** - Moves the cursor to the next position after the **Back** button has been used at least once.

Navigating with the Outline View

Oxygen XML Editor includes a very useful ***Outline*** *view* that displays a hierarchical tag overview of the currently edited XML Document.

You can use this view to quickly navigate through the current document by selecting nodes in the outline tree. It is synchronized with the editor area, so when you make a selection in the **Outline** view, the corresponding nodes are highlighted in the editor area.

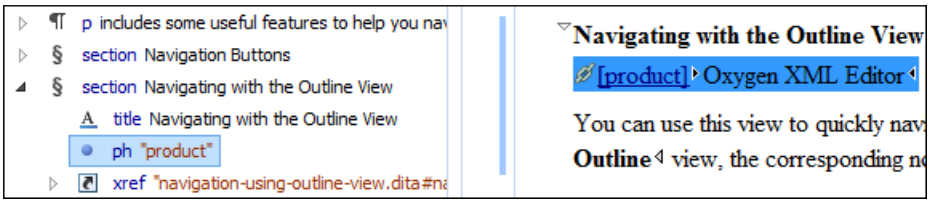


Figure 78: Outline View Navigation in Author Mode

Using the Breadcrumb to Navigate

A *breadcrumb* on the top stripe indicates the path from document root to the current element. It can also be used as a helpful tool to navigate to specific elements throughout the structure of the document.

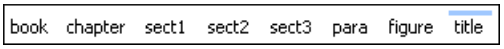


Figure 79: Breadcrumb in Author Mode

The last element listed in the *breadcrumb* is the element at the current cursor position. The last element is also highlighted by a thin light blue bar for easier identification. Clicking an element from the *breadcrumb* selects the entire element and navigates to it in the editor area.

Using the Linking Support

When working on multiple documents that reference each other (references, external entities, XInclude, DITA conref, etc), the **linking support** is useful for navigating between the documents. In the predefined customizations that are bundled with Oxygen XML Editor, links are marked with an icon representing a chain link (). When hovering over the icon, the mouse pointer changes its shape to indicate that the link can be accessed and a tooltip presents the destination location. Click the link to open the referenced resource in the editor or system browser. The same effect can be obtained by using the **Document** > **File** > **Open file at cursor (Ctrl + Enter (Command + Enter on OS X))** action when the cursor is inside a link element.

Note: Depending on the referenced file type, the target link will either be opened in the Oxygen XML Editor or in the default system application. If the target file does not exist, Oxygen XML Editor prompts you to create it.

Navigating with Bookmarks

A position in a document can be marked with a *bookmark*. You can then quickly go to the marked position with a keyboard shortcut or a menu action. This is useful when navigating large documents or working on multiple documents where the cursor needs to move between several marked positions. The *bookmarks* are displayed with a small icon on the vertical strip to the left of the editor. You can place up to nine distinct *bookmarks* in any document. Shortcut keys are available to place the *bookmarks* or to return to any of the marked positions. You can configure these shortcut keys in the ***Options > Menu Shortcut Keys*** menu.

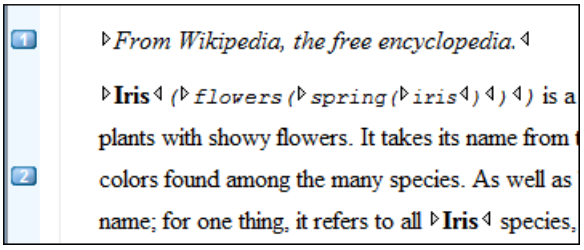


Figure 80: Editor Bookmarks

A *bookmark* can be inserted in **Author** mode by doing one of the following:

- Click in the vertical stripe on the left side of the editor.
- Select the **Create Bookmark (F9)** action from the **Edit** > **Bookmarks** menu.

A *bookmark* can be removed by right-clicking its icon on the vertical stripe and select **Remove** or **Remove all (Ctrl +F7 (Command+F7 on OS X))**.

You can navigate the *bookmarks* by using one of the actions available on the **Edit** > **Bookmarks** > **Go to** menu or by using the shortcut keys that are listed in that menu.

Author Mode Views

The content author is supported by a variety of *dockable* helper views that are displayed by default when editing in **Author** mode. These views are automatically synchronized with the current editing context of the editor panel. They present additional information about this context thus helping the author to see quickly the current location in the overall document structure and the available editing options.

There is also a large selection of additional useful views available in the **Window** > **Show View** menu. This section presents some of the most helpful views for editing in **Author** mode.

Project View

The **Project** view is designed to assist you with organizing and managing related files grouped in the same XML project. The actions available in the contextual menu and on the toolbar associated to this panel allow you to create XML projects and provide shortcuts to various operations for the project documents.

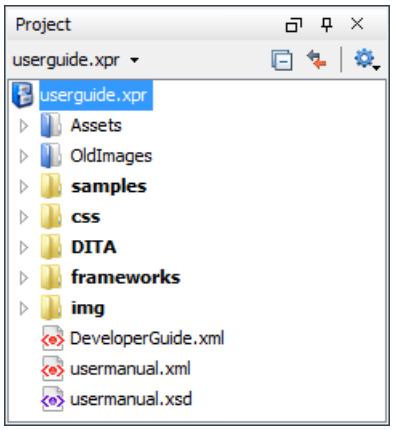


Figure 81: Project View

By default, the view is positioned on the left side of the Oxygen XML Editor window, above the ***Outline*** *view*. If the view has been closed, it can be reopened at any time from the **Window** > **Show View** menu (or using the **Show Project View** action from the **Project** menu).

The tree structure occupies most of the view area. In the upper left side of the view, there is a drop-down menu that contains all recently used projects and project management actions:

Open Project (Ctrl + F2 (Command + F2 on OS X))

Opens an existing project. Alternatively, you can open a project by dropping an Oxygen XML Editor XPR project file from the file explorer into the **Project** panel.

Notice: When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

New Project

Creates a new, empty project.

The following actions are grouped in the upper right corner:

Collapse All

Collapses all project tree folders. You can also collapse/expand a project tree folder if you select it and press the **Enter** key or **Left Arrow** to collapse and **Right Arrow** to expand.

Link with Editor

When selected, the project tree highlights the currently edited file, if it is found in the project files.

Note: This button is disabled automatically when you move to the **Debugger** *perspective*.

Settings

A submenu that contains the following actions:

Filters

Allows you to filter the information displayed in the **Project** view. Click the toolbar button to set filter patterns for the files you want to show or hide. Also, you can set filter patterns for the linked directories that are hidden.

Show Full Path

When selected, linked files and folders are presented with a full file path.

Enable Master Files Support

Select this option to enable the ***Master Files*** *support*.

Change Search and Refactor operations scope

Allows you to change the collection of documents that define the context of the *search and refactor operations*.

- **Use only Master Files, if enabled** - Restricts Oxygen XML Editor to perform the search and refactor operations starting from the *master files* that are defined for the current resource. This option is available when you select **Project** in the **Select the scope for Search and Refactor operations** dialog box and the **Master Files** support is enabled.
- **Working sets** - Allows you to specify the set of files that will be used for the scope of the search and refactor operations.

The files are usually organized in an XML project as a collection of folders. There are three types of resources displayed in the **Project** view:

- **Logical folders** - marked with a blue icon on Windows and Unix/Linux ( ) and a magenta icon on Mac OS X ( ). They help you group files within the project. This folder type has no correspondent on the physical disk, since they are used as containers for related items. Creating and deleting them does not affect the file system on disk. They are created on the project root or inside other logical folders by using the contextual action **New** > **Logical Folder**. The contextual menu action **Remove from Project** can be used to remove them from the project.
- **Physical folders and files** - marked with the operating system-specific icon for folders (usually a yellow icon on Windows and a blue icon on Mac OS X). These folders and files are mirrors of real folders or files that exist in the local file system. They are created or added to the project by using contextual menu actions (such as **New** > **File**, **New** > **Folder**, **Add Folder**, etc.) Also, the contextual menu action **Remove from Disk (Shift +Delete)** can be used to remove them from the project and local file system.
- **Shortcut folders and files** - the icons for file shortcuts include a shortcut symbol and names of folder shortcuts are displayed in bold text. All files and folders that appear as direct descendants of a logical folder are considered shortcuts. They are created and added with the contextual actions **Add Files** and **Add Folder** from the project root. Both contextual menu actions **Remove from Project** and **Remove from Disk (Shift +Delete)** are available for shortcuts. **Remove from Project** just removes the shortcut from the project, while **Remove from Disk (Shift+Delete)** removes the shortcut and the physical resource from the local file system.

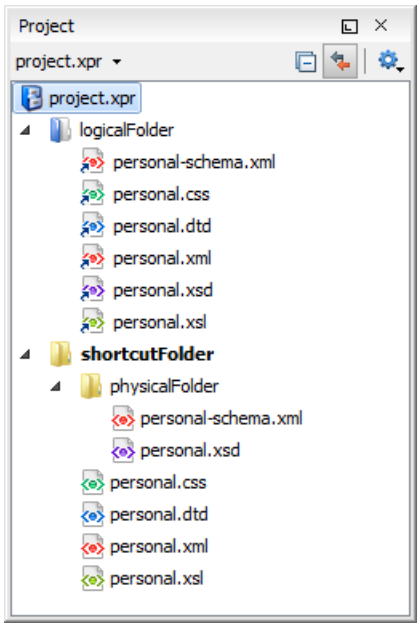


Figure 82: Project View with Examples of all Three Types of Resources

Creating New Projects

The following action is available in the **Project** menu, the **New** menu in the contextual menu, or from the drop-down menu in the top-left of the **Project** view:

New Project

Creates a new, empty project.

Creating New Project Items

The following actions are available by selecting **New** from the contextual menu, when invoked from the **Project** view:

New > File

Opens a ***New*** *file dialog box* that helps you create a new file and adds it to the project structure.

New > Folder

Opens a **New Folder** dialog box that allows you to specify a name for a new folder and adds it to the structure of the project.

New > Logical Folder

Available when invoked from the *project root*, this action creates a logical folder in the tree structure (the icon is a magenta folder on Mac OS X -).

New > Logical Folders from Web

Available when invoked from the *project root*, this action replicates the structure of a remote folder accessible over FTP/SFTP/WebDAV, as a structure of logical folders. The newly created logical folders contain the file structure of the folder it points to.

Managing Physical Folders and Files

You can create physical folders by selecting **New** > **Folder** from the contextual menu.

When adding files to a project, the default target is the project root. To change a target, select a new folder. Files may have multiple instances within the folder system, but cannot appear twice within the same folder.

Removing Files and Folders

To remove one or more linked files or folders, select them in the project tree and press the **Delete** key, or select the contextual menu action **Remove from Project**. To remove a linked file or folder from both project and local file system, select the contextual menu action **Remove from Disk (Shift+Delete)**. The **Remove from Disk (Shift+Delete)** action is also used to remove physical files or folders.

CAUTION: In most cases this action is irreversible, deleting the file permanently. Under particular circumstances (if you are running a Windows installation of Oxygen XML Editor and the *Recycle Bin* is active) the file is moved to *Recycle Bin*.

Moving Files and Folders

You can *move the resources of the project* with drag and drop operations on the files and folders of the tree (the **Enable drag-and-drop in Project view** option must be selected in the ***View*** *preferences page*).

You can also use the usual **Cut**, **Copy**, and **Paste** actions to move resources in the **Project** view.

Renaming Files and Folders

There are three ways you can *rename an item in the* ***Project*** *view*. Select the item in the **Project** view and do one of the following:

- Invoke the **Rename** action from the contextual menu.
- Press **F2** and type the new name.
- Click the selected item and type the new name.

  To finish editing the item name, press **Enter**.

Note: The **Rename** action is also available on *logical* files.

Locating and Opening Files

If a project folder contains a lot of documents, a certain document can be located quickly in the project tree by selecting the folder containing the desired document and typing the first few characters of the document name. The desired document is automatically selected as soon as the typed characters uniquely identify its name in the folder.

The selected document can be opened by pressing the **Enter** key, by double-clicking it, or with one of the **Open** actions from the contextual menu. The files with known document types are opened in the associated editor, while binary files are opened with the associated system application. To open a file with a known document type in an editor other than the default one, use the **Open with** action. Also, dragging and dropping files from the project tree to the editor area results in the files being opened.

Saving the Project

The project file is automatically saved every time the content of the **Project** view is saved or modified by actions such as adding or removing files and drag and drop.

Linked Folders

You can create linked folders (shortcuts) by dragging and dropping folders from a system explorer to the project tree (the **Enable drag-and-drop in Project view** option must be selected in the ***Views*** *preferences page*), or by selecting **Add Folder** in the contextual menu from the project root.

To create a file inside a linked folder, select the **New** > **File** action from the contextual menu. This opens the ***New Document*** *wizard*.

Note: The linked files presented in the **Project** view are marked with a special icon. Linked folders are displayed in bold text.

Logical Folders

The project itself is considered a logical folder. You can add content to a logical folder using one of the actions available in the contextual menu, when invoked from the *project root*:

Add Folder

Adds a link to a physical folder, whose name and content mirror a real folder that exists in the local file system (the icon of this action is different on Mac OS X).

Add Files

Adds links to files on the local file system.

Add Edited File

Adds a link to the currently edited file in the project.

Validate Files

The currently selected files in the **Project** view can be checked to be XML well-formed or validated against a schema (DTD, XML Schema, Relax NG, Schematron or NVDL) with one of the following contextual menu actions found in the **Validate** submenu:

Check Well-Formedness

Checks if the selected file or files are well-formed.

Validate

Validates the selected file or files against their associated schema. EPUB files make an exception, because this action triggers a ***Validate and Check for Completeness*** operation.

Validate with Schema

Validates the selected file of files against a specified schema.

Configure Validation Scenario(s)

Allows you to configure and run a *validation scenario*.

Applying Transformation Scenarios

The currently selected files in the **Project** view can be transformed in one step with one of the following actions available from contextual menu in the **Transform** submenu:

Apply Transformation Scenario(s)

Obtains the output with one of the *built-in scenarios*.

Configure Transformation Scenario(s)

Opens a *dialog box* that allows you to configure pre-defined transformation scenarios.

Transform with

Allows you to select a transformation scenario to be applied to the currently selected files.

Along with the logical folder support, this allows you to group your files and transform them very easily.

Refactoring Actions (Available for certain document types (such as XML, XSD, and XSL)

Oxygen XML Editor includes some refactoring operations that help you manage the structure of your documents. The following actions are available from the contextual menu in the **Refactoring** submenu:

Rename resource

Allows you to change the name of a resource.

Move resource

Allows you to change the location on disk of a resource.

XML Refactoring

Opens *the **XML Refactoring** tool wizard* that presents refactoring operations to assist you with managing the structure of your XML documents.

Other XML Refactoring Actions

For your convenience, the last 5 *XML Refactoring tool operations* that were finished or previewed will also appear in this submenu.

Other Contextual Menu Actions

Other actions that are available in the contextual menu from the project tree include:

Open

Opens the selected files in the corresponding editor.

Open with submenu

This submenu allows you to open the selected file with the internal editor, a system application, or other internal tools: ***DITA Maps Manager***, ***Archive Browser***, *MathML Editor*, *Generate/Convert Schema*, ***WSDL/SOAP Analyzer***, *Large File Viewer*, *Hex Viewer*, *SVG Viewer*.

Show in Explorer (or Show in Finder on OS X)

In Windows, the content of the selected folder or file is presented in a specific explorer window. On MAC OS X, the parent folder is opened and the selected folder is highlighted in a specific finder window.

Copy Location

Copies an application-specific URL for the selected resource to the clipbord.

Refresh

Refreshes the content and the dependencies between the resources in *the **Master Files** directory*.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files.

XPath in Files

Opens the ***XPath/XQuery Builder*** *view* that allows you to compose XPath and XQuery expressions and execute them over the currently edited XML document.

Open/Find Resource

Opens the ***Open/Find Resource*** *dialog box*.

Check Spelling in Files

Allows you to *check the spelling of multiple files.*

Format and Indent Files

Opens the ***Format and Indent Files*** *dialog box* that allows you to configure the format and indent (*pretty-print*) action that will be applied on the selected documents.

Open in SVN Client

Syncro SVN Client tool is opened and it highlights the selected resource in its corresponding working copy.

Compare

Allows you to compare multiple files or directories and the order of your selection determines where they are opened in the ***Compare Files*** or ***Compare Directories*** tool. If you select two files or folders, your first selection will be opened in the left panel and the other one in the right panel.

You can also select 3 files and the tool will automatically be opened in the *three-way comparison mode*. If you select three files, your first selection will be opened in the left panel, the second in the right panel, and the third selection will be the base (ancestor) file.

Generate Documentation > XML Schema Documentation

Opens *the XML Schema Documentation Dialog Box*.

Generate Documentation > XSLT Stylesheet Documentation

Opens *the XSLT Stylesheet Documentation Dialog Box*.

Generate Documentation > XQuery Documentation

Opens *the XQuery Documentation Dialog Box*.

Generate Documentation > WSDL Documentation

Opens *the WSDL Documentation Dialog Box*.

Properties

Displays the properties of the current file in a **Properties** dialog box.

Menu Level Actions

The following actions are available in the **Project** menu:

New Project

Creates a new, empty project.

Open Project (Ctrl + F2 (Command + F2 on OS X))

Opens an existing project. Alternatively, you can open a project by dropping an Oxygen XML Editor XPR project file from the file explorer into the **Project** panel.

Notice: When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

Save Project As

Allows you to save the current project under a different name.

Validate all project files

Checks if the project files are well-formed and their mark-up conforms with the specified DTD, XML Schema, or Relax NG schema rules. It returns an error list in the message panel.

Filters

Opens the **Project filters** dialog box that allows you to decide which files and directories will be shown or hidden.

Enable Master Files Support

Allows you to enable the ***Master Files Support*** for each project you are working on.

Change Search and Refactor operations scope

Opens a dialog box that allows you to define the context of search and refactor operations.

Show Project View

Displays the **Project** view.

Reopen Project

Contains a list of links of previously used projects. This list can be emptied by invoking the **Clear history** action.

Open/Find Resource View

The **Open/Find Resource** view is designed to offer advanced search capabilities either by using a simple text search or by using the *Apache Lucene - Query Parser Syntax*. By default, the view is presented in the left side of the Oxygen XML Editor layout, next to the ***Project*** *view* or ***DITA Maps Manager***. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

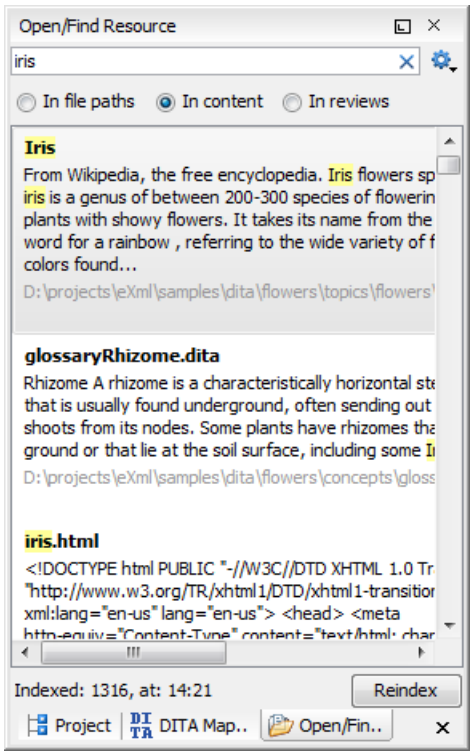


Figure 83: Open/Find Resource View

You can use this view to find a file in the current Oxygen XML Editor project or in one of the *DITA maps* opened in *the **DITA Maps Manager** view* by typing only a few letters of the file name of a document or a fragment of the content you are searching for. The **Open/Find Resource** view also supports searching in document edits (comments, tracked change insertions/deletions, and highlighted content) by selecting the ***In reviews** option*.

Note: Full support for searching in document edits (the **In reviews** option) is available only in the Enterprise edition of Oxygen XML Editor. The Professional edition offers limited support to search through a maximum of 10 edits.

Search Results

Search results are presented instantly, after you finish typing the content. The matching fragments of text are highlighted in the results list displayed in the dialog box. When you open one of the documents from the results list, the matching fragments of text are highlighted in the editing area. To remove the highlighting from your document, close the corresponding tab in the ***Results** view* at the bottom of the editor. To display the search history, position the cursor in the search field and press **Ctrl + DownArrow (Command + DownArrow on OS X)** or **Ctrl + UpArrow (Command + UpArrow on OS X)** on your keyboard. Pressing only the **DownArrow** key moves the selection to the list of results.

Note: Searches are not case sensitive. For example, if you search for `car` you get the same results as when you search for `Car`.

Tip: Suffix searches are also supported, both for searching in the content of your resources and in their name. For this, you can use wildcards. If you search for `*ing` with the **in content** option selected, you will find documents that contain the word *presenting*. If you search for `*/samples/*.gif` with the **in file paths** option selected, you will find all the *gif* images from the `samples` directory.

Options Available in the View

The **Open/Find Resource** view offers the following options:

- **Settings** - Drop-down menu that includes the following settings for the view:
  - **Clear Index** - Clears the index.

- **Show description** - Presents the search results in a more compact form, displaying only the title and the location of the resources.
- **Options** - Opens the ***Open/Find Resource*** *preferences page* where you can configure various search options. For example, you can specify a **Content language** that differs from the default UI language in case your document contains multiple languages.

- ***In file paths*** - Select this option to search for resources by their name or by its path (or a fragment of its path).
- ***In content*** - Select this option to search through the content of your resources.
- ***In reviews*** - Select this option to search through the comments, *tracked change* insertions/deletions, or highlights in your resources.
- **Reindex** - Use this option to reindex your resources.

Contextual Menu Actions

A contextual menu is available on each search result and provides actions applicable to that particular document. These actions include:

- **Open** - Opens the document in one of Oxygen XML Editor internal editors.
- **Open with** - Allows you to choose to open the document in the **Internal editor** or an external **System application**.
- **Show in Explorer** - Identifies the document in the system file explorer.
- **Copy Location** - Copies the file path and places it in the clipboard.

Indexing Process

The content of the resources used to search in is parsed from an index. The indexing is performed both automatically and on request. Automatic indexing is performed when you modify, add, or remove resources in the currently indexed project. If the index was never initialized, the index in not updated on project changes.

To improve performance, the indexing process skips the following set of common English words (the so-called **stop words**): *a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with*. This means that if you are searching for any of these words, the indexing process will not be able to match any of them. However, you can configure the list of **stop words** in the ***Open/Find Resource*** *preferences page*.

Caching Mechanism

When you perform a search, a caching mechanism is used to gather the paths of all files linked in the current project. When the first search is performed, all project files are indexed and added to the cache. The next search operation uses the information extracted from the cache, thus improving the processing time. The cache is kept for the currently loaded project only, so when you perform a search in a new project, the cache is rewritten. Also, the cache is reset when you press the **Reindex** button.

Important: Files larger than 2GB are not indexed.

If there is no file found that matches your file pattern or text search, a possible cause is that the file you are searching for was added to the Oxygen XML Editor project after the last caching operation. In this case, re-indexing the project files with the **Reindex** button enables the file to be found. The date and time of the last index operation are displayed below the file list.

Opening the Results

Once you find the files that you want to open, select them in the list and press the **Open** button (or double-click them). Each of the selected files is opened in *the editor associated with the type of the file*.

Note: You can drag a resource from the **Open/Find Resource** view and drop it in a DocBook, DITA, TEI or XHTML document to create a link to that resource.

To watch our video demonstration about the **Open/Find Resource** feature and its search capabilities, go to *https://www.oxygenxml.com/demo/Open_Find_Resource.html*.

Related Information:

Open/Find Resource Dialog Box on page 271

Outline View in Author Mode

The **Outline** view in **Author** mode displays a general tag overview of the currently edited XML Document. When you edit a document, the **Outline** view dynamically follows the changes that you make, displaying the node that you modify. This functionality gives you great insight on the location of your modifications in the current document. It also shows the correct hierarchical dependencies between elements. This makes it easy for you to be aware of the document structure and the way element tags are nested.

Outline View Features

The **Outline** view allows you to:

- Quickly navigate through the document by selecting nodes in the **Outline** tree.
- Insert or delete nodes using contextual menu actions.
- Move elements by dragging them to a new position in the tree structure.
- Highlight elements in the editor area. It is synchronized with the editor area, so when you make a selection in the editor area, the corresponding nodes are highlighted in the **Outline** view, and vice versa.
- View document errors, as they are highlighted in the **Outline** view. A tooltip also provides more information about the nature of the error when you hover over the faulted element.

Outline View Interface

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (*, ?) and separate multiple patterns with commas.

It also includes a **Settings** menu in the top-right corner that presents a variety of options to help you filter the view even further.

Drop and Drop Actions in the Outline View

Entire XML elements can be moved or copied in the edited document using only the mouse in the **Outline** view with drag-and-drop operations. Several drag and drop actions are possible:

- If you drag an XML element in the **Outline** view and drop it on another node, then the dragged element will be moved after the drop target element.
- If you hold the mouse pointer over the drop target for a short time before the drop then the drop target element will be expanded first and the dragged element will be moved inside the drop target element after its opening tag.
- You can also drop an element before or after another element if you hold the mouse pointer towards the upper or lower part of the targeted element. A marker will indicate whether the drop will be performed before or after the target element.
- If you hold down the **(Ctrl (Command on OS X))** key after dragging, a copy operation will be performed instead of a move.

Tip: You can select and drag multiple nodes in the **Outline** view when editing in **Author** mode.

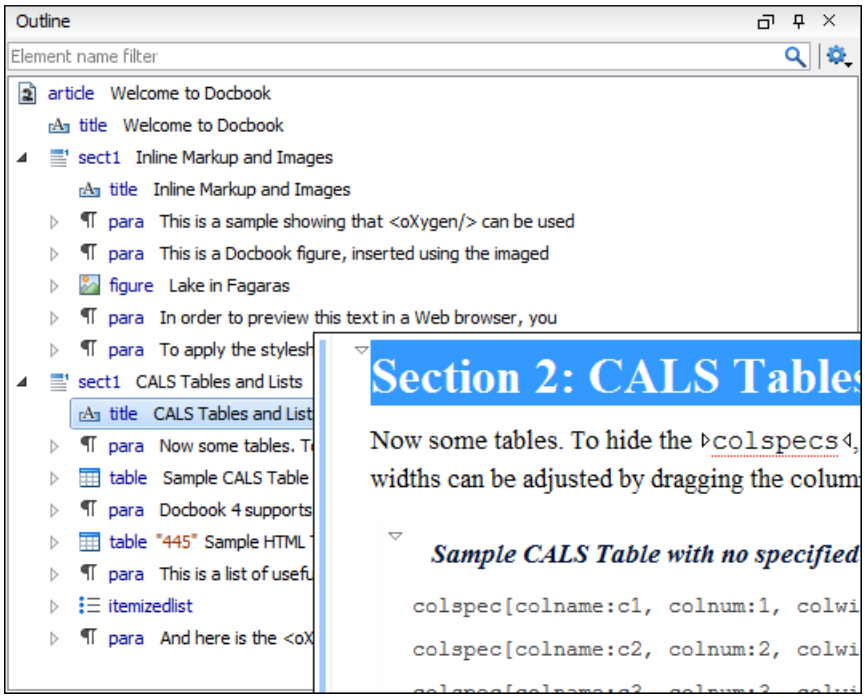


Figure 84: Outline View

Outline View Filters in Author Mode

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (*, ?) and separate multiple patterns with commas.

The following actions are available in the **Settings** menu of the **Outline** view when editing in **Author** mode:

Filter returns exact matches

The text filter of the **Outline** view returns only exact matches.

Flat presentation mode of the filtered results

When active, the application flattens the filtered result elements to a single level.

Show comments and processing instructions

Show/hide comments and processing instructions in the **Outline** view.

Show element name

Show/hide element name.

Show text

Show/hide additional text content for the displayed elements.

Show attributes

Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from *the Outline preferences panel*.

Configure displayed attributes

Displays the *XML Structured Outline preferences page*.

Outline View Contextual Menu Actions in Author Mode

The contextual menu of the **Outline** view in **Author** mode contains the following actions:

Edit Attributes

Displays an *in-place attributes editor* that allows you to edit the attributes of a selected node.

Edit Profiling Attributes

Allows you to change the *profiling attributes* defined on all selected elements.

Append Child

Invokes a content completion list with the names of all the elements that are allowed by the associated schema and inserts your selection as a child of the current element.

Insert Before

Invokes a content completion list with the names of all the elements that are allowed by the associated schema and inserts your selection immediately before the current element, as a sibling.

Insert After

Invokes a content completion list with the names of all the elements that are allowed by the associated schema and inserts your selection immediately after the current element, as a sibling.

Cut, Copy, Paste, Delete editing actions

Executes the typical editing actions on the currently selected elements. The **Cut** and **Copy** operations preserve the styles of the copied content. The **Paste before** and **Paste after** actions allow you to insert a well-formed element before or after the currently selected element. The **Paste as XML** action pastes copied content that is considered to be valid XML, preserving its XML structure.

Toggle Comment

Encloses the currently selected element in an XML comment, if the element is not already commented. If it is already commented, this action will remove the comment.

Rename Element

Invokes a **Rename** dialog box that allows you to rename the currently selected element, siblings with the same name, or all elements with the same name.

Expand More

Expands the structure tree of the currently selected element.

Collapse All

Collapses all of the structure tree of the currently selected node.

Tip: You can copy, cut or delete multiple nodes in the **Outline** by using the contextual menu after selecting multiple nodes in the tree.

Related Information:

Attributes View in Author Mode on page 224

Attributes View in Author Mode

The **Attributes** view presents all the attributes of the current element determined by the schema of the document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened from the **Window** > **Show View** menu.

You can use this view to edit or add attribute values. The attributes of an element are editable if any one of the following is true:

- The CSS stylesheet associated with the document does not specify a **false** value for the *-oxy-editable* property associated with the element.
- The element is entirely included in a deleted *Track Changes* marker.
- The element is part of a content fragment that is referenced in **Author** mode from another document.

The attributes are rendered differently depending on their state:

- The names of the attributes are rendered with a bold font, and their values with a plain font.
- Default values are rendered with a plain font, painted gray.
- Empty values display the text "[empty]", painted gray.
- Invalid attributes and values are painted red.

To edit the value of the corresponding attribute, double-click a cell in the **Value** column . If the possible values of the attribute are specified as `list` in the schema of the edited document, the **Value** column acts as a combo box that allows you to either select the value from a list or manually enter it.

Note: If the cursor is located inside read-only content, the attribute names and values are faded and you cannot add, edit, or remove values.

You can sort the attributes table by clicking the **Attribute** column header. The table contents can be sorted as follows:

- By attribute name in ascending order.
- By attribute name in descending order.
- Custom order, where the used attributes are displayed at the beginning of the table sorted in ascending order, followed by the rest of the allowed elements sorted in ascending order.

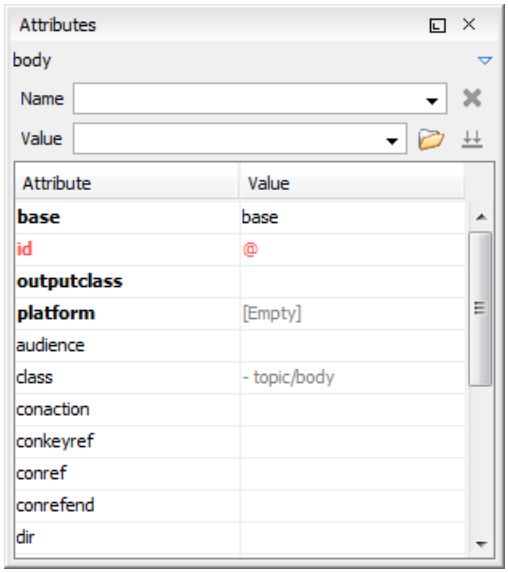


Figure 85: Attributes View

A drop-down list located in the upper part of the view allows you to select the current element or its ancestors.

Expand/Collapse Button

There is an **Expand/** **Collapse** button at the top-right of the view. When expanded, this presents the following additional combo boxes:

Name Combo Box

Use this combo box to select an attribute. The drop-down list displays the list of possible attributes allowed by the schema of the document, as in the **Attributes** view. You can use the **Remove** button to delete an attribute and its value from the selected element.

Value Combo Box

Use this combo box to add, edit, or select the value of an attribute. If the selected attribute has predefined values in the schema, the drop-down list displays those possible values. You can use the **Browse** button to select a URL for the value of an attribute. After you have entered or selected a value, use the **Update** button (or press **Enter**) to add the value to the attribute.

Contextual Menu Actions in the Attributes View

The following actions are available in the contextual menu of the **Attributes** view when editing in **Author** mode:

Set empty value

Specifies the current attribute value as empty.

Remove

Removes the attribute (action available only if the attribute is specified). You can invoke this action by pressing the **(Delete)** or **(Backspace)** keys.

Copy

Copies the `attrName="attrValue"` pair to the clipboard. The `attrValue` can be:

- The value of the attribute.
- The value of the default attribute, if the attribute does not appear in the edited document.
- Empty, if the attribute does not appear in the edited document and has no default value set.

Paste

Depending on the content of the clipboard, the following cases are possible:

- If the clipboard contains an attribute and its value, both of them are introduced in the **Attributes** view. The attribute is selected and its value is changed if they exist in the **Attributes** view.
- If the clipboard contains an attribute name with an empty value, the attribute is introduced in the **Attributes** view and you can start editing it. The attribute is selected and you can start editing it if it exists in the **Attributes** view.
- If the clipboard only contains text, the value of the selected attribute is modified.

In-place Attributes Editor

Oxygen XML Editor includes an in-place attributes editor in **Author** mode. To edit the attributes of an XML element in-place, do one of the following:

- Select an element or place the cursor inside it and then press the **Alt + Enter** keyboard shortcut.
- Double-click any named start tag when the document is edited in one of the following *display modes.*: **Full Tags with Attributes**, **Full Tags**, **Block Tags**, or **Inline Tags**.

This opens an in-place attributes editor that contains the same content as the **Attributes** view. By default, this editor presents the **Name** and **Value** fields, with the list of all the possible attributes collapsed.

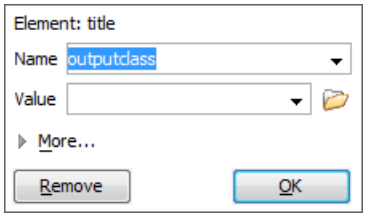


Figure 86: In-place Attributes Editor

Name Combo Box

Use this combo box to select an attribute. The drop-down list displays the list of possible attributes allowed by the schema of the document, as in the **Attributes** view.

Value Combo Box

Use this combo box to add, edit, or select the value of an attribute. If the selected attribute has predefined values in the schema, the drop-down list displays those possible values.

If you click **More** while in the collapsed version, it is expanded to the full version of the in-place attribute editor.

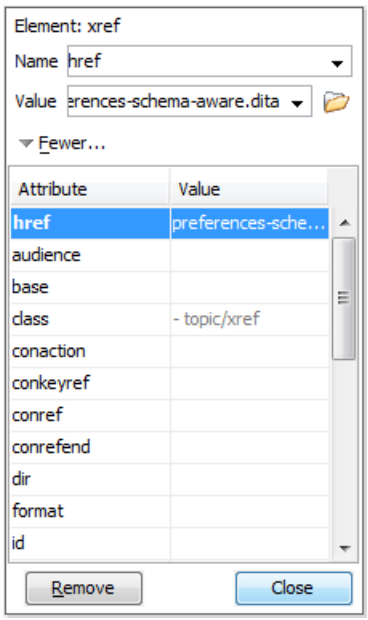


Figure 87: In-place Attributes Editor (Full Version)

The full version includes a table grid, similar to the **Atributes** view, that presents all the attributes for the selected element.

Model View

The **Model** view presents the structure of the currently selected tag, and its documentation, defined as annotation in the schema of the current document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

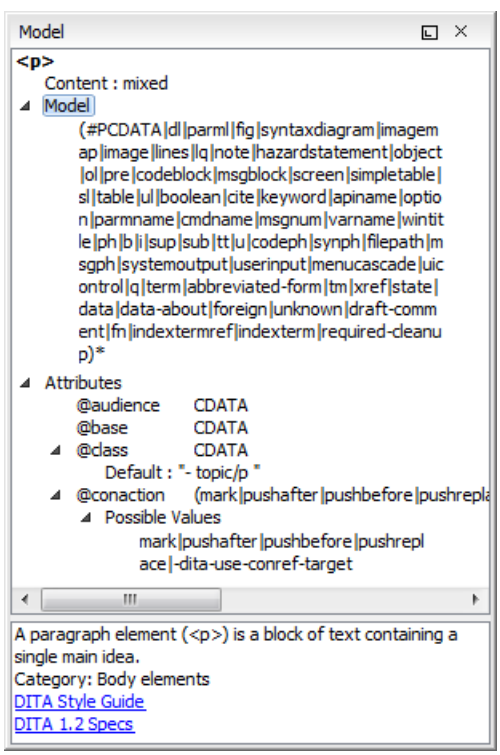


Figure 88: Model View

The **Model** view is comprised of two sections, an element structure panel and an annotations panel.

Element Structure Panel

The element structure panel displays the structure of the currently edited or selected tag in a tree-like format. The information includes the name, model, and attributes of the current tag. The allowed attributes are shown along with imposed restrictions, if any.

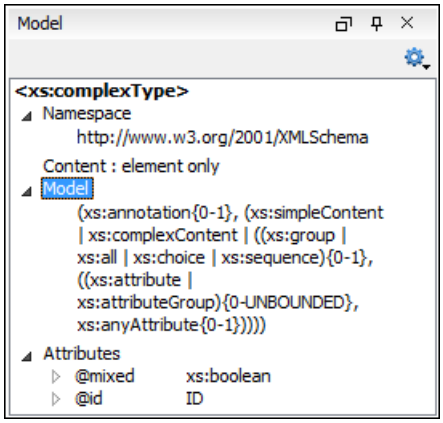


Figure 89: Element Structure Panel

Annotation Panel

The **Annotation** panel displays the annotation information for the currently selected element. This information is collected from the XML schema.

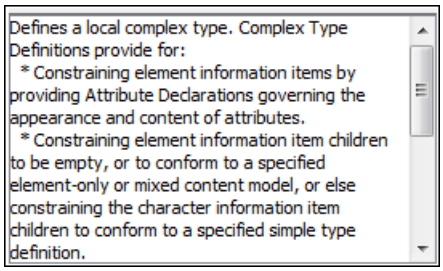


Figure 90: Annotation panel

Elements View in Author Mode

The **Elements** view presents a list of all defined elements that are valid at the current cursor position according to the schema associated to the document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

The upper part of the view features a combo box that contains the ordered ancestors of the current element. Selecting a new element in this combo box updates the list of the allowed elements. By default, only the elements that are allowed at the current cursor position are listed. However, if the ***Show only items allowed at cursor position*** *option* is not selected in the ***Views*** *preferences page*, two additional tabs (**Before** and **After**) will be displayed at the bottom of the view and they list elements that are allowed before or after the element at the current cursor position.

Double-clicking any of the listed elements inserts that element into the edited document and its position depends on the tab.

- **Cursor** tab - Double-clicking an element inserts it at the current cursor position.
- **Before** tab - Double-clicking an element inserts it before the element at the cursor position.
- **After** tab - Double-clicking an element inserts it after the element at the cursor position.

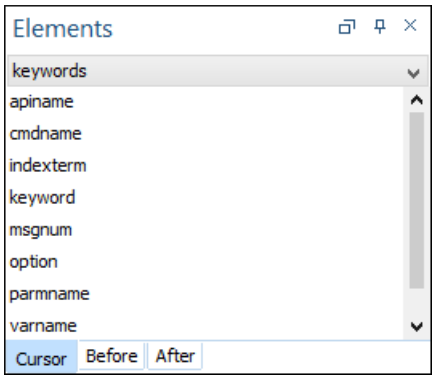


Figure 91: Elements View in Author Mode

Entities View

Entities provide abbreviated entries that can be used in XML files when there is a need of repeatedly inserting certain characters or large blocks of information. An *entity* is defined using the `ENTITY` statement either in the DOCTYPE declaration or in a DTD file associated with the current XML file.

There are three types of entities:

- **Built-in** or **Predefined** - Entities that are part of the predefined XML markup (`<`, `>`, `&`, `'`, `"`).
- **Internal** - Defined in the DOCTYPE declaration header of the current XML.
- **External** - Defined in an external DTD module included in the DTD referenced in the XML DOCTYPE declaration.

Note: If you want to add internal entities, you would need to switch to the Text editing mode and manually modify the DOCTYPE declaration. If you want to add external entities, you need to open the DTD module file and modify it directly.

The **Entities** view displays a list with all entities declared in the current document, as well as built-in ones. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Double-clicking one of the entities will insert it at the current cursor position in the XML document. You can also sort entities by name and value by clicking the column headers.

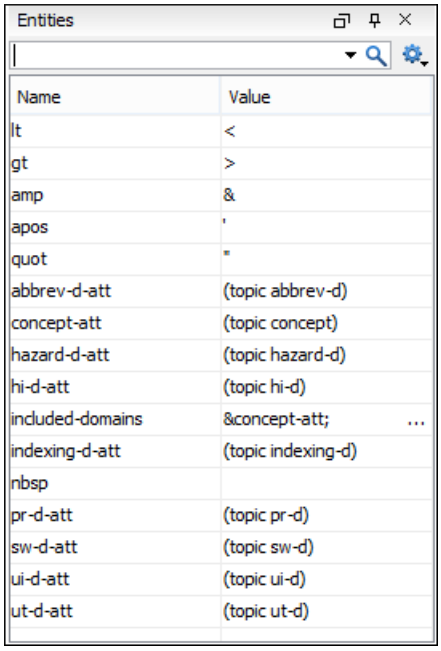


Figure 92: Entities View

The view features a filtering capability that allows you to search an entity by name, value, or both. Also, you can choose to display the internal or external entities.

Note: When entering filters, you can use the ? and * wildcards. Also, you can enter multiple filters by separating them with a comma.

Results View

The **Results** view displays the messages generated as a result of user actions such as validations, transformations, search operations, and others. Each message is a link to the location related to the event that triggered the message. Double-clicking a message opens the file containing the location and positions the cursor at the location offset. The **Results** view is automatically opened when certain actions generate result messages. Those actions include the following:

- ***Validation*** *actions*
- ***Transformation*** *actions*
- ***Check Spelling in Files*** *action*
- **Find All** action from the ***Find/Replace*** *dialog box*
- ***Find/Replace in Files*** *dialog box*
- ***Search References*** *action*
- *XPath expression results*
- *SQL results*

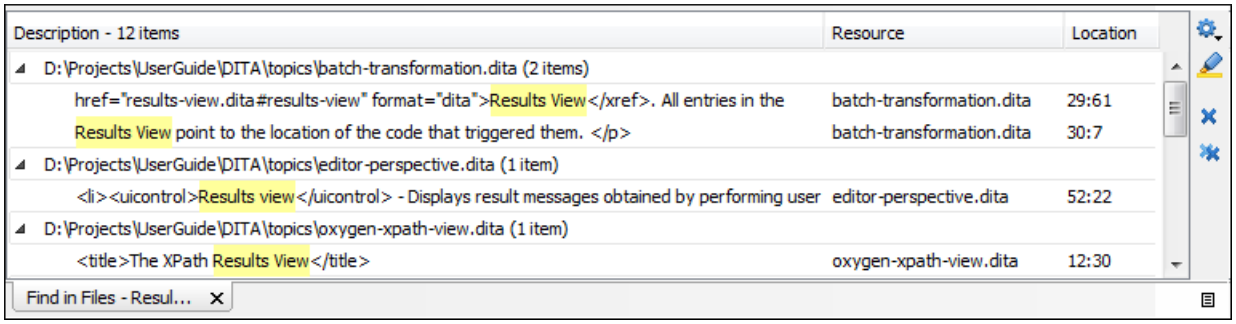


Figure 93: Results View

Results View Toolbar Actions

The view includes a toolbar with the following actions:

Grouping options drop-down menu

A set of **Group by** toggle actions that allow you to group the messages according to a selected criteria so that they can be presented in a hierarchical layout. The criteria used for grouping can be the severity of the errors (error, warning, info message, etc.), the resource name, the description of the message, and so on.

This drop-down menu also includes the following additional grouping actions:

Ungroup all

Removes the grouping rules so that the messages are presented in a continuous list.

Show group columns

If any of the **Group by** options are selected, you can use this option to show or hide grouping columns.

Restore Defaults

Restores the column size for each column and the grouping rules that were saved in the user preferences the last time when this view was used. If it is the first time this view is used, the action sets an initial default column size for each column and a grouping rule that is appropriate for the type of messages. For example:

- Group the messages by the path of the validated file if there are error messages from a validation action or spelling errors reported by the ***Check Spelling in Files*** *action*.

- No grouping rule for the results of *applying an XPath expression*.

Highlight all results in editor

Oxygen XML Editor highlights all matches obtained after executing an XPath expression, or performing one of the following operations: **Find All**, **Find in Files**, **Search References**, and **Search Declarations**. Click **Highlight all results in editor** again to turn off highlighting.

Note: To customize highlighting behavior, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Highlights category**. You can do the following customizations:

- Set a specific color of the highlights depending on the type of action you make.
- Set a maximum number of highlights that the application displays at any given time.

Remove selected

Removes the current selection from the view. This can be helpful if you want to reduce the number of messages or remove those that have already been addressed or not relevant to your task.

Remove all

Removes all messages from the view.

Results View Contextual Menu Actions

The following actions are available when the contextual menu is invoked in this view:

Show message

Displays a dialog box with the full error message, which is useful for a long message that does not have enough room to be displayed completely in the view.

Previous message

Navigates to the message above the current selection.

Next message

Navigates to the message below the current selection.

Remove selected

Removes selected messages from the view.

Remove all

Removes all messages from the view.

Copy

Copies information associated with the selected messages. For example:

- The file path of the document that triggered the output message.
- The path of the *master file* (in the case of a *validation scenario*, it is the path of the file from which the validation starts and can be different from the validated file).
- Error severity (error, warning, info message, etc.)
- Name of validating processor.
- Name of *validation scenario*.
- The line and column in the file that triggered the message.

Select All

Extends the selection to all the messages from the view.

Print Results

Sends the complete list of messages to a printer. For each message, the included details are the same as the ones for the ***Copy** action*.

Save Results

Saves the complete list of messages in a file in text format. For each message, the included details are the same as the ones for the ***Copy** action*.

Save Results as XML

Saves the complete list of messages in a file in XML format. For each message, the included details are the same as the ones for the ***Copy*** *action*.

Save Results as HTML

Saves the complete list of messages in a file in HTML format. For each message, the included details are the same as the ones for the ***Copy*** *action*.

Group by

A set of **Group by** toggle actions that allow you to group the messages according to a selected criteria so that they can be presented in a hierarchical layout. The criteria used for grouping can be the severity of the errors (error, warning, info message, etc.), the resource name, the description of the message, and so on.

Ungroup all

Removes the grouping rules so that the messages are presented in a continuous list.

Show group columns

If any of the **Group by** options are selected, you can use this option to show or hide grouping columns.

Restore Defaults

Restores the column size for each column and the grouping rules that were saved in the user preferences the last time when this view was used. If it is the first time this view is used, the action sets an initial default column size for each column and a grouping rule that is appropriate for the type of messages. For example:

- Group the messages by the path of the validated file if there are error messages from a validation action or spelling errors reported by the ***Check Spelling in Files*** *action*.
- No grouping rule for the results of *applying an XPath expression*.

Expand All

Expands all the nodes of the tree, which is useful when the messages are presented in a hierarchical mode.

Collapse All

Collapses all the nodes of the tree, which is useful when the messages are presented in a hierarchical mode.

CSS Inspector View

The purpose of the **CSS Inspector** view is to display information about the styles applied to the currently selected element. You can use this view to examine the structure and layout of the CSS rules that match the element. The matching rules displayed in this view include a link to the line in the CSS file that defines the styles. With this tool you can see how the CSS rules were applied and the properties defined, and use the link to open the associated CSS for editing purposes.

Figure 94: CSS Inspector View

Displaying the CSS Inspector View

You can open this view by selecting the **Inspect Styles** action from the contextual menu in **Author** mode, or selecting the **CSS Inspector** view in the **Window** > **Show View** menu. This action makes the view visible and also initializes it for the currently selected element.

Displaying Rules

All rules that apply to the current element are displayed in sections, which are listed in order of importance (from most specific to least specific). Rules that are overridden by other rules are crossed out. If you click the link in the top-right corner of a rule Oxygen XML Editor opens the associated CSS file at the line number where the properties of the rule are defined.

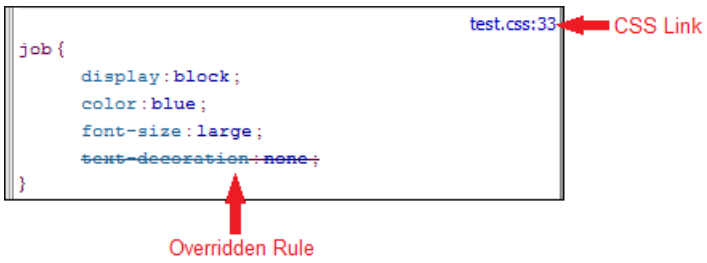


Figure 95: CSS Inspector View - Displaying Rules

The **CSS Inspector** view contains five tabs:

- **Element** - Displays the CSS rules matching the currently selected element in the **Author** page (ordered from most-specific to least-specific).
- **:before** - Displays the rules matching the `:before` pseudo-element.
- **:after** - Displays the rules matching the `:after` pseudo-element.
- **Computed** - Displays all the styling properties that apply to the current element, as a result of all the CSS rules matching the element.

- **Path** - Displays the path for the current element, and its attributes, allowing you to quickly see the attributes on all parent elements, and allows you to copy fragments from this view and paste it into the associated CSS to easily create new rules.

The information displayed in each of the five tabs is updated when you click other elements in the **Author** editing view. The first three tabs include the link to the associated CSS source, while the other two tabs simply display the style properties that match the current element.

Each of the tabbed panes include a contextual menu with the following actions:

- **Copy** - copies the current selection
- **Select all** - selects all information listed in the pane
- **Show empty rules** - forces the view to show all the matching rules, even if they do not declare any CSS properties (by default, the empty rules are not displayed)

Displaying the Markup

You can control the amount of markup that is displayed in the **Author** mode with various levels of tag modes for both *block* and *in-line* elements.

The following dedicated tag modes are available from the **Tags Display Mode** drop-down menu (available on the toolbar):

Full Tags with Attributes

Displays full tag names with attributes for both *block* and *inline* elements.

Full Tags

Displays full tag names without attributes for both *block* and *inline* elements.

Block Tags

Displays full tag names for *block elements* and simple tags without names for *inline elements*.

Inline Tags

Displays full tag names for *inline elements*, while *block elements* are not displayed.

Partial Tags

Displays simple tags without names for *inline elements*, while *block elements* are not displayed.

No Tags

No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Configure Tags Display Mode

Opens the ***Author*** *preferences page* where you can configure options in regards to *tags*, such as the default **Tags Display Mode**, **Tags Background Color**, **Tags Foreground Color**, **Tags Font**, and whether or not Oxygen XML Editor will use a **Compact Tag Layout** for displaying the tags (this option tries to group consecutive block tags on the same line).

Note: The associated CSS information is used to determine whether a tag should be considered *inline* or *block*. If the current document does not have an associated CSS stylesheet, then the **Full Tags** mode will be used.

Displaying Referenced Content

The references to entities, XInclude, and DITA conrefs are expanded by default in **Author** mode and the referenced content is displayed. You can control this behavior from the *Author preferences page*. The referenced resources are loaded and displayed inside the element or entity that refers them, however the displayed content cannot be modified directly.

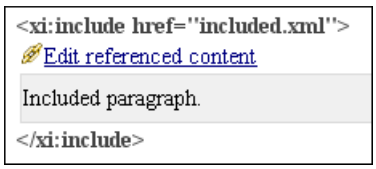


Figure 96: XInclude reference

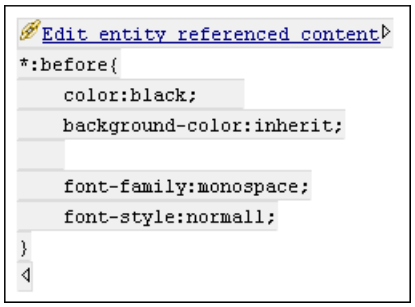


Figure 97: External entity reference

When the referenced resource cannot be resolved, an error will be presented inside the element that refers them instead of the content.

If you want to make modifications to the referenced content, you must open the referenced resource in an editor. The referenced resource can be opened quickly by clicking the link (marked with the icon) that is displayed before the referenced content or by using the **Edit Reference** action from the contextual menu (in this case the cursor is placed at the precise location where the action was invoked). The referenced resource is resolved through the *XML Catalog* set in **Preferences**.

The referenced content is refreshed:

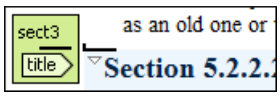
- Automatically, when it is modified and saved from Oxygen XML Editor.
- On demand, by using the *Refresh references action*. Useful when the referenced content is modified outside the Oxygen XML Editor scope.

Visual Hints for the Cursor Position

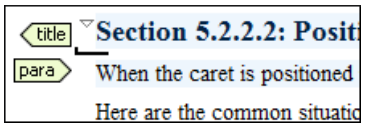
When the cursor is positioned inside a new context, a tooltip will be shown for a couple of seconds displaying the position of the cursor relative to the context of the current element.

Here are some of the common situations that can be encountered:

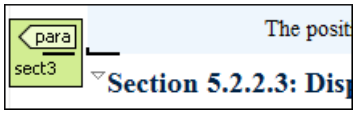
- **Before first *block*** - The cursor is positioned before the first *block* child of the current node.



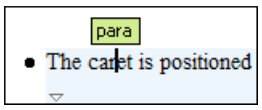
- **Between two *block elements*** - The cursor is positioned between two *block elements*.



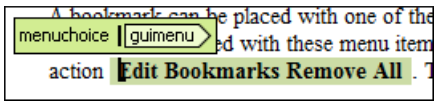
- **After last *block*** - The cursor is positioned after the last *block element* child of the current node.



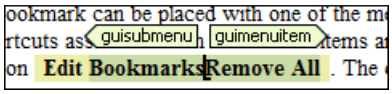
- **Inside a node** - The cursor is positioned inside a node.



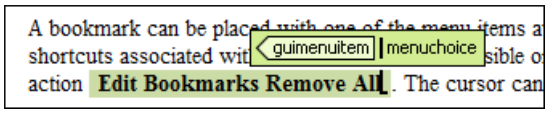
- **Before an *inline element*** - The cursor is positioned inside an element, before a child *inline element*.



- **Between two inline elements** - The cursor is positioned between two *inline elements*.



- **After an inline element** - The cursor is positioned inside an element, after an child *inline element*.



The nodes in these cases are displayed in the tooltip window using the element names.

To deactivate this feature, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Author** > **Cursor Navigation**, and deselect the ***Show cursor position tooltip** option*. Even if this option is deselected, you can still display the position tooltip by pressing **Shift+F2**.

Note: The position information tooltip is not displayed if ***Full Tags with Attributes** or **Full Tags** is selected in the **Tags display mode** drop-down menu*.

Location Tooltip

When editing XML documents in a visual environment, you might find it difficult to position the cursor between certain tags that do not have a visual representation. To counterbalance this, Oxygen XML Editor displays a transparent preview of the position information, called the **Location Tooltip**:

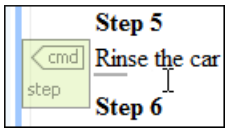


Figure 98: Location Tooltip

Oxygen XML Editor displays a **Location Tooltip** when the following conditions are met:

- You are editing the document in one of the following *tags display modes*: **Inline Tags**, **Partial Tags**, **No Tags**.
- The mouse pointer is moved between *block elements*.

To activate or deactivate this feature, use the ***Show location tooltip on mouse move** option* in the ***Cursor Navigation** preferences page*.

Presenting Validation Errors in Author Mode

Oxygen XML Editor can be configured to *automatically validate documents* while editing in the **Author** mode, and actions are also available to *manually validate documents* on-request.

In **Author** mode, validation issues are marked in the following locations:

- In the main editing pane, with the issue underlined in a color according to the type of issue.
- In the right-side vertical stripe, with a marker that is colored according to the type of issue.
- For attributes with detected issues, in the ***Attributes** view*, with the attribute and its value colored according to the type of issue.

The colors for each type of issue are as follows:

- **Validation Errors** [Red] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the **Attributes** view are colored in red.
- **Validation Warnings** [Yellow] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the **Attributes** view are colored in yellow.
- **Validation Info** [Blue] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the **Attributes** view are colored in blue.

You can configure the color for each type in the ***Document Checking** preferences page*.

Hovering over a validation issue presents a tooltip message with more details about the problem and *possible quick fixes* (if available for that issue).

Information about the issue is also displayed in the message area on the bottom of the editor panel (clicking the **Document checking options** button opens the ***Document Checking*** *preferences page* where you can configure some validation options (such as the colors used to present the validation issues). Some validation messages include an icon ( ) that provides a link to a style guide or specification.

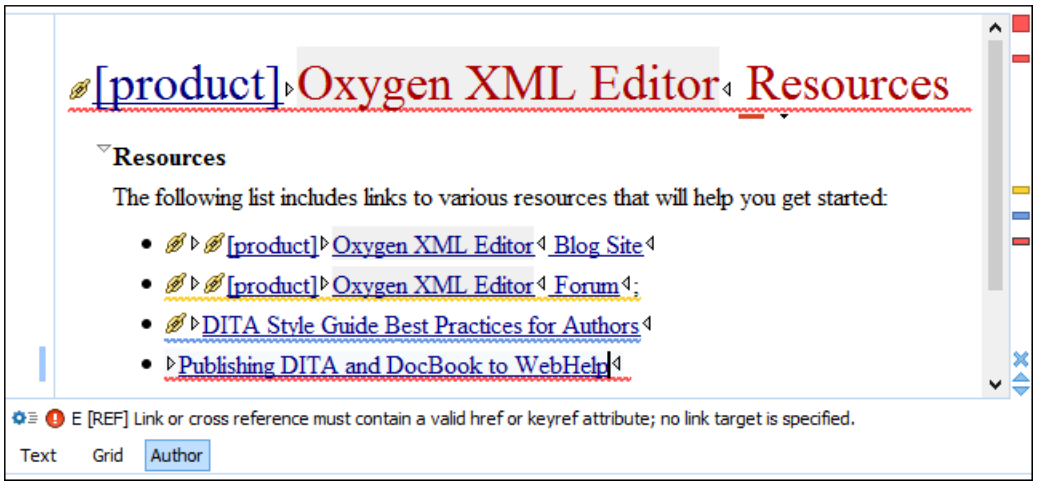


Figure 99: Presenting Validation Errors in Author Mode

Also, the stripe on the right side of the editor panel is designed to display the issues found during the validation process and to help you locate them in the document. The stripe contains the following:

Upper Part of the Stripe

A success indicator square will turn green if the validation is successful or only info messages are found, red if validation errors are found, or yellow if only validation warnings are found. More details about the issues are displayed in a tool tip when you hover over indicator square. If there are numerous problems, only the first three are presented in the tool tip.

Middle Part of the Stripe

Errors are depicted with red markers, warnings with yellow markers, and info message with blue markers. If you want to limit the number of markers that are displayed, *open the* ***Preferences*** *dialog box* ***(Options > Preferences)***, go to **Editor** > **Document checking**, and specify the desired limit in the ***Maximum number of validation highlights*** *option*.

Clicking a marker will highlight the corresponding text area in the editor. The validation message is also displayed both in a tool tip (when hovering over the marker) and in the message area on the bottom of the editor panel (clicking the **Document checking options** button opens the ***Document Checking*** *preferences page*.

Bottom Part of the Stripe

Two navigation arrows ( ) allow you to jump to the next or previous issue. The same actions can be triggered from **Document** > **Automatic validation** > **Next error** (**Ctrl + Period (Command + Period on OS X)**) and **Document** > **Automatic validation** > **Previous error** (**Ctrl + Comma (Command + Comma on OS X)**). Also, the button can be used to clear all the validation markers.

Status messages from every validation action are also logged in the ***Information*** *view*.

If you want to see all the validation messages grouped in the ***Results*** *panel*, you should use the **Validate** action from the toolbar or **Document** > **Validate** menu..

Related Information:

Validating XML Documents Against a Schema on page 446
Presenting Schematron Validation Issues

Whitespace Handling in Author Mode

When you edit a document in **Author** mode, Oxygen XML Editor must serialize the resulting document as XML. Oxygen XML Editor serializes the document when you save it or switch to another editing mode. When the document is serialized, Oxygen XML Editor *formats and indents the XML document* according to the current *format and indent settings*.

Minimizing whitespace differences between versions

When serializing a document to XML, **Author** mode will only format and indent those elements of the document that have been edited. Any element that has not been edited will be serialized exactly as it was loaded from disk. This is useful when your content is managed in a version control systems, as it avoids introducing insignificant whitespace differences between version, which in turn makes diff output easier to read.

Entering whitespace in Author mode

Oxygen XML Editor controls the entry of whitespace characters in **Author** mode according the *XML whitespace rules*, which means it will not let you insert insignificant whitespace. This means that it will not let you insert extra line-breaks or spaces inside a typical paragraph element, for instance. (Any such whitespace would be normalized away when the document was serialized to XML, so Oxygen XML Editor is saving you from any surprises when this happens.)

Of course, you will legitimately want to enter additional spaces and returns in some cases, such as code samples. Oxygen XML Editor will allow this in elements that are configured as preserve space elements according to the XML whitespace rules. For all of its *predefined document types*, Oxygen XML Editor is *correctly configured to recognize preserve space elements* and to allow you to enter additional spaces in them.

If you are using a predefined document type and you are unable to enter additional whitespace, make sure that you are using an element from that document type that is intended to be a preserve-space element.

If you are using a custom document type, make sure that it is *configured correctly* so that Oxygen XML Editor recognizes that the current element is a preserve-space element.

Bidirectional Text Support in Author Mode

Oxygen XML Editor offers support for languages that require right to left scripts. This means that authors editing documents in the **Author** mode can create and edit XML content in Arabic, Hebrew, Persian and others. To achieve this, Oxygen XML Editor implements the *Unicode Bidirectional Algorithm*, as specified by the Unicode consortium. The text arrangement is similar to what you get in a modern HTML browser. The final text layout is rendered according to the directional CSS properties matching the XML elements and the Unicode directional formatting codes.

By default, when navigating bidirectional text with the arrow keys in **Author** mode, pressing the right arrow key moves the cursor in the writing direction and the left arrow moves it in the opposite direction. However, if the ***Arrow keys move the cursor in the writing direction*** *option* in the **Cursor Navigation** preferences page is not selected, pressing the right arrow will simply move the cursor to the right (and the left arrow moves it to the left), regardless of the text direction.

If bidirectional text (such as Arabic or Hebrew languages), certain Asian languages (such as Devanagari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telugu, Kannada, Malayalam, Sinhala, Thai, Khmer) or other special characters (such as combining characters), are detected in a document, Oxygen XML Editor displays a **Special Characters Detected** dialog box that prompts you to **Enable** or **Disable** support for these special characters.

To watch our video demonstration about the bidirectional text support in the **Author** mode, go to *https://www.oxygenxml.com/demo/BIDI_Support.html*.

Tip: If you experience performance issues when editing documents that contain bidirectional text, you could try one of the following solutions:

- The Eclipse plugin distribution of Oxygen XML Editor is faster than the standalone version when working with bidirectional text.
- Changing the font. For example, you could try using the**David** font in Hebrew content. If it is not already installed in your operating system, this font is available at: *https://www.microsoft.com/typography/fonts/*

family.aspx?FID=234. To change the font in Oxygen XML Editor, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Appearance** > **Fonts**, and change the font in the ***Author default font** option*.

Related Information:

Bidirectional Text Support in Text Mode on page 207
Bidirectional Text Support in Grid Mode on page 210
Inserting Symbols

Controlling the Text Direction Using XML Markup

Oxygen XML Editor Supports the following CSS properties:

Table 6: CSS Properties Controlling Text Direction

direction	Specifies the writing direction of the text. The possible values are *ltr* (the text direction is left to right), *rtl* (the text direction is right to left, and *inherit* (the direction property is inherited from the parent element).
unicodeBidi	Used along with the *direction* property to create levels of embedded text with different text directions in the same document. The possible values of this property are *bidi-override* (creates an additional level of embedding and forces all strong characters to the direction specified in the *direction*), *embed* (creates an additional level of embedding), *normal* (does not use an additional level of embedding), and *inherit* (the value of the *unicodeBidi* property is inherited from parent element).

For instance, to declare an element as being Right to Left, you could use a stylesheet like this:

XML File:

```
<article>
  <myRTLpara>RIGHT TO LEFT TEXT</myRTLPara>
</article>
```

Associated CSS File:

```
myRTLpara{
    direction:rtl;
    unicode-bidi:embed;
}
```

Oxygen XML Editor recognizes the *dir* attribute on any XML document. The supported values are:

ltr	The text from the current element is Left to Right, embedded.
rtl	The text from the current element is Right to Left, embedded.
lro	The text from the current element is Left to Right, embedded.
rlo	The text from the current element is Right to Left, embedded.

The following XML document types make use of the *dir* attribute with the above values:

- DITA
- DocBook
- TEI
- XHTML

Note: When the *inline element* tags are visible, the text in the line is arranged according to the BIDI algorithm after replacing the tags symbols with Object Replacement Characters. This makes it possible to get a different text arrangement when viewing a document in the **No Tags** mode versus viewing it in the **Full Tags** mode.

Controlling the Text Direction Using the Unicode Direction Formatting Codes

These Unicode Direction Formatting Codes codes can be embedded in the edited text, specifying a text direction and embedding. However, it is not recommended to use them in XML as they are zero width characters, making it hard to debug the text arrangement.

Table 7: Directional Formatting Codes

U+202A (LRE)	LEFT-TO-RIGHT EMBEDDING	Treats the following text as embedded left-to-right.
U+202B (RLE)	RIGHT-TO-LEFT EMBEDDING	Treats the following text as embedded right to left.
U+202D (LRO)	LEFT-TO-RIGHT OVERRIDE	Forces the following characters to be treated as strong left-to-right characters.
U+202E (RLO)	RIGHT-TO-LEFT OVERRIDE	Forces the following characters to be treated as strong right-to-left characters.
U+202C (PDF)	POP DIRECTIONAL FORMATTING CODE	Restores the bidirectional state to what it was before the last LRE, RLE, RLO, or LRO.
U+200E (LRM)	LEFT-TO-RIGHT MARK	Left-to-right strong zero-width character.
U+200F (RLM)	RIGHT-TO-LEFT MARK	Right-to-left strong zero-width character.

To insert Unicode Direction Formatting Codes, use the ***Character Map*** dialog box. To easily find such a code, you can either enter directly the hexadecimal value, or use the **Details** tab to enter the codes name.

Oxygen XML Editor offers the support for bi-directional text in all the side views (***Outline*** *view*, ***Attributes*** *view* and so on) and text fields.

Design Editing Mode (XML Schema Diagram Editor)

XML Schemas allow document designers to specify the allowed structure and content of an XML document and to check if an XML document is valid.

Oxygen XML Editor provides a simple and expressive XML Schema diagram editor (**Design** mode) for editing XML Schemas. The schema diagram helps both the content authors who want to understand a schema and schema designers who develop complex schemas.

The diagram font can be increased using the usual Oxygen XML Editor shortcuts: **(Ctrl (Meta on Mac OS) + "+")** , **(Ctrl (Meta on Mac OS)+"-")** , **(Ctrl (Meta on Mac OS) + 0)** or **(Ctrl (Meta on Mac OS) - mouse wheel)**. The whole diagram can also be zoomed with one of the predefined factors *available in the Schema preferences panel*. The same zoom factor is applied for the print and save actions.

To switch to this mode, select **Design** at the bottom of the editing area.

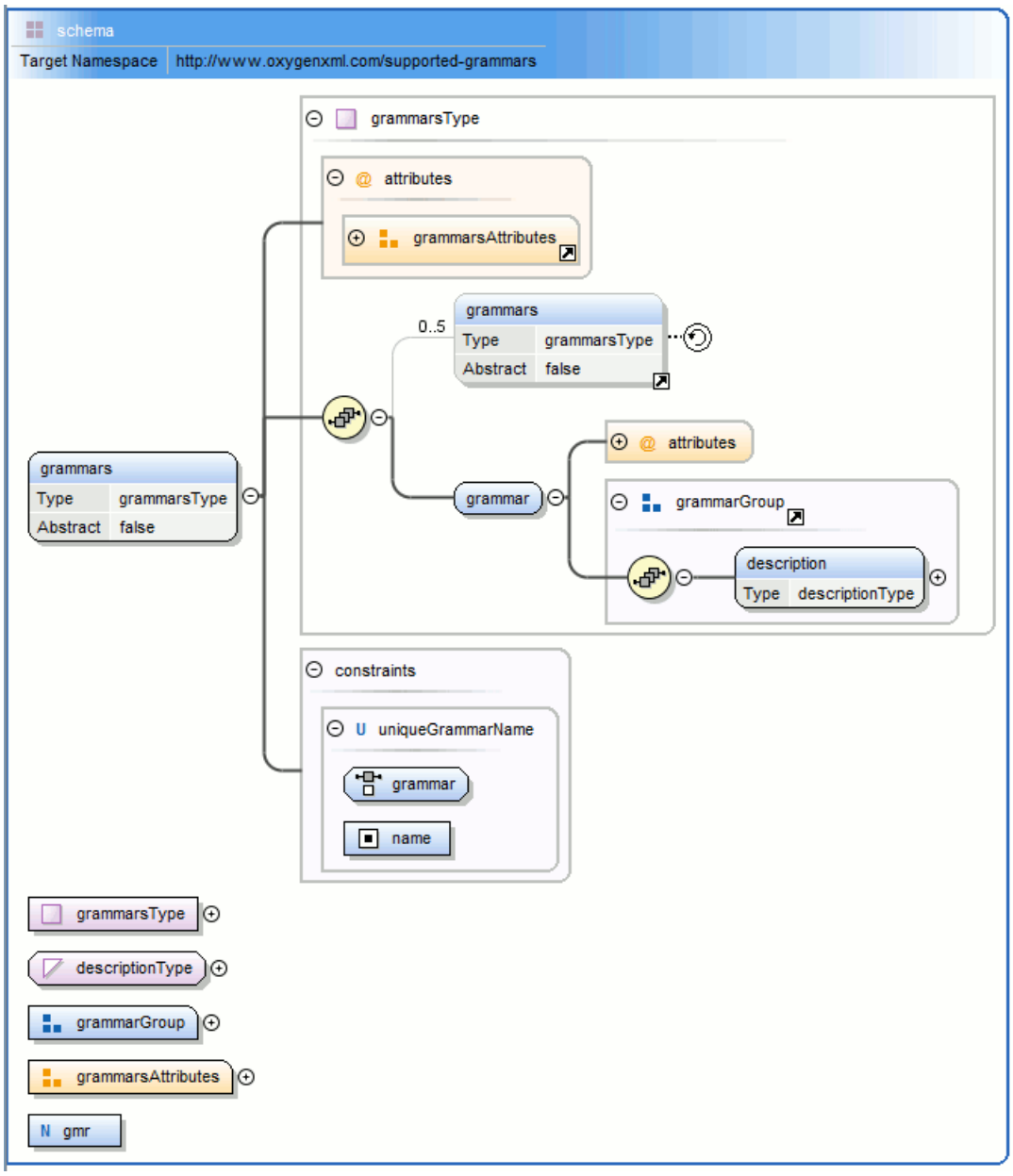


Figure 100: XML Schema Diagram

To watch our video demonstration about the basic aspects of designing an XML Schema using the new Schema Editor, go to *https://www.oxygenxml.com/demo/XML_Schema_Editing.html*.

Navigation in the XML Schema Design Mode

The following editing and navigation features work for all types of schema components in the XML Schema **Design** mode:

- Move/reference components in the diagram using drag-and-drop actions.
- Select consecutive components on the diagram (components from the same level) using the *Shift* key. You can also make discontinuous selections in the schema diagram using the **Ctrl (Meta on Mac OS)** key. To deselect one of the components, use **Ctrl + Single-Click (Command + Single-Click on OS X)**.
- Use the arrow keys to navigate the diagram vertically and horizontally.
- Use *Home*/*End* keys to jump to the first/last component from the same level. Use **Ctrl + Home (Command + Home on OS X)** key combination to go to the diagram root and **Ctrl + End (Command + End on OS X)** to go to the last child of the selected component.

- You can easily go back to a previously visited component while moving from left to right. The path will be preserved only if you use the left arrow key or right arrow key. For example, if the current selection is on the second attribute from an attribute group and you press the left arrow key to jump to the attribute group, when you press the right arrow key, then the selection will be moved to the second attribute.
- Go back and forward between components viewed or edited in the diagram by selecting them in the ***Outline view***:
  - **Back** (go to previous schema component).
  - **Forward** (go to next schema component).
  - **Go to Last Modification** (go to last modified schema component).
- Copy, reference, or move global components, attributes, and identity constraints to another position and from one schema to another using the **Cut/Copy** and **Paste/Paste as Reference** actions.
- Go to the definition of an element or attribute with the **Show Definition** action.
- Search in the diagram using the *Find/Replace dialog box* or the *Quick find toolbar*. You can find/replace components only in the current file scope.
- You can expand and see the contents of the imports/includes/redefines in the diagram. In order to edit components from other schemas the schema for each component will be opened as a separate file in Oxygen XML Editor.

  Tip: If an XML Schema referenced by the current opened schema was modified on disk, the change will be detected and you will be asked to refresh the current schema contents.
- Recursive references are marked with a *recurse symbol* ( ). Click this symbol to navigate between the element declaration and its reference.

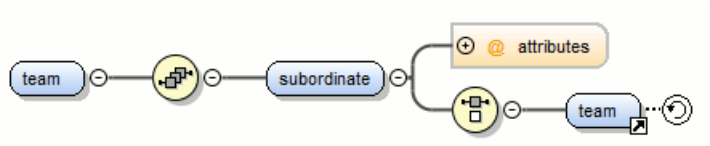


Figure 101: Recursive Reference

XML Schema Outline View

The **Outline** view for XML Schemas presents all the global components grouped by their location, namespace, or type. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

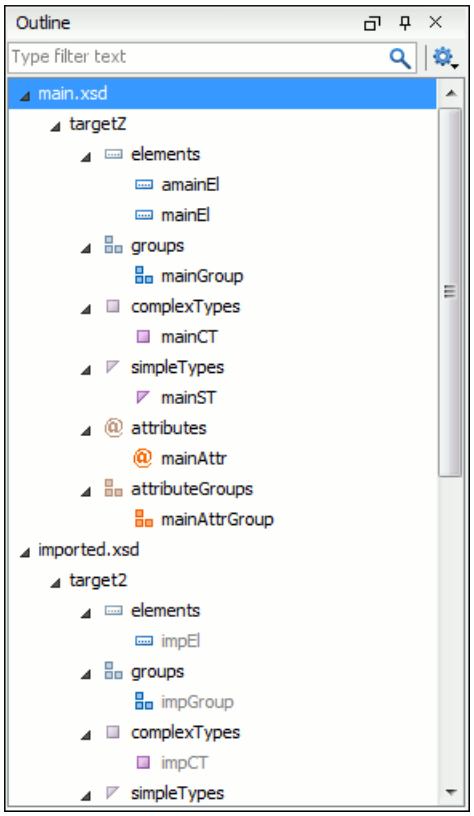


Figure 102: Outline View for XML Schema

The **Outline** view provides the following options in the **Settings** menu on the **Outline** view toolbar:

Filter returns exact matches

The text filter of the **Outline** view returns only exact matches;

Selection update on cursor move

Allows a synchronization between **Outline** view and schema diagram. The selected view from the diagram is also selected in the **Outline** view.

Sort

Allows you to sort alphabetically the schema components.

Show all components

Displays all components that were collected starting from the *master files*. Components that are not referable from the current file are marked with an orange underline. To reference them, add an import directive with the `componentNS` namespace.

Show referable components

Displays all components (collected starting from the *master files*) that can be referenced from the current file. This option is set by default.

Show only local components

Displays the components defined in the current file only.

Group by location/namespace/type

These three operations allow you to group the components by location, namespace, or type. When grouping by namespace, the main schema target namespace is the first presented in the **Outline** view.

The following contextual menu actions are available in the **Outline** view:

Remove (Delete)

Removes the selected item from the diagram.

Search References (Ctrl (Meta on Mac OS)+Shift+R)

Searches all references of the item found at current cursor position in the defined scope, if any.

Search References in

Searches all references of the item found at current cursor position in the specified scope.

Component Dependencies (Ctrl (Meta on Mac OS)+Shift+F4)

Opens the ***Component Dependencies*** *view* that allows you to see the dependencies for the current selected component.

Resource Hierarchy (F4)

Opens the ***Resource Hierarchy / Dependencies*** *view* that allows you to see the hierarchy for the current selected resource.

Resource Dependencies (Shift + F4)

Opens the ***Resource Hierarchy / Dependencies*** *view* that allows you to see the dependencies for the current selected resource.

Rename Component in

Renames the selected component.

Generate Sample XML Files

Generate XML files using the current opened schema. The selected component is the XML document root.

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (*, ?) and separate multiple patterns with commas.

Tip: The search filter is case insensitive. The following wildcards are accepted:

- ***** - any string
- **?** - any character
- **,** - patterns separator

If no wildcards are specified, the string to search will be searched as a partial match.

The content of the **Outline** view and the editing area are synchronized. When you select a component in the **Outline** view, its definition is highlighted in the editing area.

Related Information:

Searching and Refactoring Actions in XML Schemas on page 595
Component Dependencies View for XML Schema on page 593
XML Schema Resource Hierarchy / Dependencies View on page 591
Generating Sample XML Files on page 597
Editing Relax NG Schema in the Master Files Context on page 655

XML Schema Attributes View

The **Attributes** view for XML Schemas presents the properties for the selected component in the schema diagram. By default, it is displayed on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

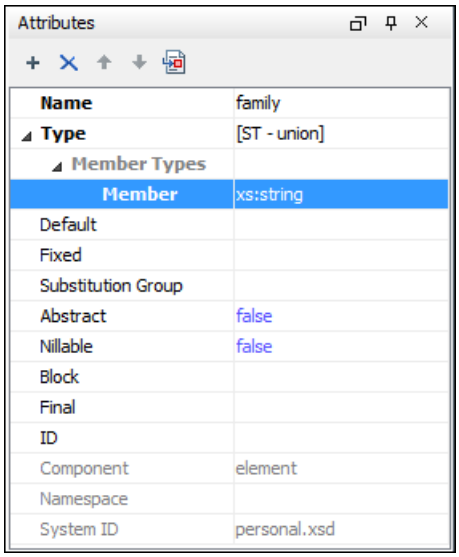


Figure 103: Attributes View

The default value of a property is presented in the **Attributes** view with blue foreground. The properties that can not be edited are rendered with gray foreground. A non-editable category that contains at least one child is rendered with bold. Bold properties are properties with values set explicitly to them.

Properties for components that do not belong to the current edited schema are read-only but if you double-click them you can choose to open the corresponding schema and edit them.

You can edit a property by double-clicking by pressing Enter. For most properties you can choose valid values from a list or you can specify another value. If a property has an invalid value or a warning, it will be highlighted in the table with the corresponding foreground color. By default, properties with errors are highlighted with red and the properties with warnings are highlighted with yellow. You can customize these colors from the *Document checking user preferences*.

For imports, includes and redefines, the properties are not edited directly in the **Attributes** view. A dialog box will open that allows you to specify properties for them.

The schema namespace mappings are not presented in **Attributes** view. You can view/edit these by choosing **Edit Schema Namespaces** from the contextual menu on the schema root. See more in the *Edit Schema Namespaces* section.

The **Attributes** view has five actions available on the toolbar and also on the contextual menu:

Add

Allows you to add a new member type to an union's member types category.

Remove

Allows you to remove the value of a property.

Move Up

Allows you to move up the current member to an union's member types category.

Move Down

Allows you to move down the current member to an union's member types category.

Copy

Copy the attribute value.

Show Definition<u>**Ctrl (Meta on MAC OS) + Click**</u>

Shows the definition for the selected type.

Show Facets

Allows you to edit the facets for a simple type.

XML Schema Facets View

The **Facets** view for XML Schemas presents the facets for the selected component, if available. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

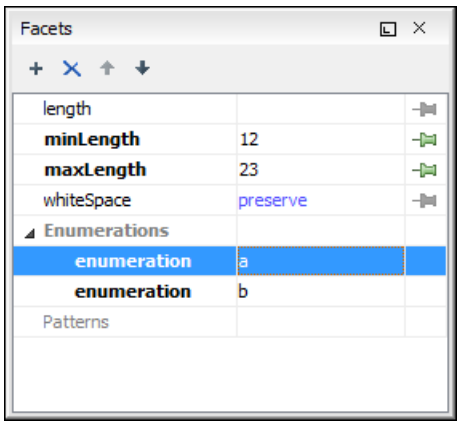


Figure 104: Facets View

The default value of a facet is rendered in the **Facets** view with a blue color. The facets that can not be edited are rendered with a gray color. The grouping categories (for example: **Enumerations** and **Patterns**) are not editable. If these categories contain at least one child they are rendered with bold. Bold facets are facets with values set explicitly to them.

Important: Usually inherited facets are presented as default in the **Facets** view but if patterns are inherited from a base type and also specified in the current simple type only the current specified patterns will be presented. You can see the effective pattern value obtained by combining the inherited and the specified patterns as a tooltip on the **Patterns** category.

Facets for components that do not belong to the current edited schema are read-only but if you double-click them you can choose to open the corresponding schema and edit them.

You can edit a facet by double-clicking it or by pressing Enter, when that facet is selected. For some facets you can choose valid values from a list or you can specify another value. If a facet has an invalid value or a warning, it will be highlighted in the table with the corresponding foreground color. By default, facets with errors are presented with red and the facets with warnings with yellow. You can customize the error colors from the *Document Checking user preferences*.

The **Facets** view provides the following actions in its toolbar and contextual menu:

Add

Allows you to add a new enumeration or a new pattern.

Remove

Allows you to remove the value of a facet.

Edit Annotations

Allows you to edit an annotation for the selected facet.

Move Up

Allows you to move up the current enumeration/pattern in **Enumerations/Patterns** category.

Move Down

Allows you to move down the current enumeration/pattern in **Enumerations/Patterns** category.

Copy

Copy the attribute value.

Open in Regular Expressions Builder

Rather than editing regular expressions manually, this action allows you to open the pattern in the *XML Schema Regular Expressions Builder* that guides you through the process of testing and constructing the pattern..

Facets can be fixed to prevent a derivation from modifying its value. To fix a facet value just press the **Pin** button.

XML Schema Palette View

The **Palette** view is designed to offer quick access to XML Schema components and to improve the usability of the XML Schema diagram builder. You can use the **Palette** to drag and drop components in the **Design** mode. The components offered in the **Palette** view depend on the XML schema version set in the ***XML Schema*** *preferences page*. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

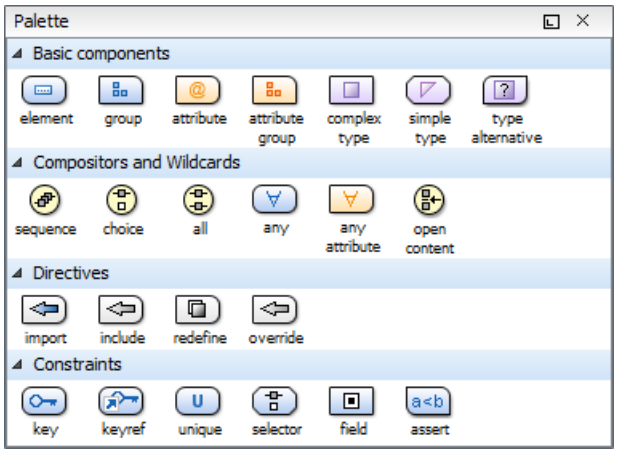


Figure 105: Palette View

Components are organized functionally into 4 collapsible categories:

- Basic components: *elements*, *group*, *attribute*, *attribute group*, *complex type*, *simple type*, *type alternative*.
- Compositors and Wildcards: *sequence*, *choice*, *all*, *any*, *any attribute*, *open content*.
- Directives: *import*, *include*, *redefine*, *override*.
- Identity constraints: *key*, *keyref*, *unique*, *selector*, *field*, *assert*.

Note: The *type alternative*, *open content*, *override*, and *assert* components are available for XML Schema 1.1.

To add a component to the edited schema:

- Click and hold a graphic symbol from the **Palette** view, then drag the component into the **Design** view.
- A line dynamically connects the component with the XML schema structure.
- Release the component into a valid position.

Note: You cannot drop a component into an invalid position. When you hover the component into an invalid position, the mouse cursor changes its shape into . Also, the connector line changes its color from the usual dark gray to the color defined in the ***Validation error highlight color*** *option* (default color is red).

To watch our video demonstration about the Schema palette and developing XML Schemas, go to *https://www.oxygenxml.com/demo/Schema_Palette.html* and *https://www.oxygenxml.com/demo/Developing_XML_Schemas.html* respectively.

Editing Documents

7

Topics:

- *Working with Unicode*
- *Creating and Working with Documents*
- *Searching Documents*
- *Using Projects to Group Documents*
- *Editing XML Documents*
- *Editing XSLT Stylesheets*
- *Editing Ant Build Files*
- *Editing XML Schemas*
- *Editing XQuery Documents*
- *Editing WSDL Documents*
- *Editing CSS Stylesheets*
- *Editing LESS CSS Stylesheets*
- *Editing Relax NG Schemas*
- *Editing NVDL Schemas*
- *Editing JSON Documents*
- *Editing StratML Documents*
- *Editing XLIFF Documents*
- *Editing JavaScript Documents*
- *Editing XProc Scripts*
- *Editing Schematron Schemas*
- *Editing Schematron Quick Fixes*
- *Editing SVG Files*
- *Editing XHTML Documents*
- *Editing Markdown Documents*
- *Editing Non-XML Files*
- *Spell Checking*
- *AutoCorrect Misspelled Words*
- *Loading Large Documents*
- *Scratch Buffer*
- *Handling Read-Only Files*
- *Editing Documents with Long Lines*
- *XML Digital Signatures*
- *Compare Files or Directories*

This chapter includes a large amount of information about editing numerous types of documents, how to create and work with documents, working with projects, and various editing features that are provided in Oxygen XML Editor. Each type of document has unique features and options when editing them in Oxygen XML Editor, while some editing features are available for all document types. This chapter also includes information about some of the special tools that are included in Oxygen XML Editor, such as the file and directory comparison tools.

Working with Unicode

Unicode provides a unique number for every character, independent of the platform and language. Unicode is an internationally recognized standard, adopted by industry leaders. The Unicode is required by modern standards (such as XML, Java, JavaScript, LDAP, CORBA 3.0, WML, etc.) and is the official way to implement ISO/IEC 10646.

It is supported in many operating systems, all modern browsers, and many other products. The emergence of the Unicode Standard, and the availability of tools supporting it, are among the most significant recent global software technology trends. Incorporating Unicode into client-server or multiple tiered applications and websites offers significant cost savings over the use of legacy character sets.

As a modern XML Editor, Oxygen XML Editor provides support for the Unicode standard enabling your XML application to be targeted across multiple platforms, languages, and countries without re-engineering. Internally, the Oxygen XML Editor uses 16 bit characters covering the Unicode Character set.

As a Java application, Oxygen XML Editor includes a default Java input method for typing characters with Unicode codes. However, the default input method does not cover all the Unicode codes (for example, the codes for some accented characters or characters found in East Asian languages). Such characters can be inserted in the editor panel of Oxygen XML Editor either with *the **Character Map** dialog box* available from menu **Edit** > **Insert from Character Map** or by installing a Java input method that supports the insertion of the needed characters. The *installation of a Java input method* depends on the platform (Windows, Mac OS X, Linux, etc.) and is the same for any Java application.

Note: Oxygen XML Editor may not be able to display characters that are not supported by the operating system (either not installed or unavailable).

Tip: On windows, you can enable the support for **CJK** (Chinese, Japanese, Korean) languages from **Control Panel / Regional and Language Options / Languages / Install files for East Asian languages**.

Opening and Saving Documents with Unsupported Characters

When loading documents, Oxygen XML Editor reads the document prolog to determine the specified encoding type. This encoding is then used to instruct the Java Encoder to load support for and to save the document using the specified code chart. When the encoding type cannot be determined, Oxygen XML Editor prompts and display the **Available Java Encodings** dialog box that provides a list of all encodings supported by the Java platform.

If the opened document contains an unsupported character, Oxygen XML Editor applies *the policy specified for handling such errors.* If the policy is set to REPORT, Oxygen XML Editor displays an error dialog box with a message about the character not allowed by the encoding. If the policy is set to IGNORE, the character is removed from the document displayed in the editor panel. If the policy is set to REPLACE, the character is replaced with a standard replacement character for that encoding.

While in most cases you are using UTF-8, simply changing the encoding name causes the application to save the file using the new encoding.

When saving a document edited in the **Text**, **Grid**, or **Design** modes, if it contains characters not included in the encoding declared in the document prolog, Oxygen XML Editor detects the problem and signals it to the user. The user is responsible to resolve the conflict before saving the document.

When saving a document edited in the **Author** mode, all characters that fall outside the detected encoding will be automatically converted to hexadecimal character entities.

To edit documents written in Japanese or Chinese, change the font to one that supports the specific characters (a Unicode font). For the Windows platform, *Arial Unicode MS* or *MS Gothic* is recommended. Do not expect *WordPad* or *Notepad* to handle these encodings. Use applications such as *Internet Explorer* or *Word* to examine XML documents.

When a document with a UTF-16 encoding is edited and saved in Oxygen XML Editor, the saved document has a byte order mark (BOM) that specifies the byte order of the document content. The default byte order is platform-dependent. That means that a UTF-16 document created on a Windows platform (where the default byte order mark is *UnicodeLittle*) has a different BOM than a UTF-16 document created on a Mac OS platform (where the byte order mark is *UnicodeBig*). The byte order and the BOM of an existing document are preserved

when the document is edited and saved. This behavior can be changed in Oxygen XML Editor from the *Encoding preferences panel*.

Inserting Symbols

You can insert symbols by using the **Character Map** dialog box that can be opened with the **Edit** > **Insert from Character Map** action. If you have *chosen to display the **Symbols** toolbar* (in the ***Configure Toolbars** dialog box*), you can also use the **Symbols** drop-down menu to insert some of the most commonly used symbols and selecting **More symbols** from that menu will also open the **Character Map** dialog box.

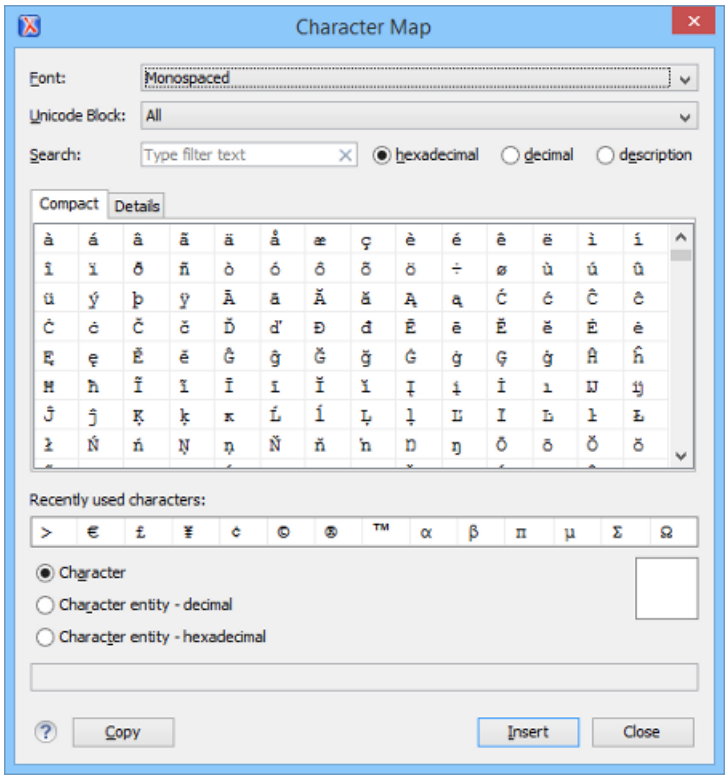


Figure 106: Character Map Dialog Box

The **Character Map** dialog box allows you to visualize all characters that are available in a particular font, pick the character you need, and insert it in the document you are editing. It includes the following fields and sections:

Font

Use this drop-down list to choose the font for which you want to display characters.

Unicode Block

Use this drop-down list to only see a certain range of characters. This will filter the number of characters displayed, showing only a contiguous range of characters corresponding to the selected block. Unassigned characters are displayed as empty squares.

Search

Use this filter to search for a character by one of the following attributes:

- **hexadecimal**
- **decimal**
- **description**

  Note: Selecting **description** opens the ***Details** tab*. If you enter a character description in the **Search** field, t**description** is selected automatically.

Character Table Section

The characters that are available to be inserted are listed in two tabs:

- **Compact** - Matrix-like table that displays a visual representation of the characters.

- **Details** - Displays the available characters in a tabular format, presenting their decimal and hexadecimal value along with their description.

Recently Used Characters Section

Displays the symbols that you have used recently and you can also select one from there to insert it in the current document.

Character Mode Section

The next section of the dialog box allows you to select how you want the character to appear in the **Text** editing mode. You can choose between the following:

- **Character**
- **Character entity - decimal**
- **Character entity - hexadecimal**

You can see the character or code that will be inserted in **Text** mode next to the selections in this section and a box on the right side of the dialog box allows you to see the character that will be inserted in **Author** mode. You can also see the name and range name of a character either at the bottom of the dialog box, or in a tooltip when hovering the cursor over the character.

Press the **Insert** button to insert the selected character in the current editor at cursor position. You will see the character in the editor if *the editor font* is able to render it. The **Copy** button copies it to the clipboard without inserting it in the editor.

Note: The **Character Map** dialog box cannot be used to insert Unicode characters in the ***Grid*** *editor*. Accordingly, the **Insert** button of the dialog box will be disabled if the current document is edited in **Grid** mode.

Unicode Fallback Font Support

Oxygen XML Editor provides fonts for most common Unicode ranges. However, if you use special symbols or characters that are not included in the default fonts, they will be rendered as small rectangles. A *fallback* font is a reserve typeface that contains symbols for as many Unicode characters as possible. When a display system encounters a character that is not part of the range of any of the available fonts, Oxygen XML Editor will try to find that symbol in a *fallback* font.

Example of a Scenario Where a Fallback Font is Needed

Suppose that you need to insert the wheelchair symbol (♿ - U+267F) into your content in a Windows operating system. By default, Oxygen XML Editor does not render this symbol correctly since it is not included in any of the default fonts. It is included in **Segoe UI Symbol**, but this font is not part of the default fonts that come with Oxygen XML Editor. To allow Oxygen XML Editor to recognize and render the symbol correctly, you can add **Segoe UI Symbol** as a *fallback* font.

Add a Fallback Font in Windows (7 or Later)

To add a fallback font to the Oxygen XML Editor installation, use the following procedure:

1. Start Windows Explorer and browse to the `[OXYGEN_INSTALL_DIR]/jre/lib/fonts` directory.
2. Create a directory called `fallback` (if it is not already there).
3. Copy a font file (True Type Font - TTF) that includes the special characters into this directory.

   Tip: You could, for example, copy the **Segoe UI Symbol Regular** font from `C:\Windows\Fonts`.
4. Restart Oxygen XML Editor for the changes to take full effect.

Result: Whenever Oxygen XML Editor finds a character that cannot be rendered using its standard fonts, it will look for the glyph in the fonts stored in the `fallback` folder.

Alternate Solution for Other Platforms

For Mac OS X or other platforms, you could use the following approach:

1. Use a font editor (such as *FontForge*) to combine multiple true type fonts into a single custom font.
2. Install the font file into the dedicated font folder of your operating system.

3. In Oxygen XML Editor, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Appearance** > **Fonts**.
4. Click the **Choose** button in the **Editor** option and select your custom font from the drop-down list in the subsequent dialog box.
5. Restart Oxygen XML Editor for the font changes to take full effect.

Creating and Working with Documents

Oxygen XML Editor includes various features, actions, and wizards to assist you with creating new files and working with existing files. This section explains many of these features, including information on creating new documents, opening, saving, and closing existing files, searching documents, viewing file properties, and more.

Creating New Documents and Templates

Oxygen XML Editor includes a handy **New Document** wizard that allows you to customize and create new files from a large list of document types and predefined new file templates. You can also create your own templates and share them with others.

New Document Wizard

Oxygen XML Editor supports a wide range of document types. The **New Document** wizard presents the default associations between a file extension and the type of editor that opens the file. To customize these default associations, *open the **Preferences** dialog box **(Options > Preferences)*** and go to ***File Types***.

The **New Document** wizard only creates a skeleton document. It may contain a root element, the document prolog, and possibly other child elements depending on options that are specific for each schema type.

New Document Wizard

The **New Document** wizard allows you to create various types of documents and provides some options that help you to configure the new document. To use this wizard to create a new document in Oxygen XML Editor, follow these steps:

1. Click the **New** button on the toolbar or select **File** > **New**.

   Result: The **New Document** wizard is displayed:

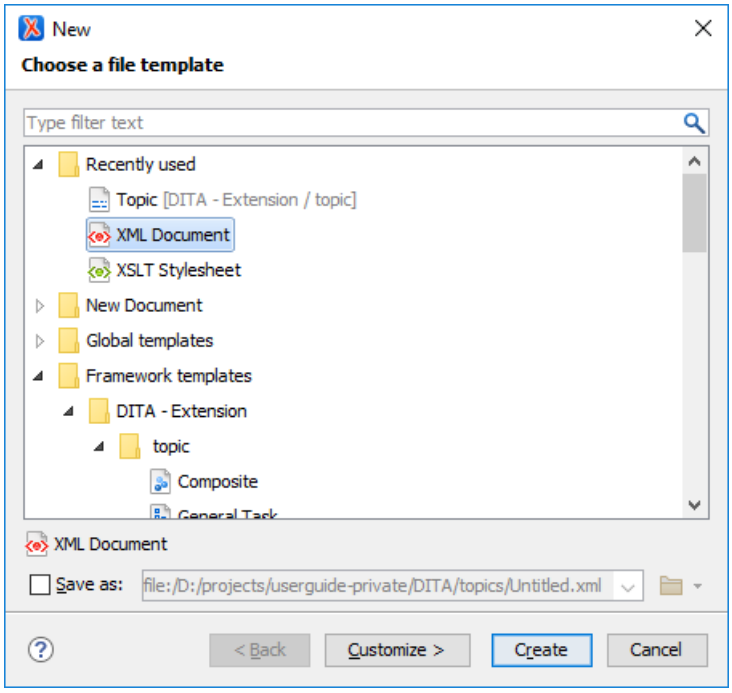


Figure 107: New Document Wizard

The first page of the wizard displays the supported document types and groups them in the following categories:

- **Recently Used** - Contains the list of the most recently used file types.
- **New Document** - Contains the list of all supported document types. This list includes XML, XSL, XML Schema, Document Type Definition, Relax NG Schema, XQuery, web Services Definition Language, Schematron Schema, CSS, Text, PHP, JavaScript, Java, C, C++, Batch, Shell, Properties, SQL, *XML Catalog*, PERL, and more.
- **Global Templates** - Contains the list of predefined templates as well as user-defined custom templates. You can *create your own custom file templates* and add them to the `templates` folder of the Oxygen XML Editor installation directory. You can also specify an additional directory to use for the templates in the ***Document Templates*** *preferences page*.
- **Framework Templates** - Contains the list of templates defined in the ***Document Type*** *configuration dialog box (**Templates** tab)* for each *framework*.

2. Select the type of document that you want to create.

   Tip: You can use the text filter field at the top of the dialog box to search for a specific template.
3. If you want to change the default name and path of the file, select the **Save as** option and specify the file path (the ***Show "Save as" option to save newly created documents in the "New" document wizard*** *option* must be selected in the **Open/Save** preferences page). Otherwise, the file will be opened in a new tab with a default *untitled* name and the document path will not yet exist until you save it.

   Note: For DITA documents, the dialog box includes some additional options for generating a title, file name, and root ID attribute. For more information, see *Creating a New DITA Topic* on page 1618.
4. If you want to use the default settings in the creation process, select **Create** at the bottom of the dialog box.

   Result: The document is created using the default settings and the new file is opened in the appropriate editor.
5. If you want to configure properties before creating the file, select **Customize**. This action is available for XML, XML Schema, Schematron, and XSL documents.

   Result: A new file configuration dialog box is opened that allows you to customize various options, depending on the document type you selected. After configuring the options in this wizard, click **Create** to create the file and open it in the appropriate editor.

XML Document File Type

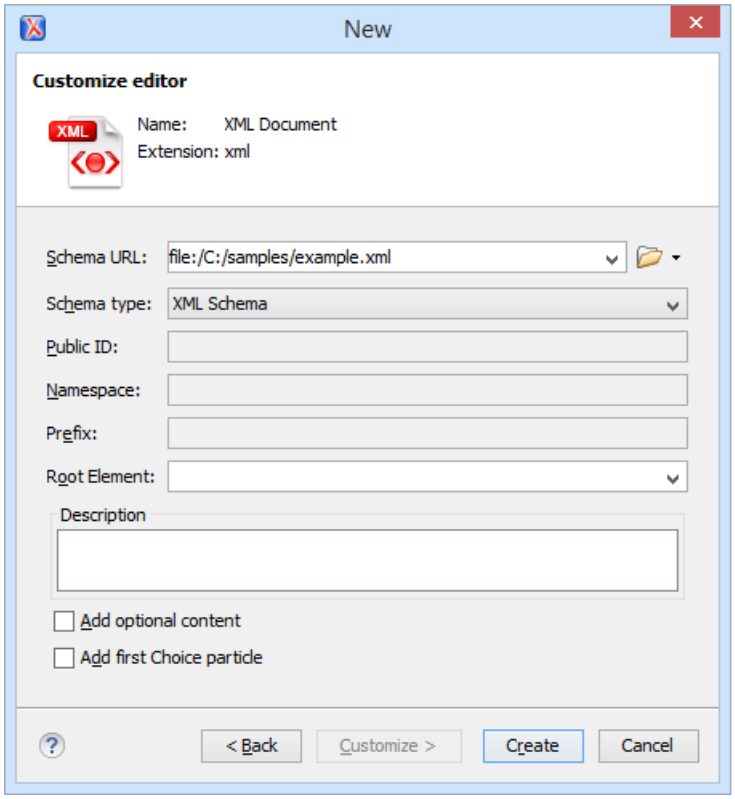


Figure 108: New XML Document Configuration Wizard Page

If you selected **XML Document** for the type of file you want to create and selected the **Customize** option, the configuration dialog box will include the following options:

- **Schema URL** - Specifies the path to the schema file. When you select a file, Oxygen XML Editor analyzes its content and tries to fill in the rest of the dialog box.
- **Schema Type** - Allows you to select the schema type. The following options are available: `XML Schema`, `DTD`, `RelaxNG XML syntax`, `RelaxNG compact syntax`, and `NVDL`.
- **Public ID** - Specifies the PUBLIC identifier declared in the document prolog.
- **Namespace** - Specifies the document namespace.
- **Prefix** - Specifies the prefix for the namespace of the document root.
- **Root Element** - Populated with elements defined in the specified schema, enables selection of the element used as document root.
- **Description** - A small description of the selected document root.
- **Add Optional Content** - If you select this option, the elements and attributes defined in the XML Schema as optional are generated in the skeleton XML document.
- **Add First Choice Particle** - If you select this option, Oxygen XML Editor generates the first element of an `xs:choice` schema element in the skeleton XML document. Oxygen XML Editor creates this document in a new editor panel when you click **OK**.

XSLT Stylesheet File Type

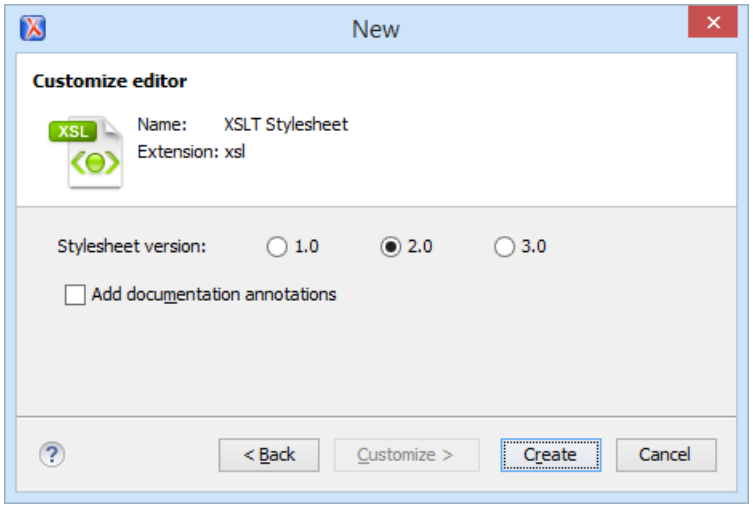


Figure 109: New XSLT Stylesheet Configuration Wizard Page

If you selected **XSLT Stylesheet** for the type of file you want to create and selected the **Customize** option, the configuration dialog box will include the following options:

- **Stylesheet version** - Allows you to select the Stylesheet version number. You can select from: `1.0`, `2.0`, and `3.0`.
- **Add documentation annotations** - Select this option to generate the stylesheet annotation documentation.

XML Schema File Type

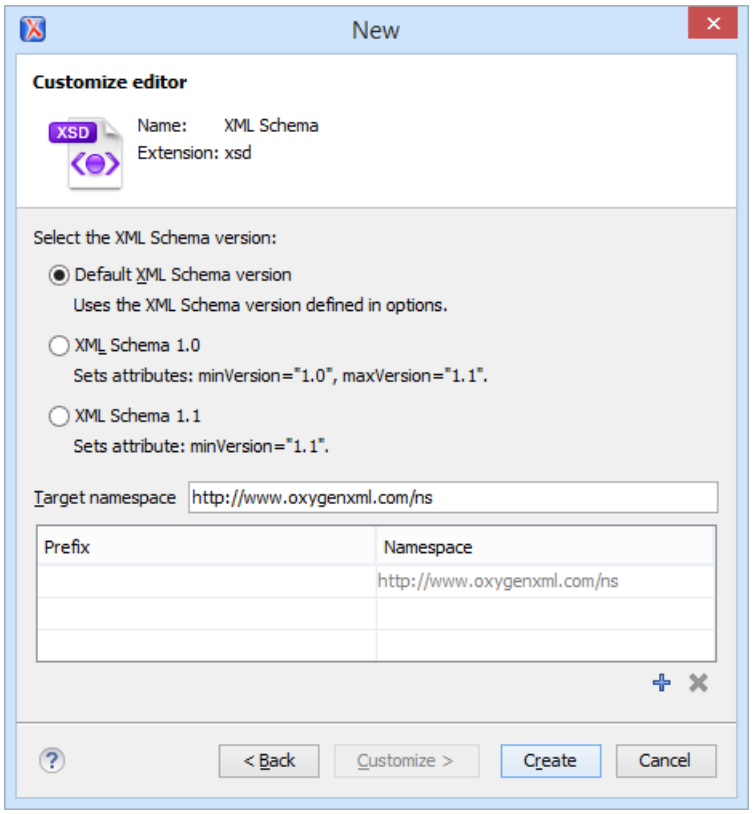


Figure 110: New XML Schema Configuration Wizard Page

If you selected **XML Schema** for the type of file you want to create and selected the **Customize** option, the configuration dialog box will include the following options:

- **Default XML Schema version** - Uses the XML Schema version defined in the ***XML Schema*** *preferences page*.
- **XML Schema 1.0** - Sets the `minVersion` attribute to *1.0* and the `maxVersion` attribute to *1.1*.
- **XML Schema 1.1** - Sets the `minVersion` attribute to *1.1*.
- **Target namespace** - Allows you to specify the schema target namespace.
- **Namespace prefix declaration table** - This table contains namespace prefix declarations. Table information can be managed using the **New** and **Delete** buttons.

  Tip: For further details on how you can set the version of an XML Schema, go to *Setting the XML Schema Version*.

Schematron File Type

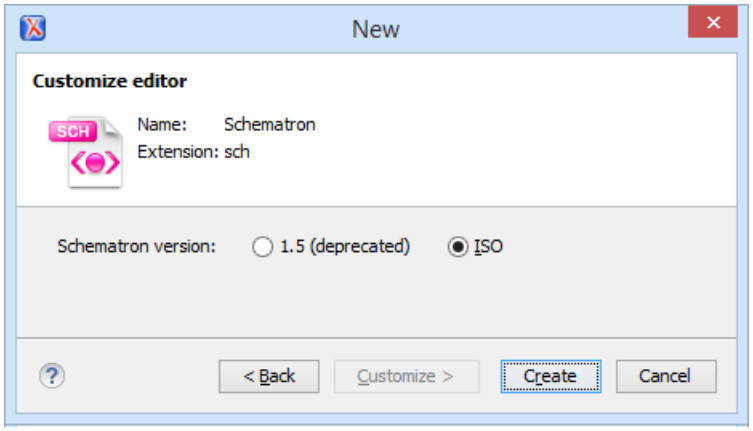


Figure 111: New Schematron Configuration Wizard Page

If you selected **Schematron** for the type of file you want to create and selected the **Customize** option, the configuration dialog box will include the following option:

- **Schematron version** - Specifies the Schematron version. Possible options: `1.5` (deprecated) and `ISO`.

  Note: Starting with version 16.0 of Oxygen XML Editor, the support for Schematron 1.5 is deprecated. It is recommended to use ISO Schematron instead.

Creating New Document Templates

Oxygen XML Editor allows you to create your own custom document templates and they will appear in the **Global templates** folder (or another specified folder) within the ***New*** *document wizard*.

Creating a New Document Template

To create your own custom document template and have it appear in the new file wizard, follow these steps:

1. Create a new file (whatever type of document you need) and customize it to become a starting point for creating new files of this type.

   Tip: You can use *editor variables* in the template file content and they will be expanded when the files are opened.
2. Save the new file template in one of the following locations:
   - The `templates` directory of the Oxygen XML Editor installation directory (*`[OXYGEN_INSTALL_DIR]`*`/templates`). File templates saved in this directory will appear in the **Global templates** category in the ***New*** *document wizard*.
   - You can also use any other directory of your choice, but you must add that directory to the list of templates in the ***Document Templates*** *preferences page*. This user-defined directory will appear in the ***New*** *document wizard* with the new file templates that you save in it.

   Attention: The name that you use to save the template will be the name that appears in the new file wizard, including capitalization, space, and characters (for example, `My Custom Template1.xml` will appear in the new file wizard as **My Custom Template1**).

3. Open the new file wizard (**New** toolbar button or **File** > **New**) and you should see your custom template in the appropriate folder. For DITA templates, they will also appear in the dialog box for creating new DITA topics from the **DITA Maps Manager**, but if you create a corresponding properties file (see the procedure below), you need to set the `type` property to `dita`.

Related Information:

Customizing Document Templates on page 257

Customizing Document Templates

Oxygen XML Editor allows you to customize certain aspects of predefined or custom document templates. For example, you can customize the icons or specify a prefix/suffix that will be used for the proposed file name in the ***New** document wizard*.

Customizing the Icons for a Document Template

If you want to customize the icons to be used for document templates, use a properties file to specify the icons using the following procedure:

1. Create a new properties file or edit an existing one.
   - If you create a new properties file, use the same name as the template file except with a `.properties` extension (for example, `MyTemplate.properties`). This properties file will specify the paths to the icons that will be used in the new file wizard. You can find some examples in the `templates` directory of the Oxygen XML Editor installation directory to help you get started.

     When defining the icons, the properties file should look like this:

     ```
     type=general
     smallIcon=../icons/Article_16.png
     bigIcon=../icons/Article_48.png
     ```

     Important: For DITA files, the `type` property needs to be set to `dita`. Otherwise, the template will not appear in the dialog box for creating new DITA topics from the ***DITA Maps Manager***. For all other types of files, set it to `general`. The icons specified in this properties file will only be used for the new file wizards and not in any other part of the interface.

     Note: If you created a new template and chose to use a custom directory for the new template (in *step 2 of the new template procedure*), make sure the path to the icons is relative to that directory.
   - If you edit an existing template, simply define the icon paths as specified *above*.
2. Save the properties file in the same directory as the document template.
3. Open the new file wizard (**File** > **New**) and you should see your custom icons next to the document template in the appropriate folder.

Add a Prefix or Suffix to File Names for a Document Template

You can use a properties file for each document template to add a prefix or suffix to the file name that is proposed in certain dialog boxes when you create a new file from that template. This applies to the following new document dialog boxes:

- The new document dialog box that appears when you select **New** > **File** from the contextual menu in the ***Project** view*. The prefix or suffix is added to the name of the file in the **File name** field.
- For DITA files, it also applies to the new document dialog box that appears when you select **Append Child** > **New**, **Insert Before** > **New**, or **Insert After** > **New** from the ***DITA Maps Manager***. The prefix or suffix is added to the name of the file in the **Save as** field.

To add a prefix or suffix to the file names for a document template, follow these steps:

1. Create a new properties file or edit an existing one.
   - If you create a new properties file, use the same name as the template file except with a `.properties` extension (for example, `MyTemplate.properties`). This properties file will specify the prefix/suffix that will be used to propose the file name in the new file wizards.

When defining the prefix/suffix, the properties file should look something like this:

```
type=general
filenamePrefix=prod_
filenameSuffix=_test
```

Important: For DITA files, the `type` property needs to be set to `dita`. For all other types of files, set it to `general`.

- If you edit an existing template, simply define the prefix/suffix as specified *above*.

2. Save the properties file in the same directory as the document template.
3. Open the new document wizard (*using the methods described above*) and when you select the appropriate template, you should see your prefix or suffix in the file name that is proposed in that dialog box.

Configure the Displayed Names for Document Templates

To change the name that is displayed for a document template, use the following procedure:

1. Create a new properties file or edit an existing one. If you create a new properties file, use the same name as the template file except with a `.properties` extension (for example, `MyTemplate.properties`).
2. Add a `displayName` property in the properties file:

```
displayName=myTemplateName
```

Tip: The names for *framework*-specific document templates (such as DITA *Topic* or DocBook *Article* as you would see in the **Framework templates** section in the **New** file wizard) can be translated via the internationalization support. In this case, the properties file should contain something like:

```
displayName=${i18n(tag)}
```

where *tag* refers to an entry in the `translation.xml` file for that specific framework (for example, `OXYGEN_INSTALL_DIR/frameworks/dita/i18n/translation.xml` for DITA).

3. Save the properties file in the same directory as the document template.
4. Open the new file wizard (**File** > **New**) and you should see the new name for the template.

Adding Placeholders or Hints in a Document Template

Document templates sometimes contain empty elements and it may not be clear to the Author what should be inserted. You can define placeholders in document templates that provide hints for Authors to help them understand what type of content should be added in any particular empty element within the document. The placeholder text is specified using a processing instruction and the placeholders are removed when the Author inserts content in the corresponding element.

To define placeholders in a document template to provide authors with hints, follow this procedure:

1. Edit the document template.
2. Add placeholders in the form of processing instructions within the elements where you want hints to be displayed when an Author creates a document from the template. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topic PUBLIC "-//OASIS//DTD DITA Topic//EN" "topic.dtd">
<topic id="pi">
  <title><?oxy-placeholder content="Please give a title"?></title>
    <shortdesc><?oxy-placeholder content="Writing short descriptions
         induces the writer to clarify the main thesis of the topic.
         We recommended a 50 word limit."?></shortdesc>
  <body>
    <p><?oxy-placeholder content="A paragraph element should be a self-contained
         unit dealing with one idea or point."?></p>
  </body>
</topic>
```

3. Save the template file.
4. Use the ***New*** *document wizard* to create a new document using your customized template and you should see the hints in the opened document.

Related Information:

Creating New Document Templates on page 256

Opening Documents

To open a document in Oxygen XML Editor, do one of the following:

- Go to **File** > **Open (Ctrl + O (Command + O on OS X))** or click the **Open** toolbar button to display the **Open File** dialog box. The start folder of the **Open** dialog box can be either the last folder visited by this dialog box or the folder of the currently edited file. This can be *configured in the user preferences.*
- Go to **File** > **Open URL** or click the **Open URL** toolbar button to display a dialog box that allows you to access any resource identified through a URL (defined by a protocol, host, resource path, and an optional port). The following actions are available in the drop-down action list:
  - **Browse for local file** - Opens a local file browser dialog box, allowing you to select a local file.
  - **Browse for remote file** - Displays the ***Open URL*** *dialog box* that allows you to open a remotely stored file.
  - **Browse for archived file** - Displays the ***Archive Browser*** that allows you to browse the content of an archive and choose a file.
  - **Browse Data Source Explorer** - Opens the ***Data Source Explorer*** that allows you to browse the data sources defined in the *Data Sources preferences page*.

    Tip: You can open the **Data Sources** preferences page by using the **Configure Database Sources** shortcut from the **Open URL** dialog box.
  - **Search for file** - Displays the ***Open/Find Resource*** *dialog box* that allows you to search for a file.
- Click the **Open/Find Resource** toolbar button to search for a file to open.
- Go to **File** > **Reload** to load the last saved file content. All unsaved modifications are lost.
- Go to **File** > **Reopen** to reopen one of the recently opened document files. The list containing recently opened files can be emptied by invoking the **Clear history** action.
- Select the **Open** or **Open with** action from the contextual menu of the ***Project*** *view*.

Related Information:

Opening Local Files at Start-up on page 259

Opening the Current Document in System Application

To open the currently edited document in the associated system application, use the **View in Browser/System Application** action that is available in the **File** menu and on the **File** toolbar. If you want to open XML files in a specific internet browser, instead of the associated system application, you can specify the internet browser to be used. To do so, *open the* ***Preferences*** *dialog box* ***(Options > Preferences)***, go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.

Opening Local Files at Start-up

To open a local file at start-up when you open Oxygen XML Editor from the command line, add the paths for one or more local files as parameters in the command line:

- `scriptName [pathToXMLFile1] [pathToXMLFile2]`
  - **scriptName** is the name of the startup script for your platform (oxygen.bat on Windows, `oxygen.sh` on Unix/Linux, `oxygenMac.sh` on Mac OS).
  - **pathToXMLFileN** is the name of a local XML file.
- An XML file and a schema file to be associated automatically to the file and used for validation and content completion:

  ```
  scriptName -instance pathToXMLFile -schema pathToSchemaFile -schemaType
  XML_SCHEMA|DTD_SCHEMA|RNG_SCHEMA|RNC_SCHEMA -dtName documentTypeName
  ```

  - **scriptName** is the name of the startup script for your platform (oxygen.bat on Windows, oxygen.sh on Unix/Linux, oxygenMac.sh on Mac OS).
  - **pathToXMLFile** is the name of a local XML file.

- **pathToSchemaFile** is the name of the schema that you want to associate to the XML file, the four constants (XML_SCHEMA, DTD_SCHEMA, RNG_SCHEMA, RNC_SCHEMA) are the possible schema types (W3C XML Schema, DTD, Relax NG schema in full syntax, Relax NG schema in compact syntax).
- **documentTypeName** specifies the name of the document type for which the schema is defined. If the document type is already set in preferences, its schema and type are updated.

The two possibilities of opening files at startup by specifying them in the command line are explained also if the startup script receives one of the *-h* or *--help* parameters.

Related Information:

Opening a File at a Specific Location Using the Command Line Interface on page 260

Opening a File at a Specific Location Using the Command Line Interface

Oxygen XML Editor offers support for opening a file at a specific position using the command line interface, by transmitting parameters to the Oxygen XML Editor batch script file. The following methods are available, depending on how you identify the position that is needed:

1. **Specific position values (line and column number, or character offset)**

   Oxygen XML Editor supports the following position parameters:

   - `line` - The line number.
   - `column` - The column number (has meaning if the `line` parameter is also defined).
   - `char` - The character offset.

   Examples for Windows:

   The following examples show how you can open an XML document in Oxygen XML Editor:

   ```
   oxygen.bat  file:samples/personal.xml#line=4
   oxygen.bat  file:samples/personal.xml#line=4column=5
   oxygen.bat  file:samples/personal.xml#line=4;column=5
   oxygen.bat  file:samples/personal.xml#char=334
   ```

2. **Simplified XPath index path**

   Oxygen XML Editor will open an XML file and select one of its elements identified by a simplified XPath index path. For example, an index path of the form 1/5/7 identifies the seventh child of the fifth child of the root element.

   Restriction: Oxygen XML Editor will display a selection that starts with the first character of the content of the identified element and spans until the end of the line.

   Examples for Windows:

   The following example shows how you can open an XML document in Oxygen XML Editor and select the third child of the root element:

   ```
   oxygen.bat  file:samples/personal.xml#element(1/3)
   ```

3. **Anchors identified by ID attribute values**

   Oxygen XML Editor will open an XML file and select the element whose `id` attribute value is an exact match of the *anchor* attached to a command line instruction.

   Examples for Windows:

   The following example shows how you can open an XML document in Oxygen XML Editor and select the element that has the `id` element set to `titleID`:

   ```
   oxygen.bat  file:samples/personal.xml#titleID
   ```

Related Information:

Opening Local Files at Start-up on page 259

Saving Documents

You can save the document you are editing with one of the following actions:

- **File** > **Save**.
- **Save** toolbar button - If the document was not yet saved, it displays the **Save As** dialog box.
- **File** > **Save As** - Displays the **Save As** dialog box, used either to name and save an open document to a file or to save an existing file with a new name.
- **File** > **Save To URL** - Displays a **Save to URL** dialog box that can be used to save a file identified by its URL (defined by a protocol, host, resource path, and an optional port). Use the drop-down action list to choose one of the available save actions:
  - **Browse for local file** - Opens a local file browser dialog box allowing you to save the document locally.
  - **Browse for remote file** - Displays a **Save to URL** dialog box that allows you to save the document to a remote location (accessible through FTP, SFTP or WebDAV).
  - **Browse for archived file** - Displays the ***Archive Browser*** that allows you to save the document inside an archive.
  - **Browse Data Source Explorer** - Opens a ***Data Source Explorer*** that allows you to browse the data sources defined in the ***Data Sources*** *preferences page*.

    Tip: You can get to the **Data Sources** preferences page, using the **Configure Database Sources** shortcut from the **Save to URL** dialog box.
  - **Search for file** - Displays the ***Open/Find Resource*** *dialog box*.
- **File** > **Save All** - Saves all open documents. If any document does not have a file, displays the **Save As** dialog box.

Opening and Saving Remote Documents

Oxygen XML Editor supports editing remote files, using the FTP, SFTP, WebDAV , SharePoint, and SharePoint Online for Office 365 protocols. You can edit remote files in the same way you edit local files. For example, you can add remote files to a project, or use them in XSL and FO transformations.

You can open one or more remote files in *the* ***Open URL*** *dialog box*.

A WebDAV resource can be locked when it is opened in Oxygen XML Editor by selecting the ***Lock WebDAV files on open*** *option* to prevent other users to modify it concurrently on the server. If a user tries to edit a locked file, Oxygen XML Editor displays an error message that contains the lock owner's name. The lock is released automatically when the editor for that resource is closed in Oxygen XML Editor.

To avoid conflicts with other users when you edit a resource stored on a SharePoint server, you can **Check Out** the resource.

To improve the transfer speed, the content exchanged between Oxygen XML Editor and the HTTP / WebDAV server is compressed using the GZIP algorithm.

The current *WebDAV Connection* details can be saved by switching to the **Database** *perspective* and then you can browse and manage the connection in the ***Data Source Explorer*** *view*.

Open URL

To open this dialog box, go to **File** > **Open URL** (or click the **Open URL** toolbar button), then choose the **Browse for remote file** option from the drop-down action list.

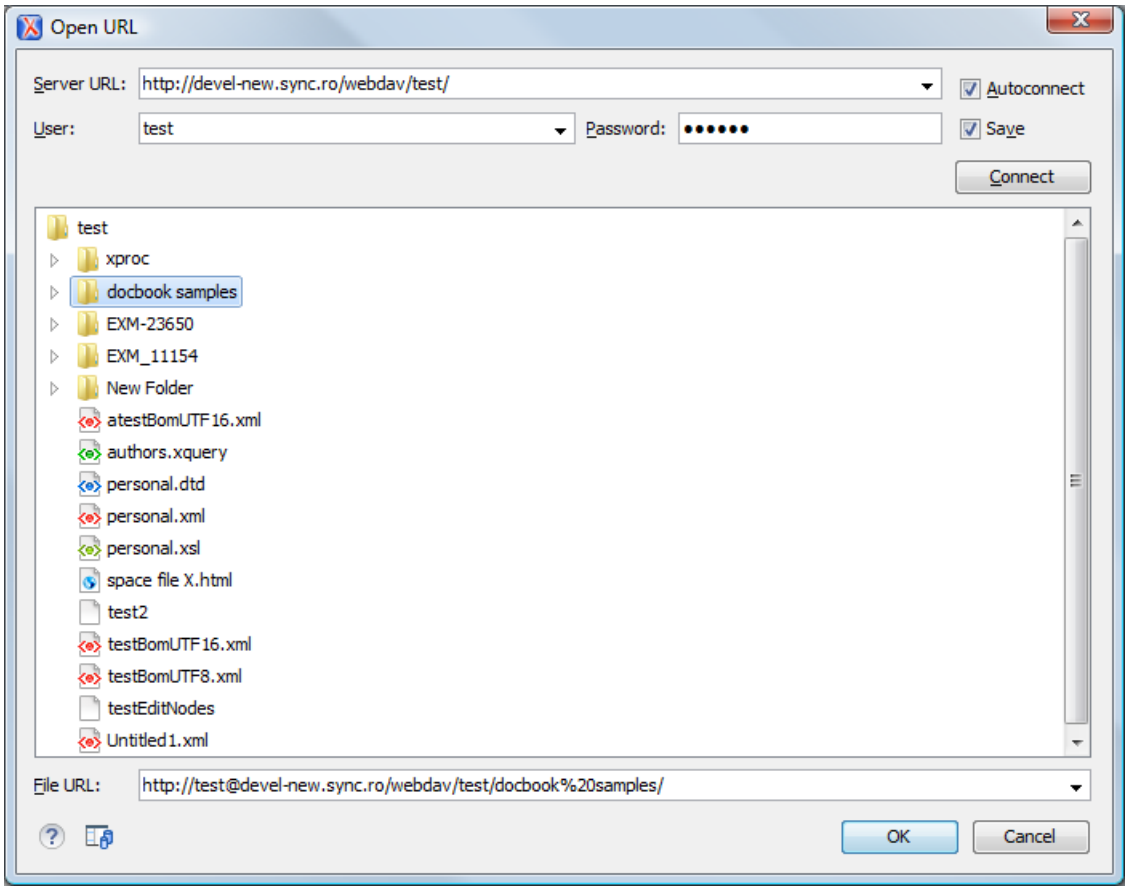


Figure 112: Open URL Dialog Box

The displayed dialog box is composed of the following:

Server URL

Specifies the protocol (HTTP, HTTPS or FTP) and the host name or IP of the server.

Tip: When specifying a URL, follow these rules:

- To access an FTP server, write the protocol, host, and port (if using a non-standard one). For example, `ftp://server.com` or `ftp://server.com:7800/`.
- To access a WebDAV server, write the path to the directory of the WebDAV repository along with the protocol and the host name. For example, `https://www.some-webdav-server.com:443/webdav-repository/`.

Important: Make sure that the repository directory ends in a slash "/". For example, `https://www.some-webdav-server.com:443/webdav-repository/`

Autoconnect

If selected, the browse action is performed every time when you open the dialog box.

User and Password

To browse for a file on a server, you have to specify the user and password for the server. This information is bound to the selected URL displayed in the **File URL** combo box, and used further in opening/saving the file. If the **Save** option is selected, then the user and password are saved between editing sessions. The password is kept encrypted in the options file.

Note: Your password is well protected. If the options file is used on another machine by a user with a different username, the password will become unreadable since the encryption is dependent on the username. This is also true if you add URLs that contain a username and password to your project.

Connect

When you press this button, the directory listing will be shown in the main section of the dialog box. If the selected URL points to a SharePoint server, a dedicated SharePoint browsing component is presented.

Browser view

- If you are browsing a WebDAV or FTP repository, the items are presented in a tree-like fashion. You can browse the directories, and make multiple selections. Additionally, you may use the **Rename**, **Delete**, and **New Folder** actions to manage the file repository.

  Note: The file names are sorted in a case-insensitive way.
- When you browse a SharePoint repository, a specialized component renders the SharePoint site content.

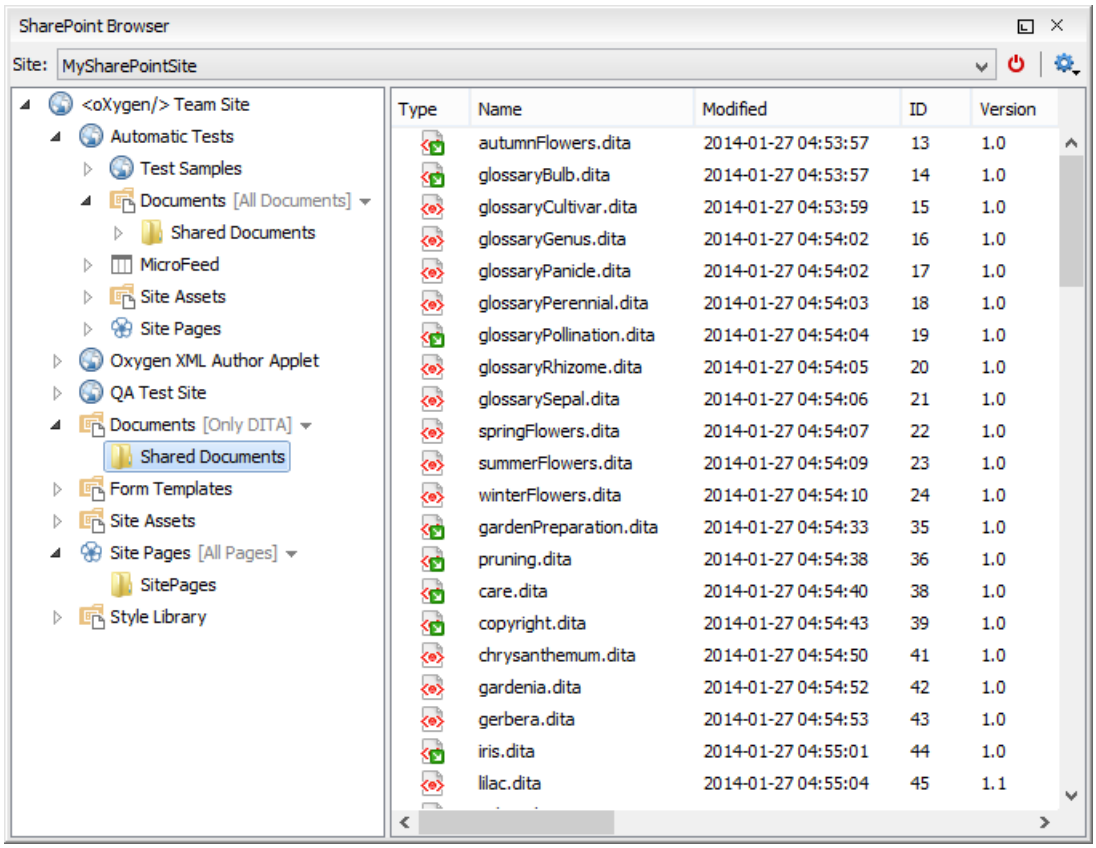


Figure 113: Browsing a SharePoint Repository

The left side navigation area presents the SharePoint site structure in a tree-like fashion with various node types (such as *sites*, *libraries*, and *folders*).

Depending on the type of node, a contextual menu offers customized actions that can be performed on that node. The contextual menu of a folder allows you to create new folders and documents, import folders and files, and to rename and delete the folder.

Note: The rename and delete actions are not available for library root folders (folders located at first level in a SharePoint library).

Each library node displays a drop-down menu next to its name where you can select what you want to display for the current library node. This functionality is also available on the contextual menu of the node.

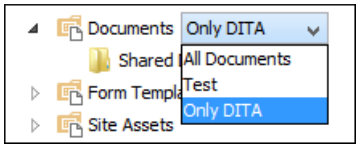


Figure 114: Drop-Down Menu to Select Which Items to Display

The content of a folder is displayed in a tabular form, where each row represents the properties of a folder or document. The list of columns and the way the documents and folders are organized depends on the currently selected view of the parent library.

You can filter and sort the displayed items. To display the available filters of a column, click the filter widget located on the column header. You can apply multiple filters at the same time.

Note: A column can be filtered or sorted only if it was configured this way on the server side.

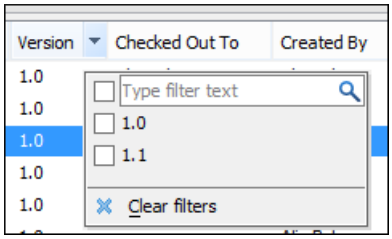


Figure 115: Column Filter

File URL

You can use this combo box to directly specify the URL to be opened or saved. You can type a URL such as `http://some.site/test.xml` (if the file is accessible through normal HTTP protocol), or `ftp://anonymous@some.site/home/test.xml` (if the file is accessible through anonymous FTP).

This combo box also displays the current selection when the user changes selection by browsing the tree of folders and files on the server.

Changing File Permissions on a Remote FTP Server

Some FTP servers allow the modification of permissions of the files served over the FTP protocol. This protocol feature is accessible directly in the FTP/WebDAV file browser dialog box by right-clicking a tree node and selecting the *Change permissions* menu item.

In this dialog box, the usual Unix file permissions *Read*, *Write*, and *Execute* are granted or denied for the file owner, owner group, and the rest of the users. The aggregate number of permissions is updated in the *Permissions* text field when it is modified with one of the checkboxes.

WebDAV over HTTPS

If you want to access a WebDAV repository across a non-secure network, Oxygen XML Editor allows you to load and save the documents over the HTTPS protocol (if the server understands this protocol) so that any data exchange with the WebDAV server is encrypted.

When a WebDAV repository is first accessed over HTTPS, the server hosting the repository will present a security certificate as part of the HTTPS protocol, without any user intervention. Oxygen XML Editor will use this certificate to decrypt any data stream received from the server. For the authentication to succeed you should make sure the security certificate of the server hosting the repository can be read by Oxygen XML Editor. This means that Oxygen XML Editor can find the certificate in the key store of the Java Runtime Environment in which it runs. You know the server certificate is not in the JRE key store if you get the error *No trusted certificate found* when trying to access the WebDAV repository.

Troubleshooting HTTPS

When Oxygen XML Editor cannot connect to an HTTPS-capable server, most likely there is no certificate set in the *Java Runtime Environment (JRE)* that Oxygen XML Editor runs into. The following procedure describes how to:

- Export a certificate to a local file using any HTTPS-capable Web browser (for example, Internet Explorer).
- Import the certificate file into the JRE using the keytool tool that comes bundled with Oxygen XML Editor.

Tip: To make Oxygen XML Editor accept a certificate even if it is invalid, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Connection settings** > **HTTP(S)/WebDAV**, and select the **Automatically accept a security certificate, even if invalid** option.

1. Export the certificate into a local file

a) Point your HTTPS-aware Web browser to the repository URL.

If this is your first visit to the repository it will be displayed a security alert stating that the security certificate presented by the server is not trusted.

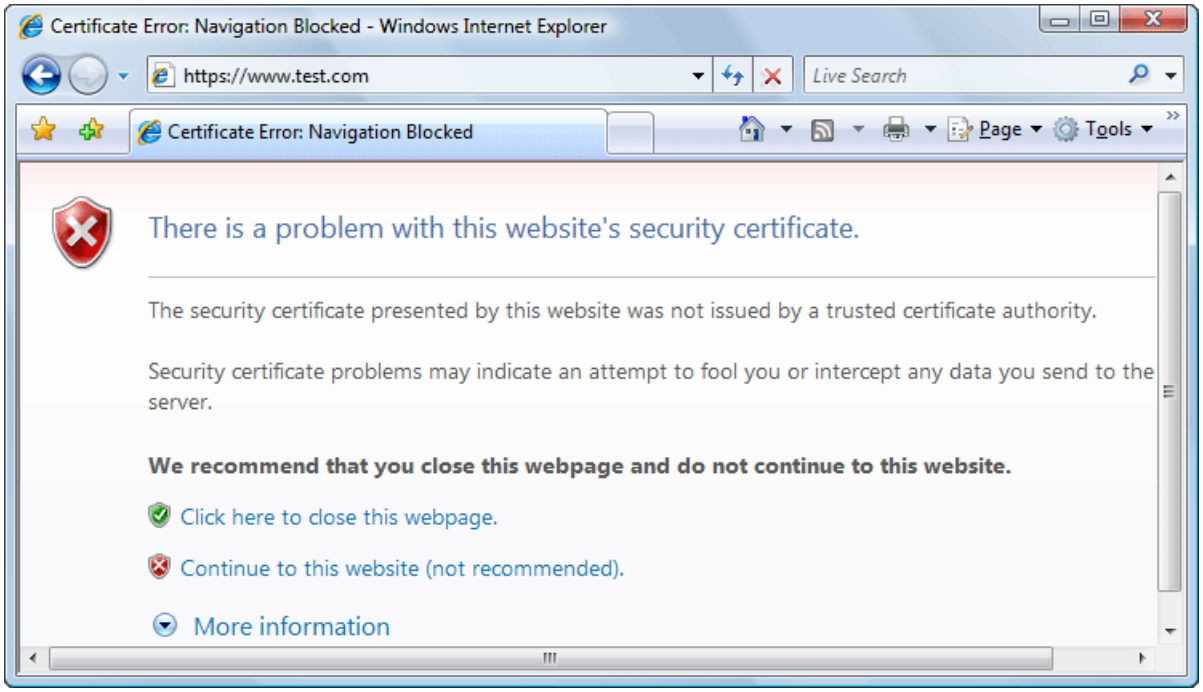


Figure 116: Security alert - untrusted certificate

b) Go to menu **Tools** > **Internet Options**.
Internet Options dialog box is opened.

c) Select **Security** tab.

d) Select **Trusted sites** icon.

e) Press **Sites** button.
This will open **Trusted sites** dialog box.

f) Add repository URL to **Websites** list.

g) Close the **Trusted sites** and **Internet Options** dialog boxes.

h) Try again to connect to the same repository URL in Internet Explorer.
The same error page as above will be displayed.

i) Select **Continue to this website** option.
A clickable area with a red icon and text **Certificate Error** is added to Internet Explorer address bar.

j) Click the **Certificate Error** area.
A dialog box containing a **View certificates** link is displayed.

k) Click the **View certificates** link.
Certificate dialog box is displayed.

l) Select **Details** tab of **Certificate** dialog box.

m) Press **Copy to File** button.
Certificate Export Wizard is started.

n) Follow indications of wizard for DER encoded binary X.509 certificate. Save certificate to local file `server.cer`.

2. Import the local file into the JRE running Oxygen XML Editor.

a) Open a text-mode console with administrative rights.

If Oxygen XML Editor has been installed in a user's home directory and includes a bundled JRE, administrative rights are not required. In all other cases administrative rights will be required.

b) Go to the `lib/security` directory of the JRE running Oxygen XML Editor. You find the home directory of the JRE in the *java.home* property that is displayed in the **About** dialog box (**System properties** tab). On Mac OS X systems, the `lib/security` directory is usually located in `/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home` directory.

On OS X, if you have installed a distribution of Oxygen XML Editor that is not bundled with a JRE, a JRE from Apple is required. The Apple Java version 1.6 stores the certificates in `/System/Library/Java/Support/CoreDeploy.bundle/Contents/Home/lib/security/cacerts` with a symbolic link pointing to it from `/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/lib/security/cacerts`.

On OS X, if you have installed a distribution of Oxygen XML Editor that bundles the JRE from Oracle, the JRE uses the `.install4j/jre.bundle/Contents/Home/jre/lib/security/cacerts` path within its installation directory.

c) Run the following command:

```
..\..\bin\keytool -import -trustcacerts -file server.cer -keystore cacerts
```

The `server.cer` file contains the server certificate, created during the previous step. keytool requires a password before adding the certificate to the JRE *keystore*. The default password is `changeit`. If someone changed the default password, then that person is the only one who can perform the import.

Tip: If you need to import multiple certificates, you need to specify a different alias for each additional imported certificate with the `-alias` command line argument, as in the following example:

```
..\..\bin\keytool -import -alias myalias1 -trustcacerts -file
server1.cer -keystore cacerts

..\..\bin\keytool -import -alias myalias2 -trustcacerts -file
server2.cer -keystore cacerts
```

3. Restart Oxygen XML Editor.

Related Information:

HTTP(S)/WebDAV Preferences on page 157

HTTP Authentication Schemes

Oxygen XML Editor supports the following HTTP authentication schemes:

- **Basic** - The *basic* authentication scheme defined in the *RFC2617 specifications*.
- **Digest** - The *digest* authentication scheme defined in the *RFC2617 specifications*.
- **NTLM** - The *NTLM* scheme is a proprietary Microsoft Windows Authentication protocol (considered to be the most secure among currently supported authentication schemes).

  Note: For NTLM authentication, the user name must be preceded by the name of the domain it belongs to, as in the following example:

  ```
  domain\username
  ```

- ***Kerberos*** - An authentication protocol that works on the basis of *tickets* to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner.

Single Sign-on

Oxygen XML Editor implements the *Single sign-on* property (meaning that you can log on once and gain access to multiple services without being prompted to log on for each of them), based on the **Kerberos** protocol and relies on a *ticket-granting ticket (TGT)* that Oxygen XML Editor obtains from the operating system.

To turn on the **Kerberos**-based authentication, you need to add the following system property in the `.vmoptions` configuration file or start-up script:

```
-Djavax.security.auth.useSubjectCredsOnly=false
```

Related Information:

Setting a Java Virtual Machine Parameter when Launching Oxygen XML Editor on page 178

Switching and Moving File Tabs

There are two keyboard shortcuts that can be used for cycling through the opened file tabs:

Ctrl + Tab (Command + Tab on OS X)

Switches between the tabs with opened files in the order most recent ones first.

Ctrl + Shift + Tab (Command + Shift + Tab on OS X)

Switches between the tabs with opened files in the reverse order.

There are also two shortcuts for moving the current tab:

Ctrl + Alt + Comma

Moves the current file tab one position to the left.

Ctrl + Alt + Period

Moves the current file tab one position to the right.

Closing Documents

To close open documents, use one of the following actions that are available in the contextual menu of the current editor tab (or from the **File** menu):

Close (Ctrl + W (Command + W on OS X))

Closes the currently selected editor.

Close Other Files

If multiple files are opened, this action is available to close all opened editors in the current group/stack of tabs except for the one you are currently viewing. If this action is selected from the **File** menu, it closes all opened editors in **all** groups/stacks of tabs except for the current one.

Close Files to the Right

Available only from the contextual menu of the current editor tab and it closes all opened editors to the right of the currently selected editor.

Close All

If multiple files are opened, this action is available to close all opened editors in the current group/stack of tabs. If this action is selected from the **File** menu, it closes all opened editors in **all** groups/stacks of tabs.

Contextual Menu of the Current Editor Tab

A contextual menu is available when you right-click the current editor tab label.

The actions that are available depend on the context and the number of files that are opened. The menu includes the following actions:

Close (Ctrl + W (Command + W on OS X))

Closes the currently selected editor.

Close Other Files

If multiple files are opened, this action is available to close all opened editors in the current group/stack of tabs except for the one you are currently viewing.

Close Files to the Right

Closes all opened editors to the right of the currently selected editor.

Close All

If multiple files are opened, this action is available to close all opened editors.

Move editor tab to the left (Ctrl + Alt + Comma)

Moves the current editor tab one position to the left.

Move editor tab to the right (Ctrl + Alt + Period

Moves the current editor tab one position to the right.

Reopen last closed editor Ctrl + Alt + T (Command + Alt + T on OS X))

Reopens the last closed editor.

Maximize/Restore Editor Area

Collapses all the side views and spans the editing are to cover the entire width of the main window.

Add to project

Adds the file you are editing to the current project.

Add all to project

If multiple files are opened, this action is available to add all the opened files in the current group/stack of tabs to the current project.

Copy Location

Copies the disk location of the file.

Show in Explorer (Show in Finder on OS X)

Opens the Explorer to the file path of the file.

Viewing File Properties

The **Properties** view displays information about the currently edited document. The information includes:

- Character encoding.
- Full path on the file system.
- Schema used for content completion and document validation.
- Document type name and path.
- Associated transformation scenario.
- Read-only state of a file.
- Bidirectional text (left to right and right to left) state.
- Total number of characters in the document.
- Line width.
- Indent with tabs state.
- Indent size.

The view can be accessed from **Window** > **Show View** > **Properties**.

To copy a value from the **Properties** view in the clipboard (for example, the full file path), use the **Copy** action available on the contextual menu of the view.

Searching Documents

Oxygen XML Editor includes advanced search capabilities to help you locate documents and resources.

Open/Find Resource View

The **Open/Find Resource** view is designed to offer advanced search capabilities either by using a simple text search or by using the *Apache Lucene - Query Parser Syntax*. By default, the view is presented in the left side of the Oxygen XML Editor layout, next to the ***Project** view* or ***DITA Maps Manager***. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

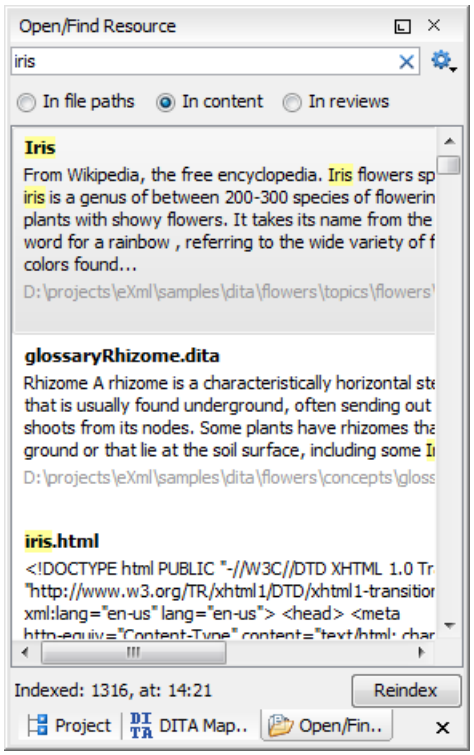


Figure 117: Open/Find Resource View

You can use this view to find a file in the current Oxygen XML Editor project or in one of the *DITA maps* opened in *the **DITA Maps Manager** view* by typing only a few letters of the file name of a document or a fragment of the content you are searching for. The **Open/Find Resource** view also supports searching in document edits (comments, tracked change insertions/deletions, and highlighted content) by selecting the ***In reviews** option*.

Note: Full support for searching in document edits (the **In reviews** option) is available only in the Enterprise edition of Oxygen XML Editor. The Professional edition offers limited support to search through a maximum of 10 edits.

Search Results

Search results are presented instantly, after you finish typing the content. The matching fragments of text are highlighted in the results list displayed in the dialog box. When you open one of the documents from the results list, the matching fragments of text are highlighted in the editing area. To remove the highlighting from your document, close the corresponding tab in the ***Results** view* at the bottom of the editor. To display the search history, position the cursor in the search field and press **Ctrl + DownArrow (Command + DownArrow on OS X)** or **Ctrl + UpArrow (Command + UpArrow on OS X)** on your keyboard. Pressing only the **DownArrow** key moves the selection to the list of results.

Note: Searches are not case sensitive. For example, if you search for `car` you get the same results as when you search for `Car`.

Tip: Suffix searches are also supported, both for searching in the content of your resources and in their name. For this, you can use wildcards. If you search for `*ing` with the **in content** option selected, you will find documents that contain the word *presenting*. If you search for `*/samples/*.gif` with the **in file paths** option selected, you will find all the *gif* images from the `samples` directory.

Options Available in the View

The **Open/Find Resource** view offers the following options:

- **Settings** - Drop-down menu that includes the following settings for the view:
  - **Clear Index** - Clears the index.

- **Show description** - Presents the search results in a more compact form, displaying only the title and the location of the resources.
- **Options** - Opens the ***Open/Find Resource*** *preferences page* where you can configure various search options. For example, you can specify a **Content language** that differs from the default UI language in case your document contains multiple languages.

- ***In file paths*** - Select this option to search for resources by their name or by its path (or a fragment of its path).
- ***In content*** - Select this option to search through the content of your resources.
- ***In reviews*** - Select this option to search through the comments, *tracked change* insertions/deletions, or highlights in your resources.
- **Reindex** - Use this option to reindex your resources.

Contextual Menu Actions

A contextual menu is available on each search result and provides actions applicable to that particular document. These actions include:

- **Open** - Opens the document in one of Oxygen XML Editor internal editors.
- **Open with** - Allows you to choose to open the document in the **Internal editor** or an external **System application**.
- **Show in Explorer** - Identifies the document in the system file explorer.
- **Copy Location** - Copies the file path and places it in the clipboard.

Indexing Process

The content of the resources used to search in is parsed from an index. The indexing is performed both automatically and on request. Automatic indexing is performed when you modify, add, or remove resources in the currently indexed project. If the index was never initialized, the index in not updated on project changes.

To improve performance, the indexing process skips the following set of common English words (the so-called **stop words**): *a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with*. This means that if you are searching for any of these words, the indexing process will not be able to match any of them. However, you can configure the list of **stop words** in the ***Open/Find Resource*** *preferences page*.

Caching Mechanism

When you perform a search, a caching mechanism is used to gather the paths of all files linked in the current project. When the first search is performed, all project files are indexed and added to the cache. The next search operation uses the information extracted from the cache, thus improving the processing time. The cache is kept for the currently loaded project only, so when you perform a search in a new project, the cache is rewritten. Also, the cache is reset when you press the **Reindex** button.

Important: Files larger than 2GB are not indexed.

If there is no file found that matches your file pattern or text search, a possible cause is that the file you are searching for was added to the Oxygen XML Editor project after the last caching operation. In this case, re-indexing the project files with the **Reindex** button enables the file to be found. The date and time of the last index operation are displayed below the file list.

Opening the Results

Once you find the files that you want to open, select them in the list and press the **Open** button (or double-click them). Each of the selected files is opened in *the editor associated with the type of the file*.

Note: You can drag a resource from the **Open/Find Resource** view and drop it in a DocBook, DITA, TEI or XHTML document to create a link to that resource.

To watch our video demonstration about the **Open/Find Resource** feature and its search capabilities, go to *https://www.oxygenxml.com/demo/Open_Find_Resource.html*.

Related Information:

Open/Find Resource Dialog Box on page 271

Open/Find Resource Dialog Box

The **Open/Find Resource** dialog box offers advanced search capabilities. To open the dialog box, go to **Find** > **Open/Find Resource (Ctrl + Shift + R (Command + Shift + R on OS X))**. You can also click the **Open/Find Resource** toolbar button or use the **Search for file** action that is available in some URL input fields.

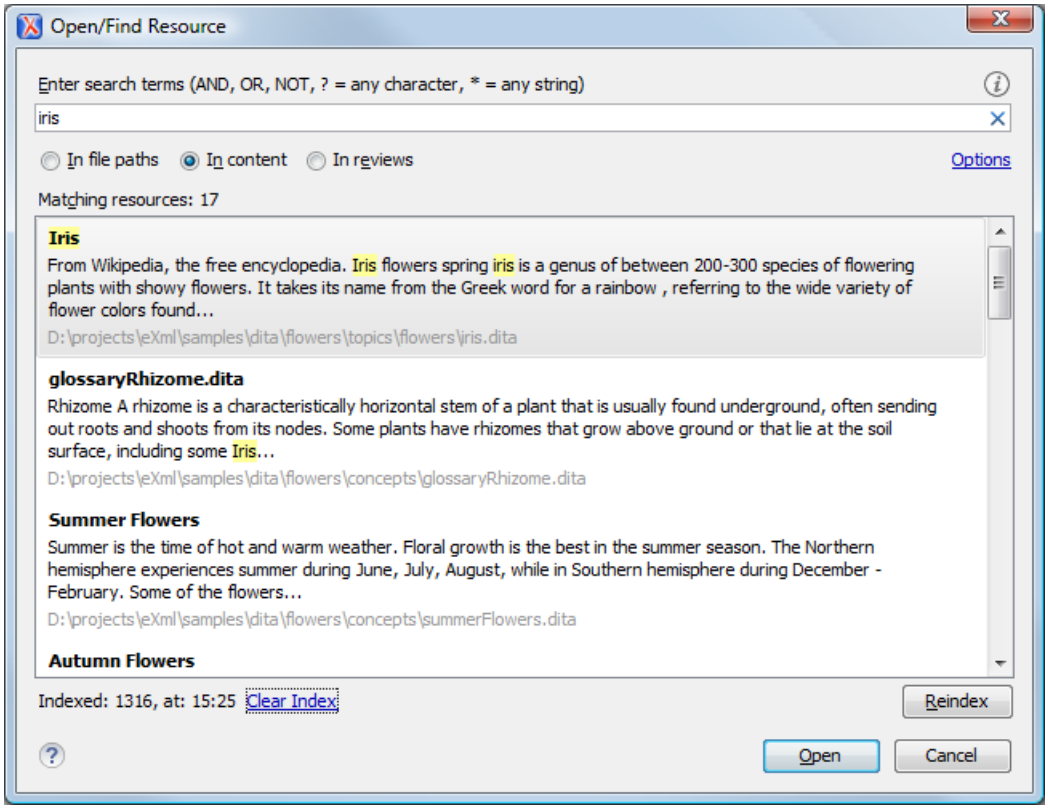


Figure 118: Open/Find Resource Dialog Box

You can use this dialog box to find a file in the current Oxygen XML Editor project or in one of the *DITA maps* opened in *the **DITA Maps Manager** view* by typing a few letters of the file name or a fragment of the content you are searching for. The **Open/Find Resource** dialog box also supports searching in document edits (comments, tracked change insertions/deletions, and highlighted content).

Note: Full support for searching in document edits (the **In reviews** option) is available only in the Enterprise edition of Oxygen XML Editor. The Professional edition offers limited support to search through a maximum of 10 edits.

Search Results

Search results are presented instantly, after you finish typing the content. The matching fragments of text are highlighted in the results list displayed in the dialog box. When you open one of the documents from the results list, the matching fragments of text are highlighted in the editing area. To remove the highlighting from your document, close the corresponding tab in the ***Results** view* at the bottom of the editor. To display the search history, position the cursor in the search field and press **Ctrl + DownArrow (Command + DownArrow on OS X)** or **Ctrl + UpArrow (Command + UpArrow on OS X)** on your keyboard. Pressing only the **DownArrow** key moves the selection to the list of results.

Note: Searches are not case sensitive. For example, if you search for `car` you get the same results as when you search for `Car`.

Tip: Suffix searches are also supported, both for searching in the content of your resources and in their name. For this, you can use wildcards. If you search for `*ing` with the **in content** option selected, you will find documents that contain the word *presenting*. If you search for `*/samples/*.gif` with the **in file paths** option selected, you will find all the *gif* images from the `samples` directory.

Options Available in the Dialog Box

The **Open/Find Resource** dialog box includes the following options:

- ***In file paths*** - Select this option to search for resources by their name or by its path (or a fragment of its path).
- ***In content*** - Select this option to search through the content of your resources.
- ***In reviews*** - Select this option to search through the comments, *tracked change* insertions/deletions, or highlights in your resources.
- **Options** - Opens the ***Open/Find Resource*** *preferences page* where you can configure various search options. For example, you can specify a **Content language** that differs from the default UI language in case your document contains multiple languages.
- **Clear Index** - Clears the index.
- **Reindex** - Use this option to reindex your resources.

Contextual Menu Actions

A contextual menu is available on each search result and provides actions applicable to that particular document. These actions include:

- **Open** - Opens the document in one of Oxygen XML Editor internal editors.
- **Open with** - Allows you to choose to open the document in the **Internal editor** or an external **System application**.
- **Show in Explorer** - Identifies the document in the system file explorer.
- **Copy Location** - Copies the file path and places it in the clipboard.

Indexing Process

The content of the resources used to search in is parsed from an index. The indexing is performed both automatically and on request. Automatic indexing is performed when you modify, add, or remove resources in the currently indexed project. If the index was never initialized, the index in not updated on project changes.

To improve performance, the indexing process skips the following set of common English words (the so-called **stop words**): *a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with*. This means that if you are searching for any of these words, the indexing process will not be able to match any of them. However, you can configure the list of **stop words** in the ***Open/Find Resource*** *preferences page*.

Caching Mechanism

When you perform a search, a caching mechanism is used to gather the paths of all files linked in the current project. When the first search is performed, all project files are indexed and added to the cache. The next search operation uses the information extracted from the cache, thus improving the processing time. The cache is kept for the currently loaded project only, so when you perform a search in a new project, the cache is rewritten. Also, the cache is reset when you press the **Reindex** button.

Important: Files larger than 2GB are not indexed.

If there is no file found that matches your file pattern or text search, a possible cause is that the file you are searching for was added to the Oxygen XML Editor project after the last caching operation. In this case, re-indexing the project files with the **Reindex** button enables the file to be found. The date and time of the last index operation are displayed below the file list.

Opening the Results

Once you find the files that you want to open, select them in the list and press the **Open** button (or double-click them). Each of the selected files is opened in *the editor associated with the type of the file*.

To watch our video demonstration about the **Open/Find Resource** feature and its search capabilities, go to *https://www.oxygenxml.com/demo/Open_Find_Resource.html*.

Related Information:

Open/Find Resource View on page 194

Open/Find Resource Preferences Page on page 154

Searching in Content

To perform a search through the content of your resources, open the ***Open/Find Resource*** *dialog box* (from the **Find** menu or with **Ctrl + Shift + R (Command + Shift + R on OS X)**) or the ***Open/Find Resource*** *view* (by default, located on the left side of the editor), select the **in content** option, and in the search field enter the terms that you want to search for.

The **Open/Find Resource** feature is powered by *Apache Lucene*. Apache Lucene is a free open source information retrieval software library.

You can use the **Open/Find Resource** feature to either perform a simple text search or a more complex search using the *Apache Lucene - Query Parser Syntax*. Using the *Apache Lucene - Query Parser Syntax* means you can perform any of the following searches:

Examples:

- **Term Searches-**

  Searching for plain text:

  ```
  Garden Preparation
  ```

- **Element-Specific Searches-**

  Searching for content that belongs to a specific element:

  ```
  title:"Garden Preparation"
  ```

- **Wildcard Searches-**

  Using wildcards to make your search more permissive:

  ```
  Garden Prepar?tion
  ```

- **Fuzzy Searches-**

  If you are not sure of the exact form of a term that you are interested in, use the fuzzy search to find the terms that are similar to the search term. To perform a fuzzy search, use the ~ symbol after the word that you are not sure of:

  ```
  Garden Preparing~
  ```

- **Proximity Searches-**

  Use proximity searches to find words that are within a specific distance away. To perform a proximity search, use the ~ symbol at the end of your search. For example, to search for the word **Garden** and the word **Preparation** within 6 words of each other use:

  ```
  "Garden Preparation"~6
  ```

- **Range Searches-**

  Use range searches to match documents whose element values are between the lower and upper bound specified in the range query. For example, to find all documents whose titles are between **Iris** and **Lilac**, use:

  ```
  title:{Iris TO Lilac}
  ```

  The curly brackets denote an exclusive query. The results you get when using this query are all the documents whose titles are between **Iris** and **Lilac**, but not including **Iris** and **Lilac**. To create an inclusive query use square brackets:

  ```
  title:[Iris to Lilac]
  ```

- **Term Prioritising Searches-**

  Use term prioritising searches if the fragment of text that you are searching for contains certain words that are more important to your search than the rest of them. For example, if you are searching for **Autumn Flowers**, a

good idea is to prioritize the word **Autumn** since the word **Flowers** occurs more often. To prioritize a word use the ^ symbol:

```
Autumn^6 Flowers
```

- **Searches Using Boolean Operators-**

  You can use the **AND**, **+**, **OR**, **-**, and **NOT** operators.

  To search for documents that contain both the words **Garden** and **Preparation**, use:

  ```
  Garden AND Preparation
  ```

  To search for documents that must contain the word **Garden** and may contain the word **Preparation**, use:

  ```
  +Garden Preparation
  ```

  To search for documents that contain either the word **Garden** or the word **Preparation**, use:

  ```
  Garden OR Preparation
  ```

  To search for documents that contain **Garden Preparation** but not **Preparation of the Flowers**, use:

  ```
  "Garden Preparation" - "Preparation of the Flowers"
  ```

- **Searches Using Grouping-**

  To search either for the word **Garden** or **Preparation**, and the word **Flowers**, use:

  ```
  (Garden OR Preparation) AND Flowers
  ```

- **Searches Using Element Grouping-**

  To search for a title that contains both the word **Flowers** and the phrase **Garden Preparation**, use:

  ```
  title:(+Flowers +"Garden Preparation")
  ```

- **Searching for Special Characters-**

  Sometimes you might need to search your content for special character, such as:

  ```
  + - && || ! ( ) { } [ ] ^ ~ * ? : \
  ```

  In this case, you should surround your search query with quotes. For example, to search for `(Hydrogen + Oxygen)=Water`, use:

  ```
  "(Hydrogen + Oxygen)=Water"
  ```

Searching in File Paths

To perform a search in the file paths of your resources, open the ***Open/Find Resource*** *dialog box* (from the **Find** menu or with **Ctrl + Shift + R (Command + Shift + R on OS X)**) or the ***Open/Find Resource*** *view* (by default, located on the left side of the editor), select the **In file paths** option, and in the search field enter the terms that you want to search for.

The **Open/Find Resource** feature allows you to search for a resource either by its name or by its path (or by a fragment of its path).

You can use wildcards when you perform such searches:

- Use "*" to match any sequence of characters.
- Use "?" to match any single character.

For example, if you search for ***-preferences-page** you will find all the resources that contain the *-preferences-page* fragment in their name. If you search for ***/samples/*.gif**, you will find all the `.gif` images from the `samples` directory.

Searching in Reviews

To perform a search in the edits of your resources, open the ***Open/Find Resource*** *dialog box* (from the **Find** menu or with **Ctrl + Shift + R (Command + Shift + R on OS X)**) or the ***Open/Find Resource*** *view* (by default, located on the left side of the editor), select the **In reviews** option, and in the search field enter the terms that you want to search for.

The following options are available:

- **Type** - Specifies whether you want to search for content in comments, tracked change insertions/deletions, or highlighted content.
- **Author** - Displays all the authors of the edits in your resources. The authors are collected when indexing. You can set a specific author for your search or search all of them.
- **Time**- Specifies the time when the edits that you are searching through were created.

Both the view and the dialog box display the edits that contain the search results and their parent topics along with a short description. To hide this description, go to **Settings** and deselect the **Show Description** option.

Technical Aspects

When Oxygen XML Editor performs the indexing of your resources, the refereed content from your documents is not taken into account. For example, when DITA documents are indexed, the content from the `conref` elements is not parsed. The files that make up the index are stored on disk in the `[user_home_directory]\AppData\Roaming\com.oxygenxml\lucene` folder.

Using Projects to Group Documents

Oxygen XML Editor includes a ***Project*** *view* that helps you organize your projects. Oxygen XML Editor offers a variety of helpful features for working with projects and makes it easy to share your projects with other members of your team. This section presents various unique features that will help you to create and work with projects.

Creating a New Project

Oxygen XML Editor allows you to organize your XML-related files into projects. This helps you manage and organize your files and also allows you to perform batch operations (such as validation and transformation) over multiple files. You can also *share your project settings and transformation/validation scenarios* with other users. Use the ***Project*** *view* to manage projects, and the files and folders contained within.

Creating a New Project

To create a new project, select **New Project** from the **Project** menu, the **New** menu in the contextual menu, or the drop-down menu at the top-left of the **Project** view. This opens a dialog box that allows you to assign a name to the new project and adds it to the structure of the project in the ***Project*** *view*.

Adding Items to the Project

To add items to the project, select any of the following actions that are available when invoking the contextual menu in the **Project** view:

New > File

Opens a **New** file dialog box that helps you create a new file and adds it to the project structure.

New > Folder

Opens a **New Folder** dialog box that allows you to specify a name for a new folder and adds it to the structure of the project.

The project itself is considered a logical folder. You can add a logical folder, or content to a logical folder, by using one of the following actions that are available in the contextual menu, when invoked from the *project root*:

New > Logical Folder

Creates a logical folder in the tree structure (the icon is a magenta folder on Mac OS X -).

New > Logical Folders from Web

Replicates the structure of a remote folder accessible over FTP/SFTP/WebDAV, as a structure of logical folders. The newly created logical folders contain the file structure of the folder it points to.

Add Folder

Adds a link to a physical folder, whose name and content mirror a real folder that exists in the local file system (the icon of this action is different on Mac OS X).

Add Files

Adds links to files on the local file system.

Add Edited File

Adds a link to the currently edited file in the project.

Using Linked Folders (Shortcuts)

Another easy way to organize your XML working files is to place them in a directory and then to create a corresponding linked folder in you project. If you add new files to that folder, you can simply use the **Refresh (F5)** action from the toolbar or contextual menu and the ***Project*** *view* will display the existing files and subdirectories. If your files are scattered amongst several folders, but represent the same class of files, you might find it useful to combine them in a logical folder.

You can create linked folders (shortcuts) by dragging and dropping folders from the Windows Explorer (Mac OS X Finder) to the project tree, or by selecting **Add Folder** in the contextual menu from the *project root*. Linked folders are displayed in the ***Project*** *view* with bold text. To create a file inside a linked folder, select the **New** > **File** action from the contextual menu. The linked files presented in the ***Project*** *view* are marked with a special icon.

Note: Files may have multiple instances within the folder system, but cannot appear twice within the same folder.

For more information on managing projects and their content, see *Project View* on page 188.

For more details about how you can share projects with other users, see *Sharing a Project - Team Collaboration* on page 284.

Related Information:

Using Projects to Group Documents on page 275

Project View

The **Project** view is designed to assist you with organizing and managing related files grouped in the same XML project. The actions available in the contextual menu and on the toolbar associated to this panel allow you to create XML projects and provide shortcuts to various operations for the project documents.

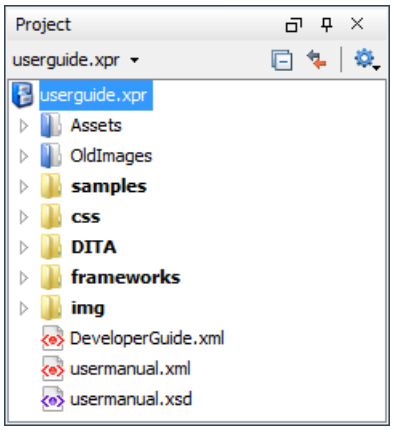


Figure 119: Project View

By default, the view is positioned on the left side of the Oxygen XML Editor window, above the ***Outline*** *view*. If the view has been closed, it can be reopened at any time from the **Window** > **Show View** menu (or using the **Show Project View** action from the **Project** menu).

The tree structure occupies most of the view area. In the upper left side of the view, there is a drop-down menu that contains all recently used projects and project management actions:

Open Project (Ctrl + F2 (Command + F2 on OS X))

Opens an existing project. Alternatively, you can open a project by dropping an Oxygen XML Editor XPR project file from the file explorer into the **Project** panel.

Notice: When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

New Project

Creates a new, empty project.

The following actions are grouped in the upper right corner:

Collapse All

Collapses all project tree folders. You can also collapse/expand a project tree folder if you select it and press the **Enter** key or **Left Arrow** to collapse and **Right Arrow** to expand.

Link with Editor

When selected, the project tree highlights the currently edited file, if it is found in the project files.

Note: This button is disabled automatically when you move to the **Debugger** *perspective*.

Settings

A submenu that contains the following actions:

Filters

Allows you to filter the information displayed in the **Project** view. Click the toolbar button to set filter patterns for the files you want to show or hide. Also, you can set filter patterns for the linked directories that are hidden.

Show Full Path

When selected, linked files and folders are presented with a full file path.

Enable Master Files Support

Select this option to enable the ***Master Files*** *support*.

Change Search and Refactor operations scope

Allows you to change the collection of documents that define the context of the *search and refactor operations*.

- **Use only Master Files, if enabled** - Restricts Oxygen XML Editor to perform the search and refactor operations starting from the *master files* that are defined for the current resource. This option is available when you select **Project** in the **Select the scope for Search and Refactor operations** dialog box and the **Master Files** support is enabled.
- **Working sets** - Allows you to specify the set of files that will be used for the scope of the search and refactor operations.

The files are usually organized in an XML project as a collection of folders. There are three types of resources displayed in the **Project** view:

- **Logical folders** - marked with a blue icon on Windows and Unix/Linux ( ) and a magenta icon on Mac OS X ( ). They help you group files within the project. This folder type has no correspondent on the physical disk, since they are used as containers for related items. Creating and deleting them does not affect the file system on disk. They are created on the project root or inside other logical folders by using the contextual action **New** > **Logical Folder**. The contextual menu action **Remove from Project** can be used to remove them from the project.
- **Physical folders and files** - marked with the operating system-specific icon for folders (usually a yellow icon on Windows and a blue icon on Mac OS X). These folders and files are mirrors of real folders or files that exist in the local file system. They are created or added to the project by using contextual menu actions (such as **New** > **File**, **New** > **Folder**, **Add Folder**, etc.) Also, the contextual menu action **Remove from Disk (Shift +Delete)** can be used to remove them from the project and local file system.
- **Shortcut folders and files** - the icons for file shortcuts include a shortcut symbol and names of folder shortcuts are displayed in bold text. All files and folders that appear as direct descendants of a logical folder are considered shortcuts. They are created and added with the contextual actions **Add Files** and **Add Folder** from the project root. Both contextual menu actions **Remove from Project** and **Remove from Disk (Shift +Delete)** are available for shortcuts. **Remove from Project** just removes the shortcut from the project, while **Remove from Disk (Shift+Delete)** removes the shortcut and the physical resource from the local file system.

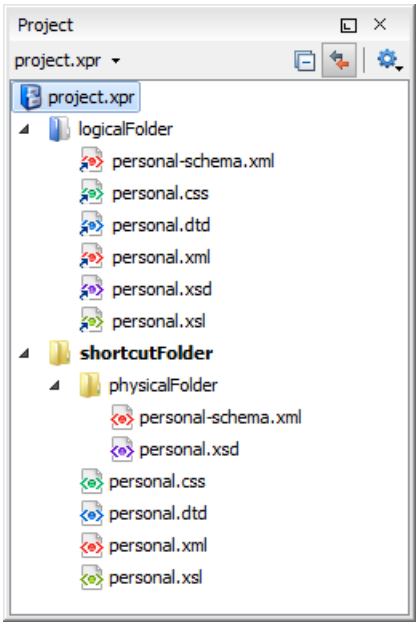


Figure 120: Project View with Examples of all Three Types of Resources

Creating New Projects

The following action is available in the **Project** menu, the **New** menu in the contextual menu, or from the drop-down menu in the top-left of the **Project** view:

New Project

Creates a new, empty project.

Creating New Project Items

The following actions are available by selecting **New** from the contextual menu, when invoked from the **Project** view:

New > File

Opens a ***New*** *file dialog box* that helps you create a new file and adds it to the project structure.

New > Folder

Opens a **New Folder** dialog box that allows you to specify a name for a new folder and adds it to the structure of the project.

New > Logical Folder

Available when invoked from the *project root*, this action creates a logical folder in the tree structure (the icon is a magenta folder on Mac OS X -).

New > Logical Folders from Web

Available when invoked from the *project root*, this action replicates the structure of a remote folder accessible over FTP/SFTP/WebDAV, as a structure of logical folders. The newly created logical folders contain the file structure of the folder it points to.

Managing Physical Folders and Files

You can create physical folders by selecting **New** > **Folder** from the contextual menu.

When adding files to a project, the default target is the project root. To change a target, select a new folder. Files may have multiple instances within the folder system, but cannot appear twice within the same folder.

Removing Files and Folders

To remove one or more linked files or folders, select them in the project tree and press the **Delete** key, or select the contextual menu action **Remove from Project**. To remove a linked file or folder from both project and local file system, select the contextual menu action **Remove from Disk (Shift+Delete)**. The **Remove from Disk (Shift+Delete)** action is also used to remove physical files or folders.

CAUTION: In most cases this action is irreversible, deleting the file permanently. Under particular circumstances (if you are running a Windows installation of Oxygen XML Editor and the *Recycle Bin* is active) the file is moved to *Recycle Bin*.

Moving Files and Folders

You can *move the resources of the project* with drag and drop operations on the files and folders of the tree (the **Enable drag-and-drop in Project view** option must be selected in the ***View*** *preferences page*).

You can also use the usual **Cut**, **Copy**, and **Paste** actions to move resources in the **Project** view.

Renaming Files and Folders

There are three ways you can *rename an item in the* ***Project*** *view*. Select the item in the **Project** view and do one of the following:

- Invoke the **Rename** action from the contextual menu.
- Press **F2** and type the new name.
- Click the selected item and type the new name.

  To finish editing the item name, press **Enter**.

Note: The **Rename** action is also available on *logical* files.

Locating and Opening Files

If a project folder contains a lot of documents, a certain document can be located quickly in the project tree by selecting the folder containing the desired document and typing the first few characters of the document name. The desired document is automatically selected as soon as the typed characters uniquely identify its name in the folder.

The selected document can be opened by pressing the **Enter** key, by double-clicking it, or with one of the **Open** actions from the contextual menu. The files with known document types are opened in the associated editor, while binary files are opened with the associated system application. To open a file with a known document type in an editor other than the default one, use the **Open with** action. Also, dragging and dropping files from the project tree to the editor area results in the files being opened.

Saving the Project

The project file is automatically saved every time the content of the **Project** view is saved or modified by actions such as adding or removing files and drag and drop.

Linked Folders

You can create linked folders (shortcuts) by dragging and dropping folders from a system explorer to the project tree (the **Enable drag-and-drop in Project view** option must be selected in the ***Views*** *preferences page*), or by selecting **Add Folder** in the contextual menu from the project root.

To create a file inside a linked folder, select the **New** > **File** action from the contextual menu. This opens the ***New Document*** *wizard*.

Note: The linked files presented in the **Project** view are marked with a special icon. Linked folders are displayed in bold text.

Logical Folders

The project itself is considered a logical folder. You can add content to a logical folder using one of the actions available in the contextual menu, when invoked from the *project root*:

Add Folder

Adds a link to a physical folder, whose name and content mirror a real folder that exists in the local file system (the icon of this action is different on Mac OS X).

Add Files

Adds links to files on the local file system.

Add Edited File

Adds a link to the currently edited file in the project.

Validate Files

The currently selected files in the **Project** view can be checked to be XML well-formed or validated against a schema (DTD, XML Schema, Relax NG, Schematron or NVDL) with one of the following contextual menu actions found in the **Validate** submenu:

Check Well-Formedness

Checks if the selected file or files are well-formed.

Validate

Validates the selected file or files against their associated schema. EPUB files make an exception, because this action triggers a ***Validate and Check for Completeness*** operation.

Validate with Schema

Validates the selected file of files against a specified schema.

Configure Validation Scenario(s)

Allows you to configure and run a *validation scenario*.

Applying Transformation Scenarios

The currently selected files in the **Project** view can be transformed in one step with one of the following actions available from contextual menu in the **Transform** submenu:

Apply Transformation Scenario(s)

Obtains the output with one of the *built-in scenarios*.

Configure Transformation Scenario(s)

Opens a *dialog box* that allows you to configure pre-defined transformation scenarios.

Transform with

Allows you to select a transformation scenario to be applied to the currently selected files.

Along with the logical folder support, this allows you to group your files and transform them very easily.

Refactoring Actions (Available for certain document types (such as XML, XSD, and XSL)

Oxygen XML Editor includes some refactoring operations that help you manage the structure of your documents. The following actions are available from the contextual menu in the **Refactoring** submenu:

Rename resource

Allows you to change the name of a resource.

Move resource

Allows you to change the location on disk of a resource.

XML Refactoring

Opens *the **XML Refactoring** tool wizard* that presents refactoring operations to assist you with managing the structure of your XML documents.

Other XML Refactoring Actions

For your convenience, the last 5 *XML Refactoring tool operations* that were finished or previewed will also appear in this submenu.

Other Contextual Menu Actions

Other actions that are available in the contextual menu from the project tree include:

Open

Opens the selected files in the corresponding editor.

Open with submenu

This submenu allows you to open the selected file with the internal editor, a system application, or other internal tools: ***DITA Maps Manager***, ***Archive Browser***, *MathML Editor*, *Generate/Convert Schema*, ***WSDL/SOAP Analyzer***, *Large File Viewer*, *Hex Viewer*, *SVG Viewer*.

Show in Explorer (or Show in Finder on OS X)

In Windows, the content of the selected folder or file is presented in a specific explorer window. On MAC OS X, the parent folder is opened and the selected folder is highlighted in a specific finder window.

Copy Location

Copies an application-specific URL for the selected resource to the clipboard.

Refresh

Refreshes the content and the dependencies between the resources in *the **Master Files** directory*.

Find/Replace in Files

Opens the ***Find/Replace in Files** dialog box* that allows you to find and replace text in multiple files.

XPath in Files

Opens the ***XPath/XQuery Builder** view* that allows you to compose XPath and XQuery expressions and execute them over the currently edited XML document.

Open/Find Resource

Opens the ***Open/Find Resource** dialog box*.

Check Spelling in Files

Allows you to *check the spelling of multiple files.*

Format and Indent Files

Opens the ***Format and Indent Files*** *dialog box* that allows you to configure the format and indent (*pretty-print*) action that will be applied on the selected documents.

Open in SVN Client

Syncro SVN Client tool is opened and it highlights the selected resource in its corresponding working copy.

Compare

Allows you to compare multiple files or directories and the order of your selection determines where they are opened in the ***Compare Files*** or ***Compare Directories*** tool. If you select two files or folders, your first selection will be opened in the left panel and the other one in the right panel.

You can also select 3 files and the tool will automatically be opened in the *three-way comparison mode*. If you select three files, your first selection will be opened in the left panel, the second in the right panel, and the third selection will be the base (ancestor) file.

Generate Documentation > XML Schema Documentation

Opens *the XML Schema Documentation Dialog Box*.

Generate Documentation > XSLT Stylesheet Documentation

Opens *the XSLT Stylesheet Documentation Dialog Box*.

Generate Documentation > XQuery Documentation

Opens *the XQuery Documentation Dialog Box*.

Generate Documentation > WSDL Documentation

Opens *the WSDL Documentation Dialog Box*.

Properties

Displays the properties of the current file in a **Properties** dialog box.

Menu Level Actions

The following actions are available in the **Project** menu:

New Project

Creates a new, empty project.

Open Project (Ctrl + F2 (Command + F2 on OS X))

Opens an existing project. Alternatively, you can open a project by dropping an Oxygen XML Editor XPR project file from the file explorer into the **Project** panel.

Notice: When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

Save Project As

Allows you to save the current project under a different name.

Validate all project files

Checks if the project files are well-formed and their mark-up conforms with the specified DTD, XML Schema, or Relax NG schema rules. It returns an error list in the message panel.

Filters

Opens the **Project filters** dialog box that allows you to decide which files and directories will be shown or hidden.

Enable Master Files Support

Allows you to enable the ***Master Files Support*** for each project you are working on.

Change Search and Refactor operations scope

Opens a dialog box that allows you to define the context of search and refactor operations.

Show Project View

Displays the **Project** view.

Reopen Project

Contains a list of links of previously used projects. This list can be emptied by invoking the **Clear history** action.

Moving/Renaming Resources in the Project View

The ***Project*** *view* allows you to move or rename files in the current project.

Moving Resources

To move a file or directory in the ***Project*** *view*, drag and drop it to the new location in the tree structure (the **Enable drag-and-drop in Project view** option must be selected in the ***View*** *preferences page*), or use the usual **Cut**, **Copy**, and **Paste** actions from the contextual menu or **Edit** menu.

You can also move certain types of files (such as XML, XML Schema, Relax NG, WSDL, and XSLT) by using the **Refactoring** > **Move resource** action from the contextual menu. This action opens the **Move resource** dialog box that includes the following options:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, based upon the selected scope. You can *select or configure the scope* by using the button.

Renaming Resources

To quickly rename a file or a directory, use the in-place editing either by pressing **F2** or by selecting **Rename** from the contextual menu.

You can also rename certain types of files (such as XML, XML Schema, Relax NG, WSDL, and XSLT) by using the **Refactoring** > **Rename resource** action from the contextual menu. This action opens the **Rename resource** dialog box that includes the following options:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references of the renamed resource** - Select this option to update the references to the resource you are renaming. You can *select or configure the scope* by using the button.

Problems Updating References of Moved/Renamed Resources

In some cases, the references of a moved or a renamed resource can not be updated. For example, when a resource is resolved through an *XML Catalog* or when the path to the moved or renamed resource contains entities. For these cases, Oxygen XML Editor displays a warning dialog box.

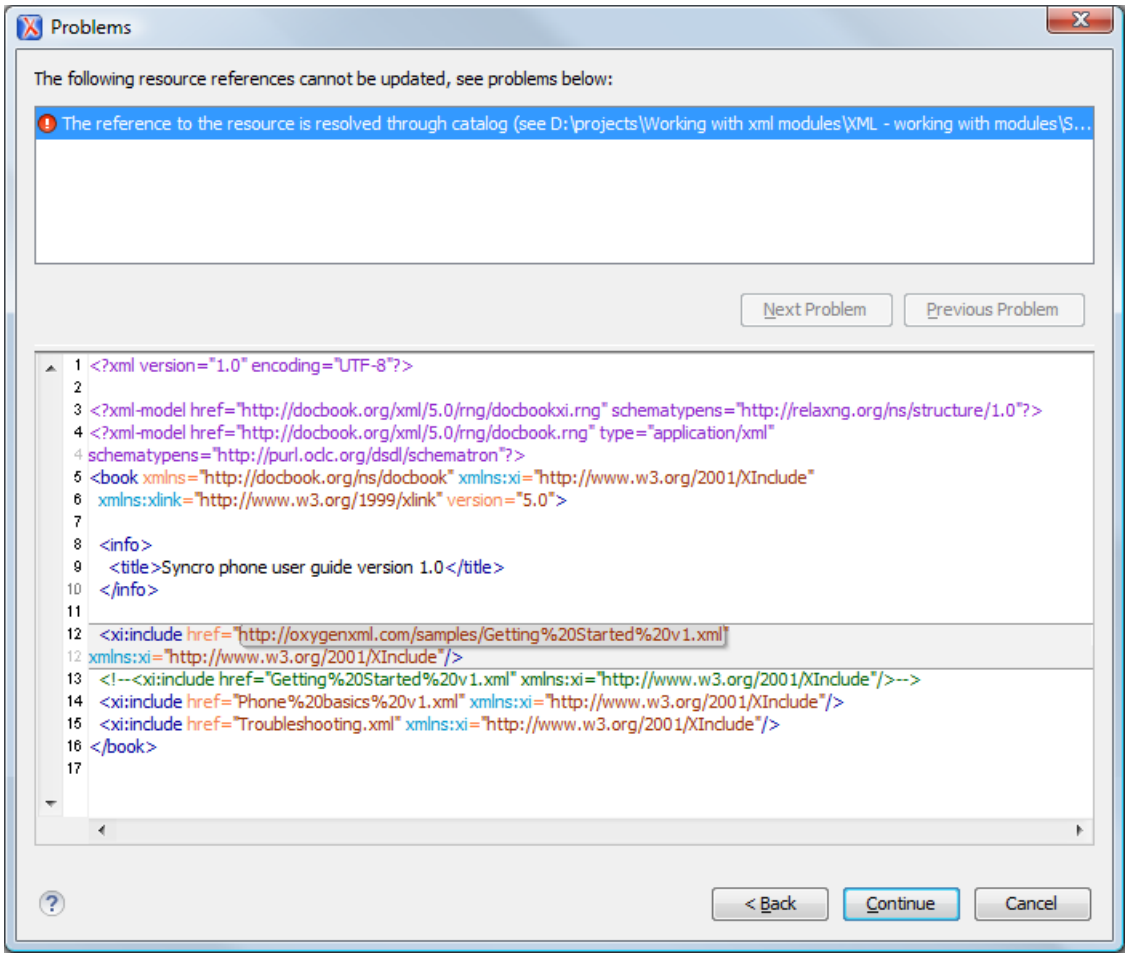


Figure 121: Problems Dialog Box

Image Preview from the Project View

Images and SVG files from the ***Project*** *view* can be previewed in a separate panel.

To preview an image, either double-click the image name or click the **Preview** action from the contextual menu of the ***Project*** *view*. Supported image types are GIF, JPEG/JPG, PNG, BMP. Once the image is displayed in the **Preview** panel using the actions from the contextual menu, you can scale the image to its original size (1:1 action) or scale it down to fit in the view's available area (**Scale to fit** action).

To preview an *SVG file*, click the **Preview** action from the contextual menu of the ***Project*** *view*. Once the SVG is displayed in the **Preview** panel, the following actions are available on the contextual menu: **Zoom in**, **Zoom out**, **Rotate** and **Refresh**.

Note: You can drag an image from the **Image Preview** view and drop it in a DITA, DocBook, or TEI document.

Sharing a Project - Team Collaboration

You can use XML projects to make team collaboration and synergy efficient and effective. Not only can you share the project files and folders, but Oxygen XML Editor also allows you to store preferences, transformation scenarios, and validation scenarios at *project level* in a *project file* (`.xpr` file extension). It can be saved on a version control system (such as SVN, CVS, or Source Safe) or in a shared folder, so that your team will have access to the same resources stored in the project file.

Sharing Preferences (Creating a Project-Level Options File)

To share options that are configured in certain preferences pages, you can store them in a *project file* (`.xpr` file extension) that can easily be shared with others. To do so, follow these steps:

1. You many want to use a fresh install for this procedure, to make sure that you do not copy personal or local preferences.
2. In the ***Project** view*, create a project or open an existing one.
3. *Open the **Preferences** dialog box **(Options** > **Preferences)***.
4. Configure the options in each preferences page that you want to be included in the project file and switch the storage preference to ***Project Options*** in each page.

   Note: Some pages do not have the **Project Options** button, since the options they host might contain sensitive data (such as passwords, for example) that is unsuitable for sharing with other users.
5. Click **OK** and close the **Preferences** dialog box.

   All explicitly set values are now saved in the project file. You can then share the project file so that your team will have the same option configuration that you stored in the project file.

   Note: The project file extension (`.xpr`) must be preserved when the file is distributed to others.

   Notice: When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

Sharing Transformation Scenarios

To share created and edited transformation scenarios, you can store them in a *project file* (`.xpr` file extension) by following these steps:

1. In the ***Project** view*, create a project or open an existing one.
2. When you create a new transformation scenario or edit an existing one, there is a **Storage** option. Switch the storage preference to ***Project Options*** in each transformation scenario you want to be included in the project file.
3. Click **OK** to store the scenario in the project file.

   You can then share the project file so that your team will have access to the same transformation scenarios that you stored in the project file. When you create a scenario at the project level, the URLs from the scenario become relative to the project URL.

   Note: The project file extension (`.xpr`) must be preserved when the file is distributed to others.

   Notice: When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

Sharing Validation Scenarios

To share created and edited validation scenarios, you can store them in a *project file* (`.xpr` file extension) by following these steps:

1. In the ***Project** view*, create a project or open an existing one.
2. When you create a new validation scenario or edit an existing one, there is a **Storage** option. Switch the storage preference to ***Project Options*** in each validation scenario you want to be included in the project file.
3. Click **OK** to store the scenario in the project file.

   You can then share the project file so that your team will have access to the same validation scenarios that you stored in the project file. When you create a scenario at the project level, the URLs from the scenario become relative to the project URL.

   Note: The project file extension (`.xpr`) must be preserved when the file is distributed to others.

   Notice: When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

Syncro SVN Client (Apache Subversion™)

To assist you with team collaboration and sharing projects, Oxygen XML Editor includes an embedded *SVN (Subversion) Client*. Even if you start developing a new project, or you want to migrate an existing one to Subversion, the Syncro SVN Client allows you to easily share it with the rest of your team.

It can be accessed from the **Tools** menu and can be used for synchronizing your working copy with a central repository.

It can also be started by selecting the **Open in SVN Client** action from the contextual menu of the ***Project*** *view*. This action opens the Syncro SVN Client and shows the selected project file in the **Working Copy** view.

Related Information:

Sharing Application Settings on page 163
Sharing Transformation Scenarios on page 929
Sharing Validation Scenarios on page 460

Minimize Differences Between Versions Saved on Multiple Computers

The number of differences between versions of the same file saved by multiple content authors on multiple computers can be minimized by imposing the same set of formatting options when saving the file, for all the content authors. An example, the following procedure can be used to minimize the differences:

1. Create an Oxygen XML Editor project file (`.xpr`) that will be shared by all content authors.
2. Configure your own formatting preferences. To do this, *open the* ***Preferences*** *dialog box* ***(Options > Preferences)***, go to **Editor** > **Format**, configure the appropriate options in this page, then go to **Editor** > **Format** > **XML** and configure the options there.
3. Save the configured options into your project file by selecting ***Project Options*** in both of the preferences pages.
4. Save the project and commit the project file to your versioning system so all the content authors can use it.
5. Make sure the project is opened in the ***Project*** *view*.
6. Open and save your XML files in the **Author** mode.
7. Commit the saved XML files to your versioning system.

When other content authors change the files, only the changed lines will be displayed in your diff tool instead of one big change that does not allow you to see the changes between two versions of the file.

Master Files Support

Oxygen XML Editor allows you to define *Master Files* at project level. These *master files* are automatically used by Oxygen XML Editor to determine the context for operations such as validation, content completion, refactoring, or searches for XML, XSD, XSL, WSDL, and RNG modules. Oxygen XML Editor maintains the hierarchy of the *master files*, helping you to determine the editing context.

Oxygen XML Editor also provides unique support for using the *Master Files* support in DITA projects. In DITA, when you rename or move non-DITA resources, it allows you to update all the references to these resources in the scope of the *Master Files* (in this case the *main DITA map (root map)*). For more information, see *Master Files Support in DITA* on page 1716.

To watch our video demonstrations about the *Master Files* support for XML documents, XSL documents, and WSDL documents, see *Working with Modular XML Files*, *Master Files Support*, and *Working with Modular WSDL Files*.

Master Files Benefits

Using the *Master Files* support in Oxygen XML Editor includes the following benefits:

- When the module is validated, Oxygen XML Editor automatically identifies the *master files* that include the module and validates all of them.
- The *Content Completion Assistant* presents all the components that are collected from the *master files* for the modules they include.
- The ***Outline*** *view* displays all the components that are defined in the *master files* hierarchy.
- The *master files* that are defined for the current module determines the *scope of the search and refactoring actions*. Oxygen XML Editor performs the search and refactoring actions in the context that the *master files* determine, thus improving the speed of execution.

Enabling the Master Files Support

Oxygen XML Editor stores the *master files* in a folder located in the ***Project** view*, as the first child of the project root. The *Master Files Support* is disabled by default and Oxygen XML Editor allows you to enable or disable the *Master Files Support* for each project you are working on.

To enable *Master Files* support, do one of the following:

- Select **Enable Master Files Support** from the **Settings** menu in the top-right corner of the ***Project** view*.
- Select **Enable Master Files Support** from the contextual menu of the project root folder in the ***Project** view*. If a disabled *Master Files* folder exists, you can also select that option from its contextual menu.
- Click the **Enable** button in the tooltip located at the bottom of the ***Project** view*. This tooltip window is displayed when the *Master Files* support is disabled. Clicking the **Read more** link takes you to the user guide. Clicking the **Enable** button opens the **Enable Master Files** dialog box. This dialog box contains general information about the **Master Files Support** and allows you to enable it. You can also use the **Detect and Enable** button in this dialog box to detect the *master files* from the current project.

  Warning: Once you close this window tip, Oxygen XML Editor hides it for all projects. You can make the window tip reappear by *resetting Oxygen XML Editor to its default settings*. However, doing so will result in you losing your customized options.

Related Information:

Detecting Master Files on page 287

Adding Files to the Master File Directory on page 288

Detecting Master Files

Oxygen XML Editor allows you to detect the *master files* using the **Detect Master Files** option. This action applies to the folders you select in the project.

To detect *master files* over the entire project, do one of the following:

- Right-click the root of the project and select **Detect Master Files**.
- Use the **Detect Master Files from Project** option, available in the contextual menu of the `Master Files` folder.

Both of these options display the **Detect Master Files** wizard. In the first panel you can select the type of *master files* you want Oxygen XML Editor to detect. In the subsequent panel the detected *master files* are presented in a tree-like fashion.The resources are grouped into three categories:

- **Possible *master files*** - The files presented on the first level in this category are not imported or included from other files. These files are most likely to be set as *master files*.

  Note: For DITA projects, only *DITA Maps* are reported as possible *master files*.
- **Cycles** - The files that are presented on the first level have circular dependencies between them. Any of the files presented on the first level of a cycle is a possible *master file*.
- **Standalone** - Files that do not include or import other files and are also not included or imported themselves. It is not necessary to set them as *master files*.

To set them as *master files*, simply select their checkboxes. Oxygen XML Editor marks all the children of a *master file* as modules. Modules are rendered in gray and their tool-tip presents a list of their *master files*. A module can be accessed from multiple *master files*.

The *master files* that are already defined in the project are automatically marked in the tree and cannot be removed. The only way to disable a *master file* is to delete it from the `Master Files` folder.

The next panel displays a list with the selected *master files*. Click the **Finish** button to add the *master files* in the `Master Files` folder.

You can use the **Select Master Files** option to automatically mark all *master files*. This action sets all the resources from the **Possible Master Files** category and the first resource of each **Cycle** as *master files* . The **Deselect All** button simply removes all of your selections.

Tip: We recommend that you to only add top-level files (files that are at the root of the include/import graph) in the `Master Files` directory. Keep the file set to a minimum and only add files that import or include other files.

Attention: If the **Master Files Support** is disabled, the `Master Files` directory is rendered only if it contains *master files*.

Related Information:

Enabling the Master Files Support on page 287
Adding Files to the Master File Directory on page 288

Adding Files to the Master File Directory

The `Master Files` directory only contains logical folders and linked files. To add files in the `Master Files` directory, use one of the following methods:

- Right-click a file from your project and select **Add to Master Files** from the contextual menu.
- Select **Add Files** or **Add Edited File** from the contextual menu of the `Master Files` directory.
- Drag and drop files into the `Master Files` directory.
- From the contextual menu of the ***Resource Hierarchy Dependencies*** *view*, use the **Add to Master Files** action.

You can view the *master files* for the current resource by selecting ***Properties*** *from the contextual menu* of the ***Project*** *view* and the *master files* for the current editor in the ***Properties*** and ***Information*** views.

Related Information:

Enabling the Master Files Support on page 287
Detecting Master Files on page 287

Project Validation and Transformation

The *Master Files Support* is also useful for *project-level* validation and transformation scenarios. When you hover the cursor over a file in the `Master Files` directory, Oxygen XML Editor displays the **Validate** and **Transform** buttons at the right of the file. Select one of these buttons to run a transformation or validation scenario. If the current node is selected, Oxygen XML Editor executes a batch transformation and validation. If the current node is not selected, Oxygen XML Editor executes the validation and transformation for the current node only. The behavior of these actions is the same as the behavior of the *corresponding actions that are available in the contextual menu*.

Note: The tooltip of the **Validate** and **Transform** buttons displays the associated scenarios that you can apply.

When you hover the cursor over the `Master Files` directory itself, Oxygen XML Editor also displays a **Help** button. Use this button (or press **F1** on your keyboard) to open the **Help** section regarding the *Master Files Support*.

Contextual Menu of the Master Files

The contextual menu of the `Master Files` directory contains the following actions:

New

Allows you to create a **File**, **Logical Folder**, or **Project**.

Add Files

Allows you to add *master files* to the `Master Files` directory.

Add Edited File

Use this option to add the currently edited file to the `Master Files` directory.

Open

Opens all the files of the `Master Files` directory.

Paste

Pastes the files you copy in the `Master Files` directory.

Rename

Allows you to rename a file in the `Master Files` directory.

Refresh

Refreshes the content of the `Master Files` directory.

Find/Replace in Files

Opens the ***Find/Replace*** *dialog box*.

XPath in Files

Opens the ***XPath/XQuery Builder*** *view* that allows you to compose XPath and XQuery expressions and execute them over the currently edited XML document.

Open/Find Resource

Opens the ***Open/Find Resource*** *dialog box*.

Check Spelling in Files

Opens the ***Check Spelling in Files*** *dialog box*.

Format and Indent Files

Opens the ***Format and Indent Files*** *dialog box* that allows you to configure the format and indent (*pretty-print*) action that will be applied on the selected documents.

Transform

Provides access to one of the following actions:

Apply Transformations Scenario(s)

Applies the transformation scenarios associated with the `Master Files` directory.

Configure Transformation Scenario(s)

Opens the ***Configure Transformation Scenario*** *dialog box*.

Transform with

Opens the **Transform with** dialog box that allows you to select the transformation scenario you want to execute.

Validate

Provides access to one of the following actions:

Check Well-Formedness

Allows you to check if a document is *Namespace Well-Formed XML*.

Validate

Oxygen XML Editor performs the validation of the *master files*.

Validate with Schema

Opens the **Validate with** dialog box. Oxygen XML Editor performs the validation of the *master files* using a schema.

Configure Validation Scenario(s)

Opens the ***Configure Validation Scenario*** *dialog box*.

Detect Master Files from Project

Enables automatic detection of *master files*.

Enable Master Files Support

Select this option to enable the **Master Files Support**.

Editing XML Documents

This section explains the various features in Oxygen XML Editor for editing XML documents. It includes information about the user interface components and actions that are available in the various editing modes and numerous features to help you edit XML documents in any mode.

Related Information:

Text Editing Mode on page 186
Author Editing Mode on page 211
Grid Editing Mode on page 208

Editing XML Documents in Text Mode

The Oxygen XML Editor **Text** editing mode is designed to be a simple, yet powerful, XML source editor. You can use this mode to edit XML code, markup, and text and it provides support to help you transform, and debug XML-based documents. It is similar to other common text editors, but Oxygen XML Editor also includes specialized editing actions, a powerful *Content Completion Assistant*, a helpful ***Outline** view*, and many other unique features.

To switch to this mode, select **Text** at the bottom of the editing area.

Navigating the Document Content in Text Mode

Oxygen XML Editor includes some useful features to help you navigate XML documents in **Text** mode.

Using the Keyboard

Oxygen XML Editor allows you to quickly navigate through a document using the **Ctrl + CloseBracket (Command + CloseBracket on OS X)** key to go to the next XML node and **Ctrl + OpenBracket (Command + OpenBracket on OS X)** to go to the previous one.

To navigate one word forward or backwards, use **Ctrl + RightArrow (Command + RightArrow on OS X)**, and **Ctrl + LeftArrow (Command + LeftArrow on OS X)**, respectively. To position the cursor at the beginning or end of the document you can use **Ctrl + Home (Command + Home on OS X)**, and **Ctrl + End (Command + End on OS X)**, respectively.

Navigation Buttons

Oxygen XML Editor includes some actions that help you to quickly navigate to a particular modification. These navigation buttons are available in the main toolbar and the actions can also be accessed from the **Find** menu. The three actions include:

- **Last Modification** - Moves the cursor to the last modification in any opened document.
- **Back** - Moves the cursor to the previous position.
- **Forward** - Moves the cursor to the next position. Available after you use the **Back** button at least once.

Navigating with the Outline View

Oxygen XML Editor includes a very useful ***Outline** view* that displays a hierarchical tag overview of the currently edited XML Document.

You can use this view to quickly navigate through the current document by selecting nodes in the outline tree. It is synchronized with the editor area, so when you make a selection in the **Outline** view, the corresponding nodes are highlighted in the editor area.

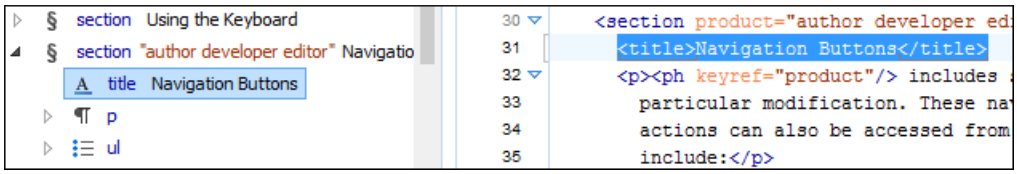


Figure 122: Outline View Navigation in Text Mode

Using the Breadcrumb to Navigate

A *breadcrumb* on the top stripe indicates the path from the document root element to the current element. It can also be used as a helpful tool to navigate to specific elements throughout the structure of the document.

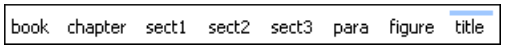


Figure 123: Breadcrumb in Text Mode

The last element listed in the *breadcrumb* is the element at the current cursor position. The current element is also highlighted by a thin light blue bar for easy identification. Clicking an element from the *breadcrumb* selects the entire element and navigates to it in the editor area.

Navigating with the Go To Dialog Box

In **Text** mode, you can navigate precisely to a location in the document you are editing by using the **Go to** dialog box. To open this dialog box, go to **Find** > **Go to** **(Ctrl+L (Command+L on OS X))**.

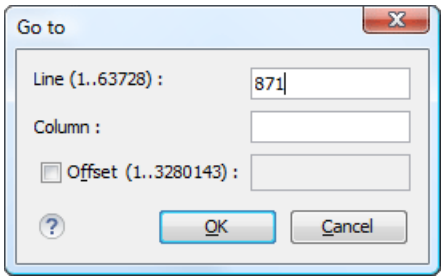


Figure 124: Go to Dialog Box

The dialog box includes the following fields for specifying a specific navigation location:

- **Line** - Destination line in the current document.
- **Column** - Destination column in the current document.
- **Offset** - Destination offset relative to the beginning of document.

Navigating with Bookmarks

By using *bookmarks*, you can mark positions in an edited document so that you can return to it later. This is especially helpful for navigating through large documents or while editing multiple documents. You can place up to nine distinct *bookmarks* in any document. Shortcut keys are available to place the *bookmarks* or to return to any of the marked positions. You can configure these shortcut keys in the ***Options*** > ***Menu Shortcut Keys*** menu.

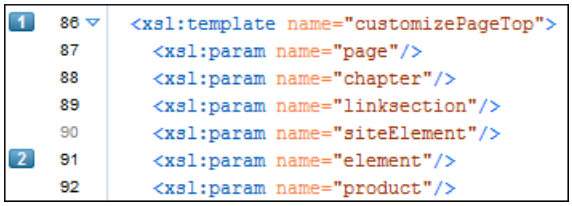


Figure 125: Editor Bookmarks

A *bookmark* can be inserted in **Text** mode by doing one of the following:

- Click in the vertical stripe on the left side of the editor (to the left of the line number).

- Select the **Create Bookmark (F9)** action from the **Edit** > **Bookmarks** menu.

A *bookmark* can be removed by right-clicking its icon on the vertical stripe and selecting **Remove** or **Remove all (Ctrl+F7 (Command+F7 on OS X))**.

You can navigate the *bookmarks* by using one of the actions available on the **Edit** > **Bookmarks** > **Go to** menu or by using the shortcut keys that are listed in that menu.

Smart Editing in Text Mode

Oxygen XML Editor includes *smart editing* features to help you edit XML documents in **Text** mode. The following smart editing features are included:

- **Closing tag auto-expansion** - This feature helps save some keystrokes by automatically inserting a closing tag when you insert a complete start tag and the cursor is automatically placed in between the start and end tags. For instance, after entering a start `<tag>`, the corresponding closing `</tag>` is automatically inserted and the cursor is placed between the two (`<tag>|</tag>`.
- **Auto-rename matching tag** - When you edit the name of a start tag, Oxygen XML Editor will mirror-edit the name of the matching end tag. This feature can be controlled from the ***Content Completion*** *option page*.
- **Auto-breaking the edited line** - The ***Hard line wrap*** *option* automatically breaks the edited line when its length exceeds the maximum line length *defined for the format and indent operation*.
- **Indent on Enter** - The ***Indent on Enter*** *option* indents the new line inserted when you press **Enter**.
- **Smart Enter** - The ***Smart Enter*** *option* inserts an empty line between the start and end tags. If you press **Enter** between a start and end tag, the action places the cursor in an indented position on the empty line between the lines that contain the start and end tag.
- **Double-click** - A double-click selects certain text, depending on the position of the click in the document:
  - If the click position is on a start tag or end tag, then the element name is selected.
  - If the click position is immediately after the opening quote or immediately before the closing quote of an attribute value, then the entire attribute value is selected.
  - Otherwise, a double-click selects contiguous text.
- **Triple-click** - A triple-click selects entire regions of text, depending on the click position:
  - If the click position is on a start or end tag, then the entire tag is selected, including the start and end tags, and the content in between.
  - If the click position is after a start tag or before an end tag, then the entire content of the element without the start and end tags is selected.
  - If the click position is before a start tag or after an end tag, then the entire tag is selected, including the start and end tags, and the content in between.
  - If the click position is immediately before an attribute, then the entire attribute and its value is selected.
  - If the click position is in between the opening and closing quotes of an attribute value, then the entire attribute value is selected.
  - Otherwise, it selects the entire current line of text.

Shortcut Actions in Text Mode

Oxygen XML Editor includes numerous shortcut actions to help you edit content in the **Text** editing mode.

Changing the Font Size (Zoom)

The font size of the editor panel can be changed with the following actions that are available with shortcuts or in the **Document** > **Font size** menu:

Increase editor font (Ctrl + NumPad+ (Command + NumPad+ on OS X) or Ctrl + MouseWheelForward (Windows/Linux)

Increases the font size (zooms in) with one point for each execution of the action.

Note: For Mac OS X, if you activate the ***Enable mouse-wheel zooming*** *option* in the **Editor** preferences page, you can use **Command + MouseWheelForward** to increase the font size (zoom in). It is disabled by default due to the way inertia affects the mouse wheel on Mac OS X.

Decrease editor font (Ctrl + NumPad- (Command + NumPad- on OS X) or Ctrl + MouseWheelBackwards (Windows/Linux)

Decreases the font size (zooms out) with one point for each execution of the action.

Note: For Mac OS X, if you activate the ***Enable mouse-wheel zooming*** *option* in the **Editor** preferences page, you can use **Command + MouseWheelBackwards** to decrease the font size (zoom out). It is disabled by default due to the way inertia affects the mouse wheel on Mac OS X.

Normal editor font (Ctrl + 0 (Command + 0 on OS X))

Resets the font size to *the value of the editor font set in the* ***Fonts*** *preferences page*.

Undo/Redo Actions

The typical undo and redo actions are available with shortcuts or in the **Edit** menu:

Undo (Ctrl + Z (Command + Z on OS X))

Reverses a maximum of 200 editing actions (configurable with the ***Undo history size*** *option* in the **Editor** preferences page) to return to the preceding state.

Note: Complex operations such as **Replace All** or **Indent selection** count as single undo events.

Redo (Ctrl + Y (Command + Shift + Z on OS X, Ctrl + Shift + Z on Linux/Unix))

Recreates a maximum of 100 editing actions that were undone by the **Undo** function.

Copy and Paste Actions

The typical copying and pasting actions are available with shortcuts or in the contextual menu (or the **Edit** menu):

Cut (Ctrl + X (Command + X on OS X))

Removes the current selected content from the document and places it in the clipboard.

Copy (Ctrl + C (Command + C on OS X))

Places a copy of the current selected content in the clipboard.

Paste (Ctrl + V (Command + V on OS X))

Inserts the current clipboard content into the document at the cursor position.

Select All (Ctrl + A (Command + A on OS X))

Selects the entire content of the current document.

Moving XML Nodes

You can use the following shortcuts to move XML elements or XSLT variables up or down in **Text** mode:

Ctrl + Alt + UpArrow (Command + Alt + UpArrow on OS X)

Moves the node up one line.

Ctrl + Alt + DownArrow (Command + Alt + DownArrow on OS X)

Moves the node down one line.

Note: The requirements for these node moving actions to work are as follows:

- The mechanism is designed to work without a selection. If you use these actions on a selection of content, it moves the entire selection. To make this mechanism work as intended, simply position the cursor somewhere on the line that you want to move.
- A start tag must be the first text occurrence on the line where the cursor is positioned.
- On the line where the element ends, only whitespaces are allowed after the end tag.

Miscellaneous Shortcut Actions in Text Mode

Oxygen XML Editor also includes the following other miscellaneous shortcut actions in **Text** mode:

Ctrl + Delete (Command + Delete on OS X)

Deletes the next word.

Ctrl + Backspace (Command + Backspace on OS X)

Deletes the previous word.

Ctrl + W (Command + W on OS X)

Cuts the previous word.

Ctrl + K (Command + K on OS X)

Cuts to end of line.

Ctrl + Single-Click (Command + Single-Click on OS X)

Use this shortcut to open any of the following:

- Any absolute URL (URLs that have a protocol), regardless of their location in the document.
- URI attributes such as: `schemaLocation`, `noNamespaceSchemaLocation`, `href` and others.
- Processing instructions used for associating resources, xml-models, xml-stylesheets.

Ctrl + Shift + Y (Command + Shift + Y on OS X) (Document > Edit > Toggle Line Wrap)

Enables or disables line wrapping. When enabled, if text exceeds the width of the displayed editor, content is wrapped so that you do not have to scroll horizontally.

Related Information:

Frequently Used Shortcut Keys on page 18

Editing XML Markup in Text Mode

Oxygen XML Editor includes some useful actions that allow you to easily edit XML markup in **Text** mode. These actions are available in the **Refactoring** submenu of the contextual menu and in the **Document** > **Markup** menu, and many of the actions can also be done with simple keyboard shortcuts.

Using the Breadcrumb

A *breadcrumb* on the top stripe indicates the path from the document root element to the current element. It can also be used as a helpful tool to insert and edit specific elements in the document structure.

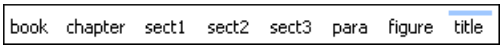


Figure 126: Breadcrumb in Text Mode

The last element listed in the *breadcrumb* is the element at the current cursor position. The current element is also highlighted by a thin light blue bar for easy identification. Clicking an element in the *breadcrumb* selects the entire element in the editor area. Also, each element provides a contextual menu with access to the following actions:

Append Child

Allows you to select an element (from a drop-down list) that is allowed by the associated schema and inserts it as a child of the current element.

Insert Before

Allows you to select an element (from a drop-down list) that is allowed by the associated schema and inserts it immediately before the current element, as a sibling.

Insert After

Allows you to select an element (from a drop-down list) that is allowed by the associated schema and inserts it immediately after the current element, as a sibling.

Edit Attributes

Opens an editing window that allows you to edit the attributes of the currently selected element.

Toggle Comment

Encloses the currently selected element in an XML comment, if the element is not already commented. If it is already commented, this action will remove the comment.

Cut

Removes the selected element and copies it to the clipboard.

Copy

Copies the selected element to the clipboard.

Delete

Deletes the currently selected element.

Move Nodes

You can easily move XML nodes in the current document by using the following shortcut keys:

Alt + UpArrow

Moves the current node or selected nodes in front of the previous node.

Alt + DownArrow

Moves the current node or selected nodes after the subsequent node.

Rename Elements

You can rename elements by using the following actions in the **Refactoring** submenu of the contextual menu (or from the **Document** > **Markup** menu):

Rename Element

The element from the cursor position, and any elements with the same name, can be renamed according with the options from the **Rename** dialog box.

Rename Prefix (Alt + Shift + P (Command + Shift + P on OS X))

The prefix of the element from the cursor position, and any elements with the same prefix, can be renamed according with the options from the **Rename** dialog box.

- If you select the **Rename current element prefix** option, the application will recursively traverse the current element and all its children. *For example*, to change the `xmlns:p1="ns1"` association in the current element to `xmlns:p5="ns1"`, if the `xmlns:p1="ns1"` association is applied on the parent element, then Oxygen XML Editor will introduce `xmlns:p5="ns1"` as a new declaration in the current element and will change the prefix from p1 to p5. If p5 is already associated with another namespace in the current element, then the conflict will be displayed in a dialog box. By pressing **OK**, the prefix is modified from p1 to p5 without inserting a new declaration.
- If you select the **Rename current prefix in all document** option, the application will apply the change on the entire document.
- To also apply the action inside attribute values, select the **Rename also attribute values that start with the same prefix** checkbox.

Surround Content with Tags (Wrap)

You can surround a selection of content with tags (*wrap* the content) by using the following action in the **Refactoring** submenu of the contextual menu (or from the **Document** > **Markup** menu):

Surround with submenu

Presents a drop-down menu that allows you to choose a tag to surround a selected portion of content.

Surround with Tags (Ctrl + E (Command + E on OS X))

Allows you to choose a tag that encloses a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.

- If the ***Position cursor between tags*** *option* is selected in the **Content Completion** preferences page, the cursor is placed between the start and end tag.
- If the ***Position cursor between tags*** *option* is not selected in the **Content Completion** preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

Surround with '[tag]' (Ctrl + ForwardSlash (Command + ForwardSlash on OS X))

Surround the selected content with the last tag used.

Unwrap the Content of Elements

You can unwrap the content of an element by using the following action in the **Refactoring** submenu of the contextual menu (or from the **Document** > **Markup** menu):

Delete element tags (Alt + Shift + X (Command + Alt + X on OS X))

Deletes the start and end tag of the current element.

Join or Split Elements

You can join or split elements in the current document by using the following actions in the **Refactoring** submenu of the contextual menu (or from the **Document** > **Markup** menu):

Join elements (Alt + Shift + J (Command + Alt + J on OS X))

Joins the left and right elements relative to the current cursor position. The elements must have the same name, attributes, and attributes values.

Split element (Alt + Shift + D (Ctrl + Alt + D on OS X))

Split the element from the cursor position into two identical elements. The cursor must be inside the element.

Other Refactoring Actions

You can also manage the structure of the markup by using the other specific XML refactoring actions that are available in the **Refactoring** submenu of the contextual menu:

Attributes submenu

Contains predefined XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

Add/Change attribute

Allows you to change the value of an attribute or insert a new one.

Delete attribute

Allows you to remove one or more attributes.

Rename attribute

Allows you to rename an attribute.

Replace in attribute value

Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

Comments submenu

Contains predefined XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

Delete comments

Allows you to delete comments found inside one or more elements.

DITA submenu

Contains predefined XML refactoring operations that pertain to DITA documents with some of the information preconfigured based upon the current context.

Change topic ID to file name

Use this operation to change the ID of a topic to be the same as its file name.

Convert conrefs to conkeyrefs

Use this operation to convert `conref` attributes to `conkeyref` attributes.

Convert simple tables to CALS tables

Use this operation to convert DITA simple tables to CALS tables.

Convert to Concept

Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept).

Convert to Reference

Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference).

Convert to Task

Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task).

Convert to Topic

Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic).

Convert to Troubleshooting

Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting).

Elements submenu

Contains predefined XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

Delete element

Allows you to delete elements.

Delete element content

Allows you to delete the content of elements.

Insert element

Allows you to insert new elements.

Rename element

Allows you to rename elements.

Unwrap element

Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

Wrap element

Allows you to surround elements with element tags.

Wrap element content

Allows you to surround the content of elements with element tags.

Fragments submenu

Contains predefined XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

Insert XML fragment

Allows you to insert an XML fragment.

Replace element content with XML fragment

Allows you to replace the content of elements with an XML fragment.

Replace element with XML fragment

Allows you to replace elements with an XML fragment.

JATSKit submenu

Contains predefined XML refactoring operations that pertain to JATS documents with some of the information preconfigured based upon the current context.

Add BITS DOCTYPE - NLM/NCBI Book Interchange 2.0

Adds an NLM 'BITS' 2.0 DOCTYPE declaration.

Add Blue DOCTYPE - NISO JATS Publishing 1.1

Adds a JATS 'Blue' 1.1 DOCTYPE declaration.

Normalize IDs

Assigned IDs are normalized and IDs are assigned to some elements that are missing them.

Related Information:

Refactoring XML Documents on page 491
Contextual Menu Actions in Text Mode on page 318
Frequently Used Shortcut Keys on page 18

Folding XML Elements in Text Mode

When working with a large document, the *folding* support in Oxygen XML Editor can be used to collapse some element content leaving only those that you need to edit in focus. Expanding and collapsing works on individual elements. Expanding an element leaves the child elements unchanged.

By default, the *folding* feature is enabled in Oxygen XML Editor, but it can be disabled in the **Text** preferences page with the ***Enable folding*** *option*.

Figure 127: Folding of XML Elements in Text Mode

The fact that the folds are persistent is a unique feature of Oxygen XML Editor. The next time you open the document the folds are restored to its last state.

Folding Actions in Text Mode

Element folds are marked with a small triangle (/) in the left stripe. To toggle the fold, simply click the icon. Also, if you right-click the icon, the following actions are available:

Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on OS X))

Folds all the elements except the current element.

Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on OS X))

Folds the child elements that are indented one level inside the current element.

Expand Child Folds

Unfolds all child elements of the currently selected element.

Expand All (Ctrl + NumPad* (Command + NumPad* on OS X))

Unfolds all elements in the current document.

To watch our video demonstration about the *folding* support in Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/FoldingSupport.html*.

Drag and Drop in Text Mode

To move a whole region of text to other location in the same edited document, just select the text, drag the selection by holding down the left mouse button and drop it to the target location.

You can also copy content from other applications and paste it into the document.

Selecting Content in Text Mode

Oxygen XML Editor includes a variety of keyboard shortcuts that allow you to select content in **Text** mode. These include numerous standard continuous selection possibilities that are common to many text editors, as well as a selection feature that allows you to select a rectangular area within a document in **Text** mode.

Standard Continuous Selection Shortcuts

Ctrl + A (Meta + A on Mac OS X)

Selects all content in the document.

Shift + Left/Right Arrow Keys

Begins a continuous selection at the cursor position and extends it one character at a time in the direction that you press the arrow keys.

Shift + Up/Down Arrow Keys

Begins a continuous selection at the cursor position and extends it one line at a time in the direction that you press the arrow keys.

Ctrl + Shift + Left/Right Arrow Keys (Meta + Shift + Left/Right Arrow Keys on Mac OS X)

Begins a continuous selection at the cursor position and extends it one word at a time in the direction that you press the arrow keys.

Shift + Home

Begins a continuous selection at the cursor position and extends it to the beginning of the current line (on Mac OS X, it extends to the beginning of the document).

Shift + End

Begins a continuous selection at the cursor position and extends it to the end of the current line (on Mac OS X, it extends to the end of the document).

Ctrl + Shift + Home

Begins a continuous selection at the cursor position and extends it to the beginning of the document.

Ctrl + Shift + End

Begins a continuous selection at the cursor position and extends it to the end of the document.

Shift + PageUp

Begins a continuous selection at the cursor position and extends it up one screen page.

Shift + PageDown

Begins a continuous selection at the cursor position and extends it down one screen page.

Double-Click

Selects certain text, depending on the position of the click in the document. See *Smart Editing Double-Click* for the specifics.

Triple-Click

Selects entire regions of text, depending on the position of the click in the document. See the *Smart Editing Triple-Click* for the specifics.

Right-Click > Select > Element

Selects the entire element at the current cursor position.

Right-Click > Select > Content

Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

Right-Click > Select > Attributes

Selects all the attributes of the element at the current cursor position.

Right-Click > Select > Parent

Selects the entire parent element at the current cursor position.

Rectangular Selection Shortcuts

Oxygen XML Editor also includes some keyboard shortcuts that allow you to select a rectangular block of content in **Text** mode and you can then copy, cut, paste, delete, or edit the selection.

Attention: The rectangular selection shortcuts will not work if the ***Line Wrap*** *option* is selected in the **Text** preferences page.

The following shortcuts can be used to create a rectangular selection:

Alt + Mouse Click + Mouse Movement (Alt + Meta + Mouse Click + Mouse Movement on Mac OS X)

Begins a rectangular selection at the mouse click position and extends it in the direction that you move the mouse. Release **Alt (Alt + Meta on Mac OS X)** to enter the *in-place editing mode*.

Shift + Alt + Left/Right Arrow Keys (Shift + Alt + Meta + Left/Right Arrow Keys on Mac OS X)

Begins a rectangular selection at the cursor position and extends it one character at a time in the direction that you press the arrow keys (you can also use the mouse to extend the selection).

Shift + Alt + Up/Down Arrow Keys (Shift + Alt + Meta + Up/Down Arrow Keys on Mac OS X)

Begins a rectangular selection at the cursor position and extends it one line at a time in the direction that you press the arrow keys (you can also use the mouse to extend the selection).

Ctrl + Shift + Alt + Left/Right Arrow Keys (Ctrl + Shift + Alt + Meta + Left/Right Arrow Keys on Mac OS X)

Begins a rectangular selection at the cursor position and extends it one word at a time in the direction that you press the arrow keys.

Shift + Alt + Home (Shift + Alt + Meta + Home on Mac OS X)

Begins a rectangular selection at the cursor position and extends it to the beginning of the current line.

Shift + Alt + End (Shift + Alt + Meta + End on Mac OS X)

Begins a rectangular selection at the cursor position and extends it to the end of the current line.

Shift + Alt + PageUp (Shift + Alt + Meta + PageUp on Mac OS X)

Begins a rectangular selection at the cursor position and extends it up one screen page.

Shift + Alt + PageDown (Shift + Alt + Meta + PageDown on Mac OS X)

Begins a rectangular selection at the cursor position and extends it down one screen page.

You can then use standard editing actions to copy, cut, paste, or delete the entire selection.

In-Place Editing Mode

To edit the content of the rectangular selection, you can enter an in-place editing mode by releasing the **Alt** key (on Mac OS X, release both **Alt** & **Meta**). Once you are in the editing mode, you can simply use your keyboard to edit the entire selection of content, or click anywhere inside the selection to edit the content at the cursor position for all lines within the selection at once (as if the rectangular selection is a selection of columns). To exit the editing mode, press either **Enter** or **Esc**.

Content Completion Assistant in Text Mode

Oxygen XML Editor includes an intelligent *Content Completion Assistant* that offers rapid, inline identification and insertion of structured language elements, attributes, and attribute values. Oxygen XML Editor shows the available entries that are valid in the current editing context.

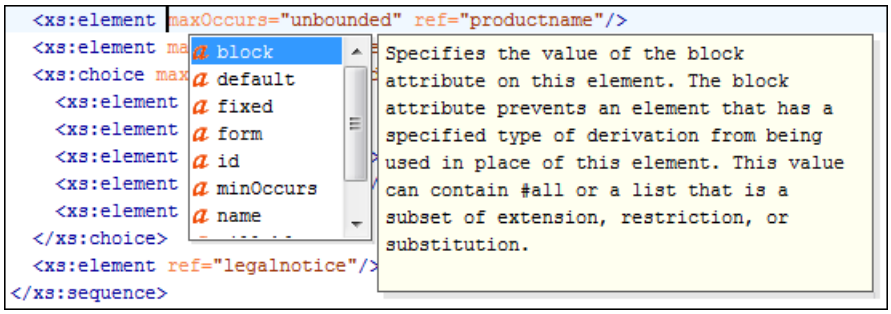


Figure 128: Content Completion Assistant

The *Content Completion Assistant* feature is schema-driven (XML Schema, DTD, and RELAX NG) and status information about the detected schema is logged in the *Information view*.

The *Content Completion Assistant* is enabled by default. To disable it, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Editor** > **Content Completion**, and deselect the ***Enable content completion** option*.

Using the Content Completion Assistant in Text Mode

The feature is activated in **Text** mode in the following situations:

- After you press the **<** character when inserting an element, it is automatically activated after a short delay. You can adjust the activation delay with the ***Activation delay of the proposals window (ms)** option* from the **Content Completion** preferences page.
- After typing a partial element or attribute name, you can manually activate it by pressing **Ctrl + Space (Command + Space on OS X)** or **Alt + ForwardSlash (Command + Alt + ForwardSlash on OS X)**. If there is only one valid proposal at the current location, it is inserted without displaying the list of proposals.

You can navigate through the list of proposals by using the **Up** and **Down** keys on your keyboard. For each selected item in the list, the *Content Completion Assistant* displays a documentation window. You can customize the size of the documentation window by dragging its top, right, and bottom borders.

To insert the selected content in **Text** mode, do one of the following:

- Press **Enter** or **Tab** to insert both the start and end tags and position the cursor inside the start tag in a position suitable for inserting attributes.
- Press **Ctrl + Enter (Command + Enter on OS X)** to insert both the start and end tags and positions the cursor between the tags in a position where you can start typing content.

Note: When the DTD, XML Schema or RELAX NG schema specifies required child elements for the newly added element, they are inserted automatically only if the ***Add Element Content** option* (in the **Content Completion** preferences page) is selected. The *Content Completion Assistant* can also add optional content and first choice particle, as specified in the DTD, XML Schema, or RELAX NG schema. To activate these features, select the ***Add optional content*** and ***Add first Choice particle*** options in the **Content Completion** preferences page.

After inserting an element, the cursor is positioned:

- Before the > character of the start tag, if the element allows attributes, to allow rapid insertion of any of the attributes supported by the element. Pressing the space bar displays the Content Completion list once again. This time it contains the list of allowed attribute names. If the attribute supports a fixed set of parameters, the assistant list displays the list of valid parameters. If the parameter setting is user-defined and therefore variable, the assistant is closed to allow manual insertion. The values of the attributes can be learned from the same elements in the current document
- After the > character of the start tag if the element has no attributes.

Where the Content Completion Assistant is Displayed

The *Content Completion Assistant* is displayed:

- Anywhere within a tag name or at the beginning of a tag name in an XML document, XML Schema, DTD,or Relax NG (full or compact syntax) schema.
- Anywhere within an attribute name or at the beginning of an attribute name in any XML document with an associated schema.
- Within attribute values or at the beginning of attribute values in XML documents where lists of possible values have been defined for that element in the schema associated with the document.

Types of Proposals Listed in the Content Completion Assistant

The following things are considered for determining the proposals that are listed in the content completion window:

- The proposals that populate the *Content Completion Assistant* depend on the element structure specified in the DTD, XML Schema, Relax NG (full or compact syntax) schema, or NVDL schema associated with the edited document.

Note: The *Content Completion Assistant* is able to offer elements defined both by XML Schemas version 1.0 and 1.1.

- The number and type of elements displayed by the *Content Completion Assistant* is dependent on the cursor's current position in the structured document. The child elements displayed within a given element are defined by the structure of the specified DTD, XML Schema, Relax NG (full or compact syntax) schema, or NVDL schema.
- A schema may declare certain attributes as *ID* or *IDREF/IDREFS*. When the document is validated, Oxygen XML Editor checks the uniqueness and correctness of the `ID` attributes. It also collects the attribute values declared in the document to prepare the list of proposals. This is available for documents that use DTD, XML Schema, and Relax NG schema.
- Values of all the `xml:id` attributes are handled as `ID` attributes. They are collected and displayed by the *Content Completion Assistant* as possible values for *anyURI* attributes defined in the schema of the edited document. This works only for XML Schema and Relax NG schemas.
- For documents that use an XML Schema or Relax NG schema, the content assistant offers proposals for attributes and elements values that have an enumeration of tokens as the type. Also, if a default value or fixed value is defined in the XML Schema used in validation for an attribute or element, then that value is offered in the *Content Completion Assistant* window.

Related Information:

Customizing the Content Completion Assistant on page 1241

Set Schema to be Used for Content Completion in Text Mode

The list of proposals in the *Content Completion Assistant* depend on the associated schemas. The DTD, XML Schema, Relax NG, or NVDL schema used to populate the *Content Completion Assistant* is specified in the following methods, in the order of their precedence:

- The *schema specified explicitly in the document*. In this case, Oxygen XML Editor reads the beginning of the document and resolves the location of the DTD, XML Schema, Relax NG schema, or NVDL schema.
- The *default schema declared* in the ***Document Type*** *configuration dialog box* that matches the edited document type.
- For XSLT stylesheets, the schema specified in the Oxygen XML Editor ***XSL*** *preferences page.* Oxygen XML Editor will read the content completion settings when the prolog fails to provide or resolve the location of a DTD, XML Schema, Relax NG, or NVDL schema.
- For XML Schemas, the schema specified in the Oxygen XML Editor ***XSD*** *preferences page.* Oxygen XML Editor will read the content completion settings and the specified schema will enhance the content completion inside the `xs:annotation/xs:appinfo` elements of the XML Schema.

Oxygen XML Editor creates the content completion lists by analysing the detected schema and the current context (the position in the editor). If you change the schema, then the list of tags to be inserted is also updated.

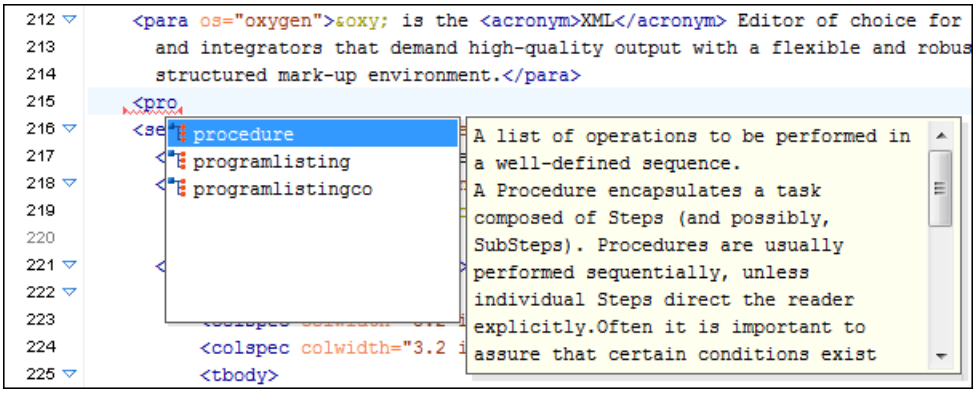


Figure 129: Example: Content Completion Driven by DocBook DTD

Schema Annotations in Text Mode

A schema annotation is a documentation snippet associated with the definition of an element or attribute in a schema. If such a schema is associated with an XML document, the annotations are displayed in:

- The *Content Completion Assistant*.
- A small tooltip window shown when the mouse hovers over an element or attribute. The tooltip window can be invoked at any time by using the **F2** shortcut.

The schema annotations support is available if the schema type is one of the following:

- XML Schema
- Relax NG
- NVDL schema
- DTD

This feature is enabled by default, but you can disable it by deselecting the ***Show annotations in Content Completion Assistant*** option in the **Annotations** preferences page.

Styling Annotations with HTML

You can use HTML format in the annotations you add in an XML Schema or Relax NG schema. This improves the visual appearance and readability of the documentation window displayed when editing XML documents validated against such a schema. An annotation is recognized and displayed as HTML if it contains at least one HTML element (such as `div`, `body`, p, `br`, `table`, `ul`, or `ol`).

The HTML rendering is controlled by the ***Show annotations using HTML format, if possible*** option in the **Annotations** preferences page. When this options is deselected, the annotations are converted and displayed as plain text and if the annotation contains one or more HTML tags (p, `br`, `ul`, `li`), they are rendered as an HTML document loaded in a web browser. For example, p begins a new paragraph, `br` breaks the current line, `ul` encloses a list of items, and `li` encloses an item of the list.

Collecting Annotations from XML Schemas

In an XML Schema, the annotations are specified in an `<xs:annotation>` element like this:

```
<xs:annotation>
  <xs:documentation>
            Description of the element.
  </xs:documentation>
</xs:annotation>
```

If an element or attribute does not have a specific annotation, then Oxygen XML Editor looks for an annotation in the type definition of that element or attribute.

Collecting Annotations from Relax NG Schemas

For Relax NG schema, element and attribute annotations are made using the `<documentation>` element from the `http://relaxng.org/ns/compatibility/annotations/1.0` namespace. However, any element outside the Relax NG namespace (`http://relaxng.org/ns/structure/1.0`) is handled as annotation and the text content is displayed in the annotation window. To activate this behavior, select the ***Use all Relax NG annotations as documentation*** option in the **Annotations** preferences page.

Collecting Annotation from DTDs

For DTD, Oxygen XML Editor defines a custom mechanism for annotations using comments enabled by the ***Prefer DTD comments that start with "doc:" as annotations*** option in the **Annotations** preferences page. The following is an example of a DTD annotation:

```
<!--doc:Description of the element. -->
```

Content Completion Helper Views

Information about the current element being edited is also available in various *dockable* views, such as the ***Model*** *view*, ***Attributes*** *view*, ***Elements*** *view*, and ***Entities*** *view*. By default, they are located on the right-hand side of the main editor window. These views, along with the powerful ***Outline*** *view*, provide spatial and insight information about the edited document and the current element. If any particular view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Model View

The **Model** view presents the structure of the currently selected tag, and its documentation, defined as annotation in the schema of the current document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

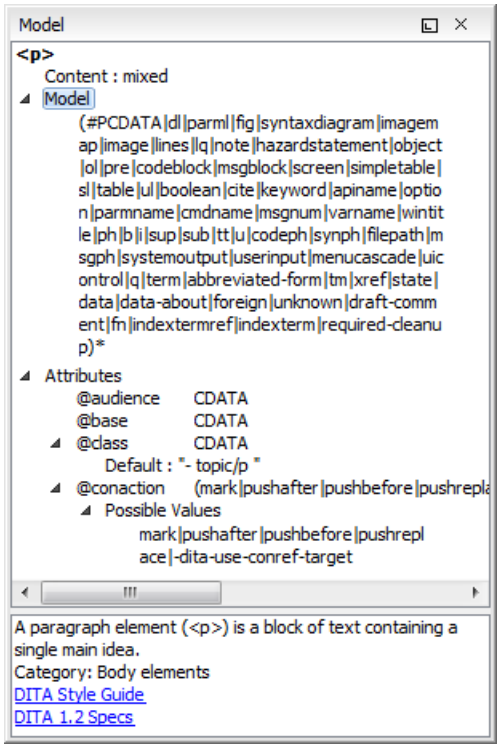


Figure 130: Model View

The **Model** view is comprised of two sections, an element structure panel and an annotations panel.

Element Structure Panel

The element structure panel displays the structure of the currently edited or selected tag in a tree-like format. The information includes the name, model, and attributes of the current tag. The allowed attributes are shown along with imposed restrictions, if any.

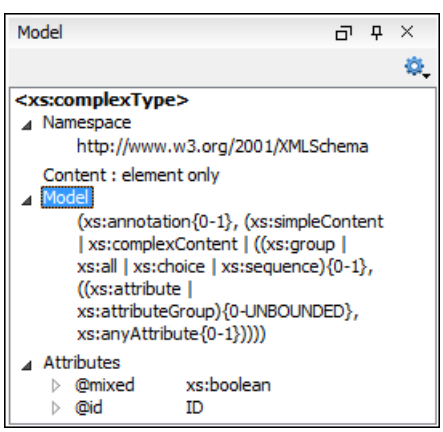


Figure 131: Element Structure Panel

Annotation Panel

The **Annotation** panel displays the annotation information for the currently selected element. This information is collected from the XML schema.

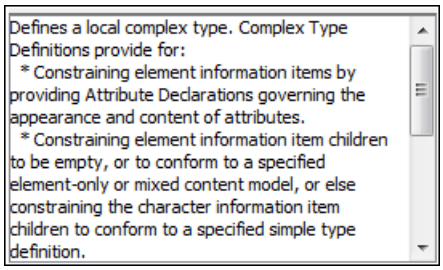


Figure 132: Annotation panel

Attributes View in Text Mode

The **Attributes** view presents all the attributes of the current element determined by the schema of the document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened from the **Window** > **Show View** menu.

You can use the **Attributes** view to insert attributes, edit their values, or add values to existing attributes.

The attributes are rendered differently depending on their state:

- The names of the attributes are rendered with a bold font, and their values with a plain font.
- Default values are rendered with a plain font, painted gray.
- Empty values display the text "[empty]", painted gray.
- Invalid attributes and values are painted red.

To edit the value of the corresponding attribute, double-click a cell in the **Value** column . If the possible values of the attribute are specified as `list` in the schema of the edited document, the **Value** column acts as a combo box that allows you to either select the value from a list or manually enter it.

You can sort the attributes table by clicking the **Attribute** column header. The table contents can be sorted as follows:

- By attribute name in ascending order.
- By attribute name in descending order.
- Custom order, where the used attributes are displayed at the beginning of the table sorted in ascending order, followed by the rest of the allowed elements sorted in ascending order.

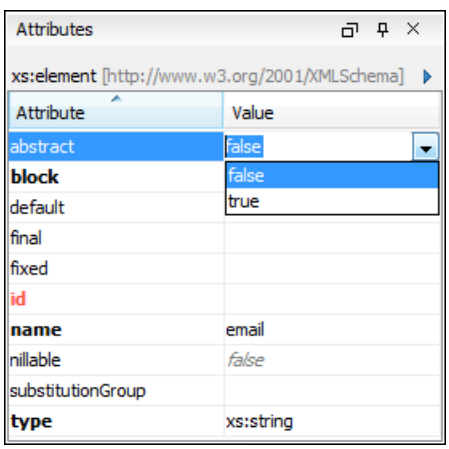


Figure 133: Attributes View

Expand/Collapse Button

There is an **Expand/** **Collapse** button at the top-right of the view. When expanded, this presents the following additional combo boxes:

Name Combo Box

Use this combo box to select an attribute. The drop-down list displays the list of possible attributes allowed by the schema of the document, as in the **Attributes** view. You can use the **Remove** button to delete an attribute and its value from the selected element.

Value Combo Box

Use this combo box to add, edit, or select the value of an attribute. If the selected attribute has predefined values in the schema, the drop-down list displays those possible values. You can use the **Browse** button to select a URL for the value of an attribute. After you have entered or selected a value, use the **Update** button (or press **Enter**) to add the value to the attribute.

Contextual Menu Actions in the Attributes View

The following actions are available in the contextual menu of the **Attributes** view when editing in **Text** mode:

Add

Allows you to insert a new attribute. Adding an attribute that is not in the list of all defined attributes is not possible when the *Allow only insertion of valid elements and attributes* schema-aware option is selected.

Set empty value

Specifies the current attribute value as empty.

Remove

Removes the attribute (action available only if the attribute is specified). You can invoke this action by pressing the **(Delete)** or **(Backspace)** keys.

Copy

Copies the `attrName="attrValue"` pair to the clipboard. The `attrValue` can be:

- The value of the attribute.
- The value of the default attribute, if the attribute does not appear in the edited document.
- Empty, if the attribute does not appear in the edited document and has no default value set.

Paste

Depending on the content of the clipboard, the following cases are possible:

- If the clipboard contains an attribute and its value, both of them are introduced in the **Attributes** view. The attribute is selected and its value is changed if they exist in the **Attributes** view.
- If the clipboard contains an attribute name with an empty value, the attribute is introduced in the **Attributes** view and you can start editing it. The attribute is selected and you can start editing it if it exists in the **Attributes** view.
- If the clipboard only contains text, the value of the selected attribute is modified.

Elements View in Text Mode

The **Elements** view presents a list of all defined elements that are valid at the current cursor position according to the schema associated to the document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Double-clicking any of the listed elements inserts that element into the edited document, at the current cursor position.

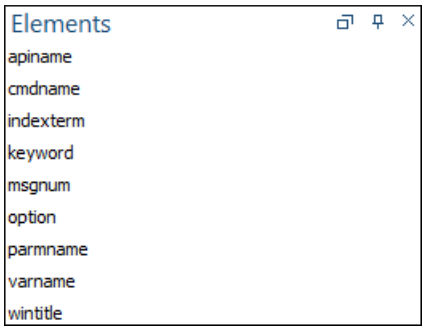


Figure 134: Elements View in Text Mode

Entities View

Entities provide abbreviated entries that can be used in XML files when there is a need of repeatedly inserting certain characters or large blocks of information. An *entity* is defined using the `ENTITY` statement either in the DOCTYPE declaration or in a DTD file associated with the current XML file.

There are three types of entities:

- **Built-in** or **Predefined** - Entities that are part of the predefined XML markup (`<`, `>`, `&`, `'`, `"`).
- **Internal** - Defined in the DOCTYPE declaration header of the current XML.
- **External** - Defined in an external DTD module included in the DTD referenced in the XML DOCTYPE declaration.

Note: If you want to add internal entities, you would need to switch to the Text editing mode and manually modify the DOCTYPE declaration. If you want to add external entities, you need to open the DTD module file and modify it directly.

The **Entities** view displays a list with all entities declared in the current document, as well as built-in ones. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Double-clicking one of the entities will insert it at the current cursor position in the XML document. You can also sort entities by name and value by clicking the column headers.

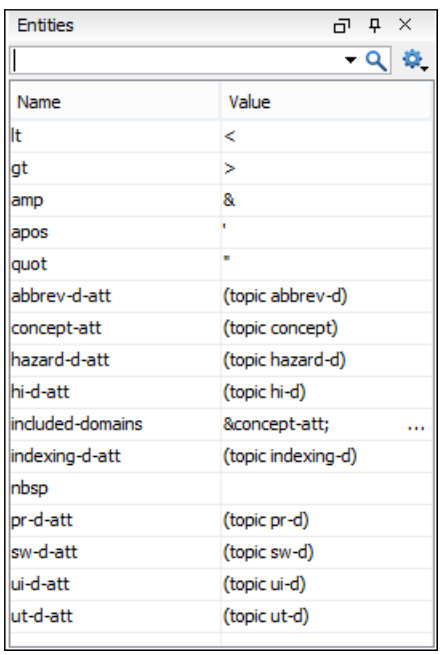


Figure 135: Entities View

The view features a filtering capability that allows you to search an entity by name, value, or both. Also, you can choose to display the internal or external entities.

Note: When entering filters, you can use the *?* and *** wildcards. Also, you can enter multiple filters by separating them with a comma.

Code Templates

Code templates are code fragments that can be inserted quickly at the current editing position . Oxygen XML Editor includes a set of built-in code templates for CSS, LESS, Schematron, XSL, XQuery, and XML Schema document types. You can also *define your own code templates for any type of file and share them with others*.

To get a complete list of available code templates, press **Ctrl + Shift + Space** in **Text** mode. To enter the code template, select it from the list or type its code and press **Enter**. If a shortcut key has been assigned to the code template, you can also use the shortcut key to enter it. Code templates are displayed with a symbol in the content completion list.

When the *Content Completion Assistant* is invoked (**Ctrl + Space (Command + Space on OS X)** in **Text** mode or **Enter** in **Author** mode), it also presents a list of code templates specific to the type of the active editor.

To watch our video demonstration about code templates, go to *https://www.oxygenxml.com/demo/Code_Templates.html*.

Syntax Highlighting in XML Documents

Oxygen XML Editor supports syntax highlighting in XML documents to improve the readability of the content and reduce the time it takes to internalize the semantics of the structured content.

To customize the colors or styles used for the syntax highlighting colors for XML files, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)***.
2. Go to ***Editor*** > ***Syntax Highlight***.
3. Select and expand the **XML** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the **XML** tab in the **Preview** pane to see the effects of your changes.

Tip: Oxygen XML Editor also allows you to specify syntax highlighting colors for specific XML elements and attributes with specific namespace prefixes. This can be done in the ***Editor*** > ***Syntax Highlight*** > ***Elements/Attributes by Prefix*** *preferences page*.

Related Information:

Customize Syntax Highlight colors on page 104

Outline View in Text Mode

The **Outline** view in **Text** mode displays a general tag overview of the currently edited XML Document. When you edit a document, the **Outline** view dynamically follows the changes that you make, displaying the node that you modify. This functionality gives you great insight on the location of your modifications in the current document. It also shows the correct hierarchical dependencies between elements. This makes it easy for you to be aware of the document structure and the way element tags are nested.

Outline View Features

The **Outline** view allows you to:

- Quickly navigate through the document by selecting nodes in the **Outline** tree.
- Insert or delete nodes using contextual menu actions.
- Move elements by dragging them to a new position in the tree structure.
- Highlight elements in the editor area. It is synchronized with the editor area, so when you make a selection in the editor area, the corresponding nodes are highlighted in the **Outline** view, and vice versa.
- View document errors, as they are highlighted in the **Outline** view. A tooltip also provides more information about the nature of the error when you hover over the faulted element.

Outline View Interface

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (*, ?) and separate multiple patterns with commas.

It also includes a **Settings** menu in the top-right corner that presents a variety of options to help you filter the view even further.

Drop and Drop Actions in the Outline View

Entire XML elements can be moved or copied in the edited document using only the mouse in the **Outline** view with drag-and-drop operations. Several drag and drop actions are possible:

- If you drag an XML element in the **Outline** view and drop it on another node, then the dragged element will be moved after the drop target element.
- If you hold the mouse pointer over the drop target for a short time before the drop then the drop target element will be expanded first and the dragged element will be moved inside the drop target element after its opening tag.
- You can also drop an element before or after another element if you hold the mouse pointer towards the upper or lower part of the targeted element. A marker will indicate whether the drop will be performed before or after the target element.
- If you hold down the **(Ctrl (Command on OS X))** key after dragging, a copy operation will be performed instead of a move.

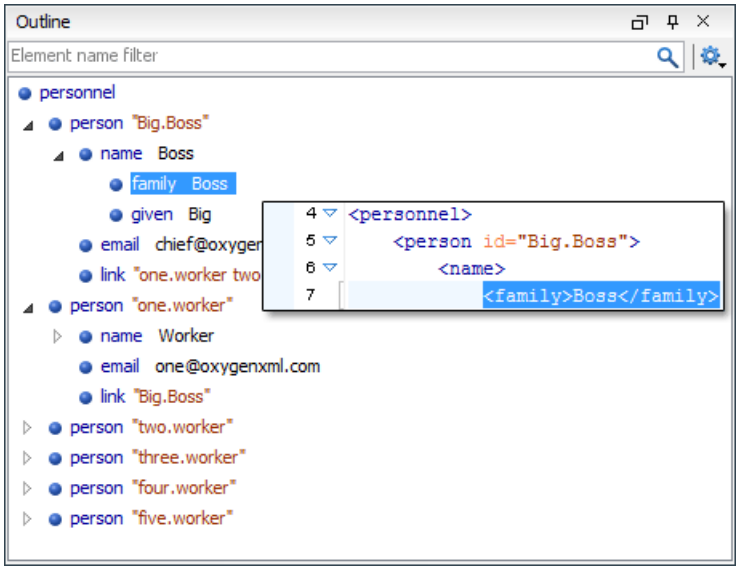


Figure 136: Outline View in Text Mode

Related Information:

Outline View in Author Mode on page 222

Outline View Filters in Text Mode

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (*, ?) and separate multiple patterns with commas.

The following actions are available in the **Settings** menu of the **Outline** view when editing in **Text** mode:

Filter returns exact matches

The text filter of the **Outline** view returns only exact matches.

Selection update on cursor move

Controls the synchronization between **Outline** view and source document. The selection in the **Outline** view can be synchronized with the cursor moves or the changes in the editor. Selecting one of the components from the **Outline** view also selects the corresponding item in the source document.

Flat presentation mode of the filtered results

When active, the application flattens the filtered result elements to a single level.

Show comments and processing instructions

Show/hide comments and processing instructions in the **Outline** view.

Show element name

Show/hide element name.

Show text

Show/hide additional text content for the displayed elements.

Show attributes

Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from *the Outline preferences panel*.

Configure displayed attributes

Displays the ***XML Structured Outline*** *preferences page*.

Outline View Contextual Menu Actions in Text Mode

The following actions are available from the contextual menu in the **Outline** view in **Text** mode:

Append Child

Allows you to select an element (from a drop-down list) that is allowed by the associated schema and inserts it as a child of the current element.

Insert Before

Allows you to select an element (from a drop-down list) that is allowed by the associated schema and inserts it immediately before the current element, as a sibling.

Insert After

Allows you to select an element (from a drop-down list) that is allowed by the associated schema and inserts it immediately after the current element, as a sibling.

Edit Attributes

Opens a dialog box that allows you to edit the attributes of the currently selected component.

Toggle Comment

Encloses the currently selected element in an XML comment, if the element is not already commented. If it is already commented, this action will remove the comment.

Cut

Cuts the currently selected component.

Copy

Copies the currently selected component.

Delete

Deletes the currently selected component.

Expand More

Expands the structure of a component in the **Outline** view.

Collapse All

Collapses the structure of all the component in the **Outline** view.

Inserting or Opening a File at Cursor Location

When editing content in **Text** mode, the following actions (in regards to inserting, opening, or comparing files) are available in the **Document** > **File** menu:

Insert File

Inserts the content of the file with the specified file path into the current document, at the current position of the cursor.

Open File at Cursor

Opens the file at the cursor position in a new panel. If the file path represents a directory path, it will be opened in system file browser. If the file at the specified location does not exist, an error dialog box is displayed and it includes a **Create new file** button that starts the **New document** wizard. This allows you to choose the type or the template for the file. If the action succeeds, the file is created with the referenced location and name and is opened in a new editor panel.

Open File at Cursor in System Application

Opens the file (identified by its link) or web page (identified by a web link) found at cursor position. The target is opened in the default system application associated with that file type.

Compare

Opens the current file in *the **Compare Files** tool*.

Adjusting the Transparency of XML Markup

Most of the time you want the content of a document displayed on screen with zero transparency. However, if you want to focus your attention only on editing text content inside XML markup, Oxygen XML Editor offers the option of reducing the visibility of the markup by increasing their transparency when displayed in **Text** mode. To change the level of transparency, use the **[Tags Transparency Selector]** drop-down menu that is available from the **Source** toolbar. By default, this drop-down menu is not visible. You can add it to the toolbar by using *the **Configure Toolbars** action*. There are several levels of transparency that can be adjusted to make the content more or less visible:

- **Normal Contrast** - Resets the transparency level back to normal.
- **Semi-transparent Text** - Slightly reduces the visibility of text to place greater emphasis on the visibility of the XML markup.
- **Transparent Text** - Greatly reduces the visibility of text to place even greater emphasis on the visibility of the XML markup.
- **Semi-transparent Markup** - Slightly reduces the visibility of the XML markup to place greater emphasis on the visibility of the text.
- **Transparent Markup** - Greatly reduces the visibility of the XML markup to place even greater emphasis on the visibility of the text.

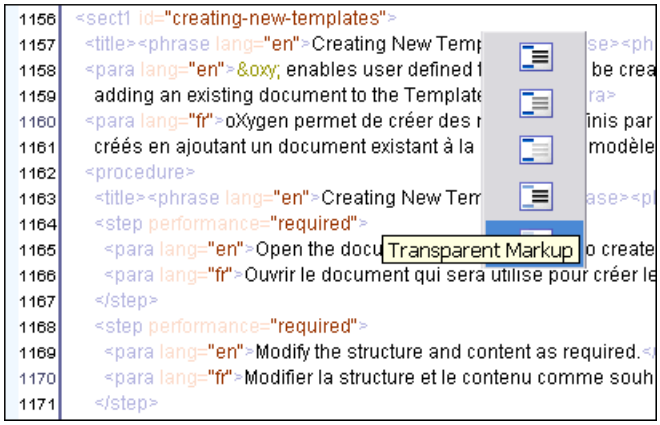


Figure 137: Tags Transparency Selector

Locking and Unlocking XML Markup

For documents with fixed markup, such as forms in which the XML tags are not allowed to be modified (only their text content), the possibility to edit the XML tag names can be toggled on or off with the **Lock / Unlock the XML tags** action available in **Text** editing mode from the **Source** submenu from the contextual menu (or **Document** > **Source** menu).

You can set the default lock state for all opened editors using the ***Lock the XML tags*** *option in the* ***Text*** *preferences page*.

Formatting and Indenting XML Documents

Oxygen XML Editor creates XML documents using several *edit modes*. In ***Text*** *mode*, you as the author decide how the XML file is formatted and indented. In the other modes, and when you switch between modes, Oxygen XML Editor must decide how to format and indent the XML. Oxygen XML Editor will also format and indent your XML for you in **Text** mode if you use one of the Format and Indent options:

- **Document** > **Source** > **Format and Indent** - Formats and indents the whole document.
- **Document** > **Source** > **Indent Selection** - Indents the current selection (but does not add line breaks). This action is also available in the **Source** submenu of the contextual menu.
- **Document** > **Source** > **Format and Indent Element** - Formats and indents the current element (the inmost nested element that currently contains the cursor) and its child-elements. This action is also available in the **Source** submenu of the contextual menu.

A number of settings affect how Oxygen XML Editor formats and indents XML. Many of these settings have to do with how whitespace is handled.

Significant and insignificant whitespace in XML

XML documents are text files that describe complex documents. Some of the white space (spaces, tabs, line feeds, etc.) in the XML document belongs to the document it describes (such as the space between words in a paragraph) and some of it belongs to the XML document (such as a line break between two XML elements). Whitespace belonging to the XML file is called *insignificant whitespace*. The meaning of the XML would be the same if the insignificant whitespace were removed. Whitespace belonging to the document being described is called *significant whitespace*.

Knowing when whitespace is significant or insignificant is not always easy. For instance, a paragraph in an XML document might be laid out like this:

```
<p>NO Free man shall be taken or imprisoned, or be stripped of his Freedom,
or Liberties, or free Customs, or be outlawed, or exiled, or any otherwise
destroyed; nor will we not pass upon him, nor condemn him, but by lawful
judgment of his Peers, or by the <xref
href="http://en.wikipedia.org/wiki/Law_of_the_land" format="html"
scope="external">Law of the land</xref>.
We will sell to no man, we will not deny to any man either Justice or Right.</p>
```

By default, XML considers a single whitespace between words to be significant, and all other whitespace to be insignificant. The paragraph above could have been written on one line because the XML parser would see it as exactly the same paragraph since all multiple consecutive whitespaces will be replaced with a single whitespace. Removing the insignificant space in markup like this is called *normalizing space*.

In some cases, all the spaces inside an element should be treated as significant. For example, in a code sample:

```
<codeblock>
class HelloWorld
{
   public static void main(String args[])
   {
      System.out.println("Hello World");
   }
}
</codeblock>
```

Here every whitespace character between the `codeblock` tags should be treated as significant.

How Oxygen XML Editor determines when whitespace is significant

When Oxygen XML Editor formats and indents an XML document, it introduces or removes insignificant whitespace to produce a layout with reasonable line lengths and elements indented to show their place in the hierarchy of the document. To correctly format and indent the XML source, Oxygen XML Editor needs to know when to treat whitespace as significant and when to treat it as insignificant. However it is not always possible to tell this from the XML source file alone. To determine what whitespace is significant, Oxygen XML Editor assigns each element in the document to one of four categories:

Ignore space

In the ignore space category, all whitespace is considered insignificant. This generally applies to content that consists only of elements nested inside other elements, with no text content.

Normalize space

In the normalize space category, a single whitespace character between character strings is considered significant and all other spaces are considered insignificant. Therefore, all consecutive whitespaces will be replaced with a single space. This generally applies to elements that contain text content only.

Mixed content

In the mixed content category, a single whitespace between text characters is considered significant and all other spaces are considered insignificant. However,

- Whitespace between two child elements embedded in the text is normalized to a single space (rather than to zero spaces as would normally be the case for a text node with only whitespace characters, or the space between elements generally).
- The lack of whitespace between a child element embedded in the text and either adjacent text or another child element is considered significant. That is, no whitespace can be introduced here when formatting and indenting the file.

For example:

```
<p>The file is located in <i>HOME</i>/<i>USER</i>/hello.
     This is a <strong>big</strong>

<emphasis>deal</emphasis>.
</p>
```

In this example, whitespace should not be introduced around the i tags as it would introduce extra significant whitespace into the document. The space between the end `</strong>` tag and the beginning `<emphasis>` tag should be normalized to a single space, not zero spaces.

Preserve space

In the preserve space category, all whitespace in the element is regarded as significant. No changes are made to the spaces in elements in this category. However, child elements may be in another category, and may be treated differently.

Attribute values are always in the preserve space category. The spaces between attributes in an element tag are always in the default space category.

Oxygen XML Editor consults several pieces of information to assign an element to one of these categories. An element is always assigned to the most restrictive category (from Ignore to Preserve) that it is assigned to by any of the sources Oxygen XML Editor consults. For instance, if the element is named on the **Default elements** list (as described below) but it has an `xml:space="preserve"` attribute in the source file, it will be assigned to the preserve space category. If an element has the `xml:space="default"` attribute in the source, but is listed on the **Mixed content** elements list, it will be assigned to the mixed content category.

To assign elements to these categories, Oxygen XML Editor consults information from the following sources:

`xml:space`

If the XML element contains the `xml:space` attribute, the element is promoted to the appropriate category based on the value of the attribute.

CSS whitespace property

If the CSS stylesheet controlling the **Author** mode editor applies the `whitespace: pre` setting to an element, it is promoted to the preserve space category.

CSS display property

If a text node contains only white-spaces:

- If the node has a parent element with the CSS `display` property set to `inline` then the node is promoted to the mixed content category.
- If the left or right sibling is an element with the CSS `display` property set to `inline` then the node is promoted to the mixed content category.
- If one of its ancestors is an element with the CSS `display` property set to `table` then the node is assigned to the ignore space category.

Schema aware formatting

If a schema is available for the XML document, Oxygen XML Editor can use information from the schema to promote the element to the appropriate category. For example:

- If the schema declares an element to be of type `xs:string`, the element will be promoted to the preserve space category because the string built-in type has the whitespace facet with the value preserve.
- If the schema declares an element to be mixed content, it will be promoted to the mixed content category.

Schema aware formatting can be turned on and off.

- To turn it on or off for **Author** mode, *open the **Preferences** dialog box **(Options** > **Preferences)***, go to **Editor** > **Edit modes** > **Author** > **Schema aware**, and select/deselect the ***Schema aware normalization, format and indent** option*.
- To turn it on or off for the **Text** editing mode ,*open the **Preferences** dialog box **(Options** > **Preferences)***, go to **Editor** > **Format** > **XML**, and select/deselect the ***Schema aware format and indent** option*.

Preserve space elements list

If an element is listed in the ***Preserve space** tab of the **Element Spacing** list* in the *XML formatting preferences*, it is promoted to the preserve space category.

Default space elements list

If an element is listed in the ***Default space** tab of the **Element Spacing** list* in the *XML formatting preferences*, it is promoted to the default space category

Mixed content elements list

If an element is listed in the ***Mixed content** tab of the **Element Spacing** list* in the *XML formatting preferences*, it is promoted to the mixed content category.

Element content

If an element contains mixed content, that is, a mix of text and other elements, it is promoted to the mixed content category. (Note that, in accordance with these rules, this happens even if the schema declares the element to have element only content.)

If an element contains text content, it is promoted to the default space category.

Text node content

If a text node contains any non-whitespace characters then the text node is promoted to the normalize space category.

Exception to the Rule

In general, a element can only be promoted to a more restrictive category (one that treats more whitespace as significant). However, there is one exception. In **Author** mode, if an element is marked as mixed content in the schema, but the actual element contains no text content, it can be demoted to the space ignore category if all of its child elements are displayed as *blocks* by the associated CSS (that is, they have a CSS property of `display: block`). For example, in some schemas, a section or a table entry can be defined as having mixed content but in many cases they contain only *block elements*. In these cases, any whitespace they contain cannot be significant

and they can be treated as space ignore elements. This exception can be turned on or off using the ***Schema Aware Editing*** *option* in the **Schema-Aware** preferences page.

How Oxygen XML Editor formats and indents XML

You can control how Oxygen XML Editor formats and indents XML documents. This can be particularly important if you store your XML document in a version control system, as it allows you to limit the number of trivial changes in spacing between versions of an XML document. The following preference pages include options that control how XML documents are formatted:

- ***Format*** *preferences page*
- ***XML Formatting*** *preferences page*
- ***Whitespaces*** *preferences page*

When Oxygen XML Editor formats and indents XML

Oxygen XML Editor formats and indents a document, or part of it, on the following occasions:

- In **Text** mode when you select one of the format and indent actions (**Document** > **Source** > **Format and Indent**, **Document** > **Source** > **Indent Selection**, or **Document** > **Source** > **Format and Indent Element**).
- When saving documents in **Author** mode.
- When switching from **Author** mode to another mode.
- When saving documents in **Design** mode.
- When switching from **Design** mode to another mode.
- When saving or switching to **Text** mode from **Grid** mode, if the ***Format and indent when passing from grid to text or on save*** *option* is selected in the **Grid** preferences page.

Setting an Indent Size to Zero

Oxygen XML Editor will automatically *format and indent* documents at certain times. This includes indenting the content from the margin to reflect its structure. In some cases, you may not want your content indented. To avoid your content being indented, you can set an indent size of zero.

Note: Changing the indent size does not override the rules that Oxygen XML Editor uses for handling whitespace when formatting and indenting XML documents. Therefore, changing the indent size will have no effect on elements that require whitespaces to be maintained.

There are two cases to consider.

Maintaining zero indent in documents with zero indent

If you have existing documents with zero indent and you want Oxygen XML Editor to maintain a zero indent when editing or formatting those documents:

1. *Open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to ***Editor*** > ***Format***.
2. Select **Detect indent on open**.
3. Select **Use zero-indent if detected**.

Oxygen XML Editor will examine the indent of each document as it is opened and if the indent is zero for all lines, or for nearly all lines, a zero indent will be used when formatting and indenting the document. Otherwise, Oxygen XML Editor will use the indent closest to what it detects in the document.

Enforcing zero indent for all documents

If you want all documents to be formatted with zero indent, regardless of their current indenting:

1. *Open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to ***Editor*** > ***Format***.
2. Deselect **Detect indent on open**.
3. Set **Indent size** to `0`.

All documents will be formatted and indented with an indent of zero.

Warning: Setting the indent size to zero can change the meaning of some file types, such as Python source files.

Format and Indent (Pretty-Print) Multiple Files

Oxygen XML Editor provides support for formatting and indenting (*pretty-print*) multiple files at once. This action is available for any document in XML format, as well as for XQuery, CSS, JavaScript, and JSON documents.

To format and indent multiple files, use the **Format and Indent Files** action that is available in the contextual menu of the ***Project*** *view* or from the **Tools** menu. This opens the **Format and Indent Files** dialog box that allows you to configure options for the action.

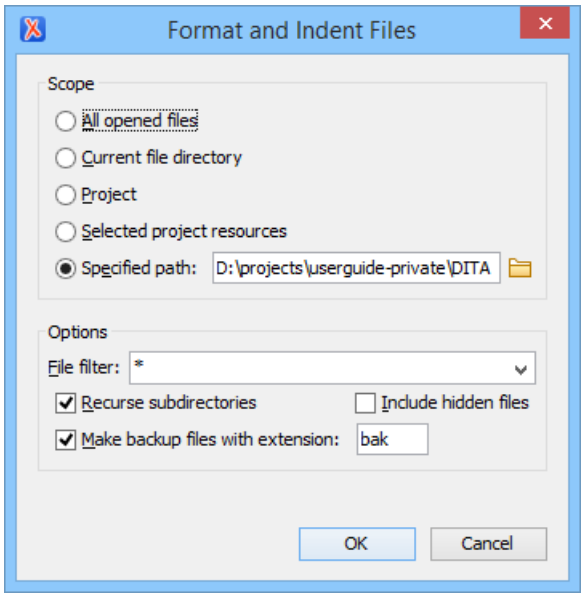


Figure 138: Format and Indent Files Dialog Box

The **Scope** section allows you choose from the following scopes:

- **All opened files** - The *pretty-print* is performed in all opened files.
- **Directory of the current file** - All the files in the folder of the current edited file.
- **Project files** - All files from the current project.
- **Selected project files** - The selected files from the current project.
- **Specified path** - the *pretty-print* is performed in the files located at a specified path.

The **Options** section includes the following options:

- **File filter** - Allow you to filter the files from the selected scope.
- **Recurse subdirectories** - When selected, the *pretty-print* is performed recursively for the specified scope. The one exception is that this option is ignored if the scope is set to **All opened files**.
- **Include hidden files** - When selected, the *pretty-print* is also performed in the hidden files.
- **Make backup files with extension** - When selected, Oxygen XML Editor makes backup files of the modified files. The default extension is `.bak`, but you can change the extension as you prefer.

Managing Highlighted Content

While working with XML documents you often have frequent changes to the structure and content. You are often faced with a situation where you need to make a slight change in multiple places in the same document. Oxygen XML Editor includes a feature, **Manage Highlighted Content**, that is designed to help you achieve this.

When you are in **Text** mode and you perform a search operation or apply an XPath that highlights multiple results, you can select the **Manage Highlighted Content** action from the contextual menu of any highlight in the document, and the following options are available in its submenu:

- **Modify All** - Use this option to modify (in-place) all the occurrences of the selected content. When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Note: If you select a very large number of highlights that you want to modify using this feature, a dialog box informs you that you may experience performance issues. You have the option to either use the ***Find/Replace*** *operation*, or continue the operation.

- **Surround All** - Use this option to surround the highlighted content with a specific tag. This option opens the **Tag** dialog box. The **Specify the tag** drop-down menu presents all the available elements that you can choose from.
- **Remove All** - Removes all the highlighted content.

If you right-click content in another part of the document, other than a highlight, you have the option to select the following option:

- **Modify All Matches** - Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment in which the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Quick Assist Support for IDs and IDREFS

The *Quick Assist support* is activated automatically when you place the cursor inside an ID or IDREF in **Text** mode. To access it, click the yellow bulb help marker placed on the current line, in the line number stripe of the editor. You can also invoke the *Quick Assist* menu from the contextual menu or by pressing **Alt 1 (Meta Alt 1 on Mac OS X)** on your keyboard.

The following actions are available:

Rename in

Renames the ID and all its occurrences. Selecting this action opens the **Rename XML ID** dialog box. This dialog box lets you insert the new ID value and *choose the scope of the rename operation*. For a preview of the changes you are about to make, click **Preview**. This opens the **Preview** dialog box, which presents a list with the files that contain changes and a preview zone of these changes.

Search Declarations

Searches for the declaration of the ID reference. By default, the scope of this action is the current project. If you configure a scope using the ***Select the scope for the Search and Refactor operations*** dialog box, this scope will be used instead.

Search References

Searches for the references of the ID. By default, the scope of this action is the current project. If you configure a scope using the ***Select the scope for the Search and Refactor operations*** dialog box, this scope will be used instead.

Change scope

Opens the ***Select the scope for the Search and Refactor operations*** dialog box.

Rename in File

Renames the ID you are editing and all its occurrences from the current file.

Search Occurrences

Searches for the declaration an references of the ID located at the cursor position in the current document.

Related Information:

Working with Modular XML Files in the Master Files Context on page 481

Highlight ID Occurrences in Text Mode

To see the occurrences of an ID in an XML document in the **Text** mode, place the cursor inside the ID declaration or reference. The occurrences are marked in the vertical side bar at the right of the editor. Click a marker on the side bar to jump to the occurrence that it corresponds to. The occurrences are also highlighted in the editing area.

Note: Highlighted ID declarations are rendered with a different color than highlighted ID references. To customize these colors or disable this feature, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to ***Editor** > **Mark Occurrences***.

Related Information:

Working with Modular XML Files in the Master Files Context

Contextual Menu Actions in Text Mode

When editing XML documents in **Text** mode, Oxygen XML Editor provides the following actions in the contextual menu (many of them also appear in the submenus of the **Document** menu):

Cut, Copy, Paste

Executes the typical editing actions on the currently selected content.

Copy XPath

Copies the XPath expression of the current element or attribute from the current editor to the clipboard.

Toggle Comment (Ctrl + Shift + Comma (Command + Shift + M on OS X))

Comments the current selection of the current editor. If the selection already contains a comment the action removes the comment from around the selection. If there is no selection in the current editor and the cursor is not positioned inside a comment the current line is commented. If the cursor is positioned inside a comment then the commented text is uncommented.

Go to submenu

This submenu includes the following actions:

Go to Matching Tag (Ctrl + Shift + G)

Moves the cursor to the end tag that matches the start tag, or vice versa.

Go after Next Tag (Ctrl + CloseBracket (Command + CloseBracket on OS X))

Moves the cursor to the end of the next tag.

Go after Previous Tag (Ctrl + OpenBracket (Command + OpenBracket on OS X))

Moves the cursor to the end of the previous tag.

Select submenu

This submenu allows you to select the following:

Element

Selects the entire element at the current cursor position.

Content

Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

Attributes

Selects all the attributes of the element at the current cursor position.

Parent

Selects the parent element at the current cursor position.

Source submenu

This submenu includes the following actions:

Shift Right (Tab)

Shifts the currently selected block to the right.

Shift Left (Shift + Tab)

Shifts the currently selected block to the left.

Indent selection (Ctrl + I (Command + I on OS X))

Corrects the indentation of the selected block of lines if it does not follow the current *indenting preferences*.

Escape Selection

Escapes a range of characters by replacing them with the corresponding character entities.

Unescape Selection

Replaces the character entities with the corresponding characters.

Format and Indent Element (Ctrl + Shift + I (Command + Shift + I on OS X))

Pretty-prints the element that surrounds the current cursor position.

To Upper Case

Converts the content selection to upper case characters.

To Lower Case

Converts the content selection to lower case characters.

Capitalize Lines

It capitalizes the first letter found on every new line that is selected. Only the first letter is affected, the rest of the line remains the same. If the first character on the new line is not a letter then no changes are made.

Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on OS X))

Converts a sequence of hexadecimal characters to the corresponding Unicode character. The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed of 2 to 4 hexadecimal characters and may or may not be preceded by the `0x` or `0X` prefix. Examples of valid sequences: `0x0045`, `0X0125`, `1253`, `265`, `43`.

Base64 Encode/Decode submenu

This submenu include the following actions for encoding or decoding **base64** schemes:

Import File to Encode and Insert

Encodes a file and then inserts the encoded content into the current document at the cursor position.

Decode Selection and Export to File

Decodes a selection of text from the current document and then exports (saves) the result to another file.

Encode Selection

Replaces a selection of text with the result of encoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the ***Encoding for Base64, Base32, Hex conversions*** *option in the* ***Encoding*** *preferences page* will be used. Likewise, the same is true if the ***Show the dialog box for choosing the encoding for Base64, Base 32, Hex conversions*** *option is not selected in the* ***Messages*** *preference page*.

Decode Selection

Replaces a selection of text with the result of decoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the ***Encoding for Base64, Base32, Hex conversions*** *option in the* ***Encoding*** *preferences page* will be used. Likewise, the same is true if the ***Show the dialog box for choosing the encoding for Base64, Base 32, Hex conversions*** *option is not selected in the* ***Messages*** *preference page*.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box

is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Base32 Encode/Decode submenu

This submenu include the following actions for encoding or decoding **base32** schemes:

Import File to Encode and Insert

Encodes a file and then inserts the encoded content into the current document at the cursor position.

Decode Selection and Export to File

Decodes a selection of text from the current document and then exports (saves) the result to another file.

Encode Selection

Replaces a selection of text with the result of encoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the ***Encoding for Base64, Base32, Hex conversions*** *option in the* ***Encoding*** *preferences page* will be used. Likewise, the same is true if the ***Show the dialog box for choosing the encoding for Base64, Base 32, Hex conversions*** *option is not selected in the* ***Messages*** *preference page*.

Decode Selection

Replaces a selection of text with the result of decoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the ***Encoding for Base64, Base32, Hex conversions*** *option in the* ***Encoding*** *preferences page* will be used. Likewise, the same is true if the ***Show the dialog box for choosing the encoding for Base64, Base 32, Hex conversions*** *option is not selected in the* ***Messages*** *preference page*.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Hex Encode/Decode submenu

This submenu include the following actions for encoding or decoding **hex** schemes:

Import File to Encode and Insert

Encodes a file and then inserts the encoded content into the current document at the cursor position.

Decode Selection and Export to File

Decodes a selection of text from the current document and then exports (saves) the result to another file.

Encode Selection

Replaces a selection of text with the result of encoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the ***Encoding for Base64, Base32, Hex conversions*** *option in the* ***Encoding*** *preferences page* will be used. Likewise, the same is true if the ***Show the dialog box for choosing the encoding for Base64, Base 32, Hex conversions*** *option is not selected in the* ***Messages*** *preference page*.

Decode Selection

Replaces a selection of text with the result of decoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the ***Encoding for Base64, Base32, Hex conversions*** *option in the* ***Encoding*** *preferences page* will be used. Likewise, the same is true if

the ***Show the dialog box for choosing the encoding for Base64, Base 32, Hex conversions*** *option is not selected in the* ***Messages*** *preference page*.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Join and Normalize Lines (Ctrl + J (Command + J on OS X))

For the current selection, this action joins the lines by replacing the *line separator* with a single space character. It also normalizes the whitespaces by replacing a sequence of such characters with a single space.

Insert new line after (Ctrl + Alt + Enter (Command + Alt + Enter on OS X))

This action has the same result as moving the cursor to the end of the current line and pressing the *ENTER* key.

Insert XInclude

Displays a dialog box that allows you to browse and select the content to be included and automatically generates the corresponding XInclude instruction.

Note: In the **Author** mode, this dialog box presents a preview of the inserted document as an author page in the **Preview** tab and as a text page in the **Source** tab. In the **Text** mode, the **Source** tab is presented.

Import entities list

Displays a dialog box that allows you to select a list of files as sources for external DTD entities. The internal subset of the DOCTYPE declaration of your document will be updated with the chosen entities. For instance, choosing the files `chapter1.xml` and `chapter2.xml` inserts the following section in the DOCTYPE:

```
<!ENTITY chapter1 SYSTEM "chapter1.xml">
<!ENTITY chapter2 SYSTEM "chapter2.xml">
```

Lock / Unlock the XML Tags

Disables or enables the ability to edit XML tags.

Canonicalize

Opens the **Canonicalize** dialog box that allows you to select a *canonicalization* algorithm to standardize the format of the document.

Sign

Opens the **Sign** dialog box that allows you to configure a digital signature for the document.

Verify Signature

Allows you to specify the location of a file to verify its digital signature.

Manage Highlighted Content submenu

This submenu is available from the contextual menu when it is invoked from a highlight after you perform a search operation or apply an XPath expression that highlights more than one result. The following options are available in this submenu:

Modify All

Allows you to modify (in-place) all the occurrences of the selected content. A thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Surround All

Surround the highlighted content with a specific tag. This option opens the **Tag** dialog box. The **Specify the tag** drop-down menu presents all the available elements that you can choose from.

Remove All

Removes all the highlighted content.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Show Definition (Ctrl + Shift + Enter)

Moves the cursor to the definition of the current element or attribute in the schema (DTD, XML Schema, Relax NG schema) associated with the edited XML document. If the current attribute is a "type" belonging to the "*http://www.w3.org/2001/XMLSchema-instance*" namespace, the cursor is moved in the XML schema to the definition of the type referenced in the value of the attribute.

Refactoring submenu

This submenu includes the following actions:

Rename Element

The element from the cursor position, and any elements with the same name, can be renamed according with the options from the **Rename** dialog box.

Rename Prefix (Alt + Shift + P (Command + Shift + P on OS X))

The prefix of the element from the cursor position, and any elements with the same prefix, can be renamed according with the options from the **Rename** dialog box.

- If you select the **Rename current element prefix** option, the application will recursively traverse the current element and all its children. *For example*, to change the `xmlns:p1="ns1"` association in the current element to `xmlns:p5="ns1"`, if the `xmlns:p1="ns1"` association is applied on the parent element, then Oxygen XML Editor will introduce `xmlns:p5="ns1"` as a new declaration in the current element and will change the prefix from p1 to p5. If p5 is already associated with another namespace in the current element, then the conflict will be displayed in a dialog box. By pressing **OK**, the prefix is modified from p1 to p5 without inserting a new declaration.
- If you select the **Rename current prefix in all document** option, the application will apply the change on the entire document.
- To also apply the action inside attribute values, select the **Rename also attribute values that start with the same prefix** checkbox.

Surround with submenu

Presents a drop-down menu that allows you to choose a tag to surround a selected portion of content.

Surround with Tags (Ctrl + E (Command + E on OS X))

Allows you to choose a tag that encloses a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.

- If the ***Position cursor between tags*** *option* is selected in the **Content Completion** preferences page, the cursor is placed between the start and end tag.
- If the ***Position cursor between tags*** *option* is not selected in the **Content Completion** preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

Surround with '[tag]' (Ctrl + ForwardSlash (Command + ForwardSlash on OS X))

Surround the selected content with the last tag used.

Delete element tags (Alt + Shift + X (Command + Alt + X on OS X))

Deletes the start and end tag of the current element.

Split element (Alt + Shift + D (Ctrl + Alt + D on OS X))

Split the element from the cursor position into two identical elements. The cursor must be inside the element.

Join elements (Alt + Shift + J (Command + Alt + J on OS X))

Joins the left and right elements relative to the current cursor position. The elements must have the same name, attributes, and attributes values.

Attributes submenu

Contains predefined XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

Add/Change attribute

Allows you to change the value of an attribute or insert a new one.

Delete attribute

Allows you to remove one or more attributes.

Rename attribute

Allows you to rename an attribute.

Replace in attribute value

Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

Comments submenu

Contains predefined XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

Delete comments

Allows you to delete comments found inside one or more elements.

DITA submenu

Contains predefined XML refactoring operations that pertain to DITA documents with some of the information preconfigured based upon the current context.

Change topic ID to file name

Use this operation to change the ID of a topic to be the same as its file name.

Convert conrefs to conkeyrefs

Use this operation to convert `conref` attributes to `conkeyref` attributes.

Convert simple tables to CALS tables

Use this operation to convert DITA simple tables to CALS tables.

Convert to Concept

Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept).

Convert to Reference

Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference).

Convert to Task

Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task).

Convert to Topic

Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic).

Convert to Troubleshooting

Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting).

Elements submenu

Contains predefined XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

Delete element

Allows you to delete elements.

Delete element content

Allows you to delete the content of elements.

Insert element

Allows you to insert new elements.

Rename element

Allows you to rename elements.

Unwrap element

Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

Wrap element

Allows you to surround elements with element tags.

Wrap element content

Allows you to surround the content of elements with element tags.

Fragments submenu

Contains predefined XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

Insert XML fragment

Allows you to insert an XML fragment.

Replace element content with XML fragment

Allows you to replace the content of elements with an XML fragment.

Replace element with XML fragment

Allows you to replace elements with an XML fragment.

JATSKit submenu

Contains predefined XML refactoring operations that pertain to JATS documents with some of the information preconfigured based upon the current context.

Add BITS DOCTYPE - NLM/NCBI Book Interchange 2.0

Adds an NLM 'BITS' 2.0 DOCTYPE declaration.

Add Blue DOCTYPE - NISO JATS Publishing 1.1

Adds a JATS 'Blue' 1.1 DOCTYPE declaration.

Normalize IDs

Assigned IDs are normalized and IDs are assigned to some elements that are missing them.

Manage IDs submenu

This submenu is available for XML documents that have an associated DTD, XML Schema, or Relax NG schema. It includes the following actions:

Rename in

Renames the ID and all its occurrences. Selecting this action opens the **Rename XML ID** dialog box. This dialog box lets you insert the new ID value and *choose the scope of the rename operation*. For a preview of the changes you are about to make, click **Preview**. This opens the **Preview** dialog box, which presents a list with the files that contain changes and a preview zone of these changes.

Rename in File

Renames the ID you are editing and all its occurrences from the current file.

Search References

Searches for the references of the ID. By default, the scope of this action is the current project. If you configure a scope using the ***Select the scope for the Search and Refactor operations*** dialog box, this scope will be used instead.

Search References in

Searches for the references of the ID. Selecting this action opens the ***Select the scope for the Search and Refactor operations***.

Search Declarations

Searches for the declaration of the ID reference. By default, the scope of this action is the current project. If you configure a scope using the ***Select the scope for the Search and Refactor operations*** dialog box, this scope will be used instead.

Search Declarations in

Searches for the declaration of the ID reference. Selecting this action opens the ***Select the scope for the Search and Refactor operations***.

Search Occurrences in file

Searches for the declaration an references of the ID in the current document.

Quick Assist (Alt + 1 (Command + Alt + 1 on OS X))

Available when the cursor is inside an ID or IDREF, this action opens the *Quick Assist* window that allows you to select some search and refactoring actions for the selected ID or IDREF.

Open submenu

The following actions are available in this submenu:

Open File at Cursor

Opens the file at the cursor position in a new panel. If the file path represents a directory path, it will be opened in system file browser. If the file at the specified location does not exist, an error dialog box is displayed and it includes a **Create new file** button that starts the **New document** wizard. This allows you to choose the type or the template for the file. If the action succeeds, the file is created with the referenced location and name and is opened in a new editor panel.

Open File at Cursor in System Application

Opens the file (identified by its link) or web page (identified by a web link) found at cursor position. The target is opened in the default system application associated with that file type.

Compare

Opens the current file in *the **Compare Files** tool*.

Resource Hierarchy

Opens the ***Resource Hierarchy/Dependencies*** *view* that allows you to see the resource hierarchy for an XML document.

Resource Dependencies

Opens the ***Resource Hierarchy/Dependencies*** *view* that allows you to see the resource dependencies for an XML document.

Editing XML Documents in Grid Mode

The **Grid** mode in Oxygen XML Editor displays the XML document as a structured grid of nested tables where the text content can be modified without directly interacting with the XML markup. This is helpful for non-technical users who want to edit text content without modifying the XML markup. You can easily expand or collapse elements within the table and the document structure can be changed with simple drag/drop or copy/paste operations. A useful *Content Completion Assistant* is also available in **Grid** mode.

To switch to this mode, select **Grid** at the bottom of the editing area.

To watch our video demonstration about some of the features available in the **Grid** editor, go to *https://www.oxygenxml.com/demo/Grid_Editor.html*.

Editing Actions in Grid Mode

A variety of editing actions are available in **Grid** mode from the contextual menu, the **Document** menu, the toolbar, and with shortcut keys. This section explains some of those useful editing actions.

Sorting a Table Column

You can sort certain table columns by using the **Sort Ascending** or **Sort Descending** actions that are available on the toolbar or from the contextual menu.

The sorting result depends on the data type of the column content. It could be a numerical sorting for numbers or an alphabetical sorting for text information. The editor automatically analyzes the content and decides what type of sorting to apply. When a mixed set of values is present in the sorted column, a dialog box is displayed that allows you to choose the desired type of sorting between *numerical* and *alphabetical*.

Inserting a Row in a Table

You can add a new row using the **Copy**/**Paste** actions, or by selecting **Insert row** from the contextual menu or the toolbar.

For a faster way to insert a new row, move the selection over the row header, and then press **Enter**. The row header is the zone in the left of the row that holds the row number. The new row is inserted below the selection.

Inserting a Column in a Table

You can insert a column after the selected column by using the **Insert column** action from the contextual menu or the toolbar.

Clearing the Content of a Column

You can clear all the cells from a column by using the **Clear content** action from the contextual menu.

Adding Nodes

You can add nodes before, after, or as a child of the currently selected node by using the various actions in the following submenus of the contextual menu:

- **Insert before** - Offers a list of valid nodes, depending on the context, and inserts your selection before the currently selected node, as a sibling.
- **Insert after** - Offers a list of valid nodes, depending on the context, and inserts your selection after the currently selected node, as a sibling.
- **Append child** - Offers a list of valid nodes, depending on the context, and appends your selection as a child of the currently selected node.

Duplicating Nodes

You can quickly create new nodes by duplicating existing ones. The **Duplicate** action is available in the contextual menu and in the **Document** > **Grid Edit** menu.

Refresh Layout

When using drag and drop to reorganize the document, the resulting layout can be different from the expected one. For instance, the layout can contain a set of sibling tables that can be joined together. To force the layout to be recomputed, you can use the **Refresh selected** action that is available in the contextual menu and in the **Document** > **Grid Edit** menu.

Editing a Cell Value

To edit the value of a cell, simply select the grid cell and press **(Enter)** or you can use the **Start Editing** action found in the **Document** > **Grid Edit** menu.

To stop editing a cell value, press **(Enter)** again or use the **End Editing** action found in the **Document** > **Grid Edit** menu.

To cancel the editing without saving the current changes in the document, press the **(Esc)** key.

Drag and Drop in the Grid Editing Mode

You can easily arrange sections in your XML document in the **Grid** mode by using drag and drop actions.

You can do the following with drag and drop:

- Copy or move a set of nodes.

- Change the order of columns in the tables.
- Move the rows from the tables.

These operations are available for both single and multiple selections. To deselect one of the selected fragments, use **Ctrl + Single-Click (Command + Single-Click on OS X)**.

While dragging, the editor paints guide-lines showing the locations where you can drop the nodes. You can also drag nodes outside the **Grid** editor and text from other applications into the **Grid**. For more information, see *Copy and Paste in the Grid Editor*.

Copy and Paste in the Grid Editing Mode

The selection in the **Grid** mode is a bit complex compared to the selection in a text component. It consists of a currently selected cell and additional selected cells. These additional cells are either selected with the cursor, or implied by the currently selected cell. To be more specific, consider that you click the name of the column (this becomes the current selected cell), but the editor automatically extends the selection so that it contains all the cells from that column. The currently selected cell is painted with a color that is different from the rest of the selection.

You can also select discontinuous regions of nodes and place them in the clipboard with the copy action. To deselect one of the selected fragments, use **Ctrl + Single-Click (Command + Single-Click on OS X)**.

Pasting Content Within Grid Mode

You can paste the copied nodes relative to the currently selected cell using one of the following actions (available in the contextual menu):

- **Paste (Ctrl + V (Command + V on OS X))** - Pastes the content, as a sibling, just below (after) the current selection.
- **Paste as Child** - Pastes the content as the last child of the current selection.

Pasting Content from Grid Mode to Other Edtiors

Nodes that are copied from the **Grid** editor can also be pasted into the **Text** editor or other applications. When copying from the **Grid** into the **Text** editor or other text based applications, the inserted string represents the nodes serialization. The nodes from tables can be copied using HTML or RTF in table format. The resulting cells contain only the concatenated values of the text nodes.

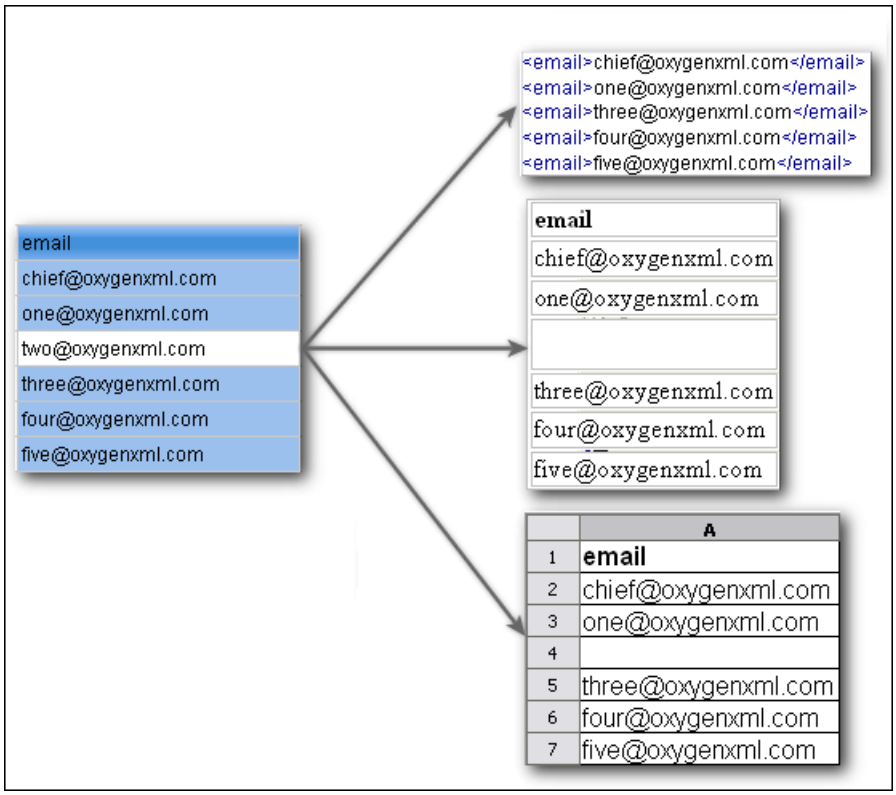


Figure 139: Copying from Grid to Other Editors

Pasting Content from Other Editors into Grid Mode

You can also paste well-formed XML content or tab separated values from other editors into the **Grid** editor. If you paste XML content, the result will be the insertion of the nodes obtained by parsing this content.

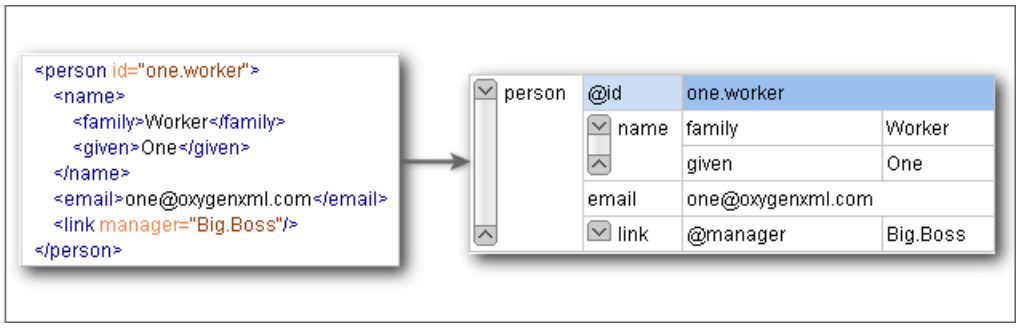


Figure 140: Pasting XML Data into Grid

If the pasted text contains multiple lines of tab-separated values, it can be considered as a matrix of values. By pasting this matrix of values into the **Grid** editor, the result will be a matrix of cells. If the operation is performed inside existing cells, the existing values will be overwritten and new cells will be created when needed. This is useful, for example, when trying to transfer data from spreadsheet-like editors to the **Grid** editor.

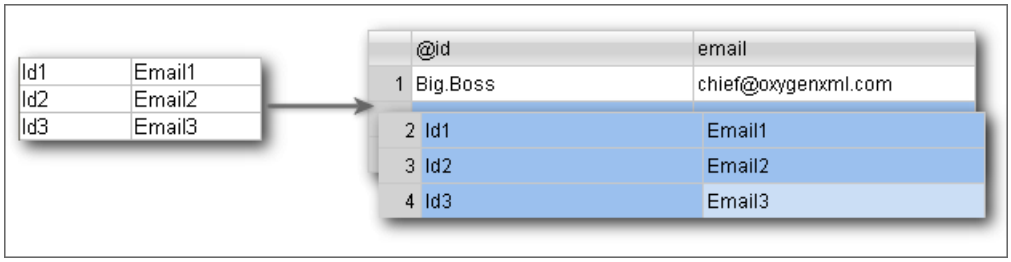


Figure 141: Pasting Tab-Separated Values into Grid

Content Completion Assistant in Grid Mode

If the edited document is associated with a schema (DTD, XML Schema, Relax NG, etc.), the **Grid** editing mode offers a *Content Completion Assistant* for the names and values of elements and attributes. If you choose to insert an element that has required content, the sub-tree of needed elements and attributes are automatically included.

To display the content completion pop-up menu, simply start editing a cell (for example, double-click a cell) or press **Ctrl + Space (Command + Space on OS X)** on your keyboard.

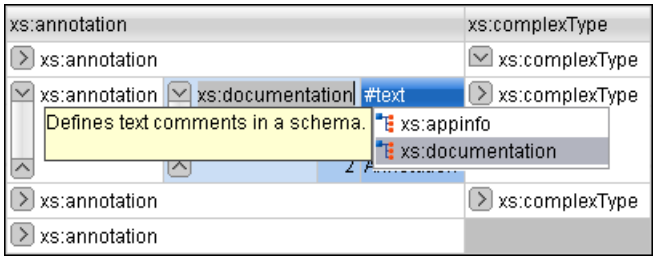


Figure 142: Content Completion in Grid Editing Mode

Editing XML Documents in Author Mode

The **Author** editing mode in Oxygen XML Editor allows you to visually edit XML documents in a user-friendly interface that is similar to a WYSIWYG word processor. This makes structured authoring easier for people who are not familiar with XML and it also provides easier access to the XML structure for XML experts. Oxygen XML Editor provides support for visually editing the most commonly used XML vocabularies in **Author** mode, including DITA, DocBook, TEI, and XHTML.

Adding text content in **Author** mode is as simple as doing so in a standard text editor but the content is rendered similar to how you will see it in the output. Tables, images, and media objects (such as videos) are also rendered comparable to the output. You can even play audio and video objects directly in **Author** mode and it includes an intuitive *Image Map Editor*. You can easily change the rendering by selecting one of the preset *main styles* from the ***Styles*** *drop-down menu* (available on the toolbar) and combine multiple *alternate styles* that behave like layers. You can also use the options in the ***Tags Display Mode*** *drop-down menu* to control how much XML markup is displayed in **Author** mode and there are various features and views that provide information about the XML structure based upon your current location within the document.

Author mode provides numerous helpful editing actions, many of which are specific to the type of document you are editing and it includes a variety of other powerful editing features, such as keyboard shortcuts, *drag and drop support*, a *Smart Paste mechanism*, and an *intelligent Content Completion Assistant*. **Author** mode also allows you to visualize and *manage profiled content*, you can collaborate with others with various *review features* (such as the ability to add comments, track changes, or highlight content), and includes many other unique features.

To switch to this mode, click the **Author** button at the bottom of the editing area.

To watch our video demonstration about some of the features available in the visual **Author** editing mode, go to *https://www.oxygenxml.com/demo/WYSIWYG_XML_Editing.html*.

Author Mode User Roles

There are two main types of users for the **Author** mode: *framework developers* and *content authors*.

Framework Developers

A *framework developer* is a technical person with advanced XML knowledge who defines the *framework* for authoring XML documents in the visual editor. Once the *framework* is created or edited by the developer, it is distributed as a deliverable component ready to plug into the application for the content authors.

The *framework* (document type) configuration defines a type of XML document by specifying all the details needed for editing the content of XML documents in **Author** mode.

The *framework* details that are created and customized by the developer include:

- The CSS stylesheet that drives the visual rendering of the document.
- The rules for associating an XML schema with the document, which is needed for the content completion assistance and validation of the document.
- Transformation scenarios for the document.
- Configuration of *XML Catalogs*.
- Custom actions available as buttons on the toolbar or in menus.

Oxygen XML Editor includes some ready-to-use predefined document types for XML *frameworks*, such as DocBook, DITA, TEI, JATS, and XHTML.

Content Authors

A *content author* does not need to have advanced knowledge about XML markup, operations such as validation of XML documents, or applying XPath expressions to an XML document. The content author just uses the *framework* set up by the developer in the application and starts editing the content of XML documents without editing the XML tags directly.

Styling XML Documents in Author Mode

The structure of an XML document and the required restrictions on its elements and their attributes are defined with an XML schema. This makes it easier to edit XML documents in a visual editor. For more information about schema association, see *Associating a Schema to XML Documents* on page 464. The **Author** mode renders the content of the XML documents visually, based on a CSS stylesheet associated with the document.

Associating a Stylesheet with an XML Document

The rendering of an XML document in the **Author** mode is driven by a CSS stylesheet that conforms to the *version 2.1 of the CSS specification* from the W3C consortium. Some CSS 3 features, such as namespaces and custom extensions, of the CSS specification are also supported. Oxygen XML Editor also supports stylesheets coded with the LESS dynamic stylesheet language.

There are several methods for associating a stylesheet (CSS or LESS) with an XML document:

1. Insert the `xml-stylesheet` processing instruction with the `type` attribute at the beginning of the XML document. If you do not want to alter your XML documents, *you should create a new document type (framework)*.

   CSS example:

   ```
   <?xml-stylesheet type="text/css" href="test.css"?>
   ```

   LESS example:

   ```
   <?xml-stylesheet type="text/css" href="test.less"?>
   ```

   Note: XHTML documents need a `link` element, with the `href` and `type` attributes in the `head` child element, as specified in the *W3C CSS specification*. XHTML example:

   ```
   <link href="/style/screen.css" rel="stylesheet" type="text/css"/>
   ```

   Tip: You can also insert the `xml-stylesheet` processing instruction by using the **Associate XSLT/CSS Stylesheet** action that is available on the toolbar or in the **Document** > **XML Document** menu.
2. Add a new CSS or LESS file to a *framework* (document type). To do so, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Document Type Association**. Edit the appropriate *framework*, open the **Author** tab, then the **CSS** subtab. Press the **New** button to add a new CSS or LESS file.

   Note: The predefined *frameworks* are read-only, so you need to ***Extend*** or ***Duplicate*** them to configure them as custom *frameworks*.

You can read more about associating a CSS to a document type in *Customizing the Main CSS of a Framework* on page 1237.

If a document has no CSS association or the referenced stylesheet files cannot be loaded, a default one is used. A warning message is also displayed at the beginning of the document, presenting the reason why the CSS cannot be loaded.

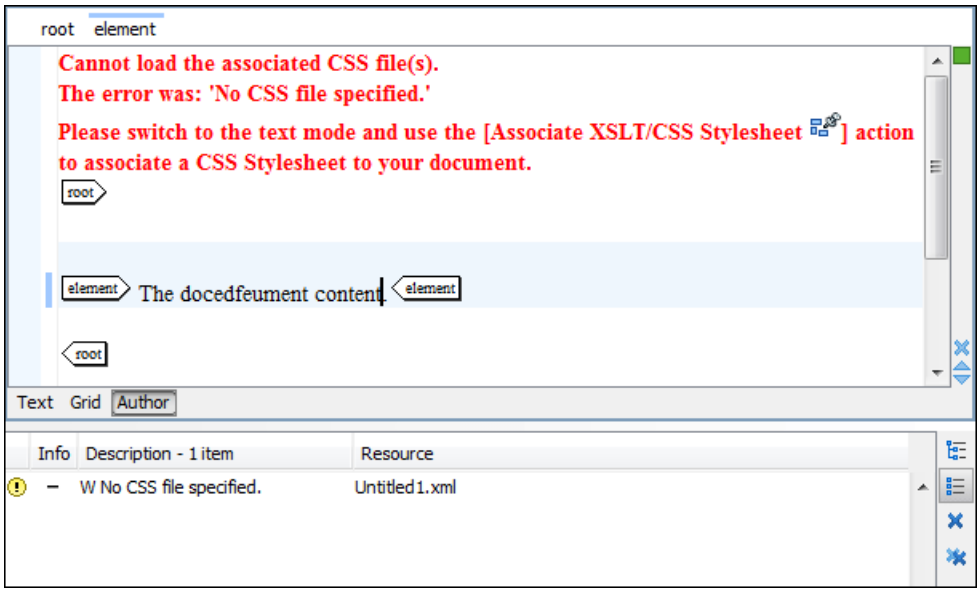


Figure 143: Document with no CSS association default rendering

Selecting and Combining Multiple CSS Styles

Oxygen XML Editor provides a **Styles** drop-down menu on the toolbar that allows you to select one *main (non-alternate) CSS style* and multiple *alternate CSS styles*. This makes it easy to change the look of the document as it appears in **Author** mode and the output without having to continually edit the CSS stylesheets.

Tip: For information about configuring the **Styles** drop-down menu, see *Configuring and Managing Multiple CSS Styles* on page 1256.

You can select a *main CSS style* that applies to the whole document and then *alternate styles* that are applied as layers to specific parts of the document. For example, in the subsequent figure, a DITA document has the **Century** style selected for the *main CSS* and the *alternate styles* **Full width**, **Show table column specification**, **Hints**, and **Inline actions** are combined for additive styling to specific parts of the document.

The selections from the **Styles** drop-down menu are persistent, meaning that Oxygen XML Editor will remember the selections when subsequent documents are opened.

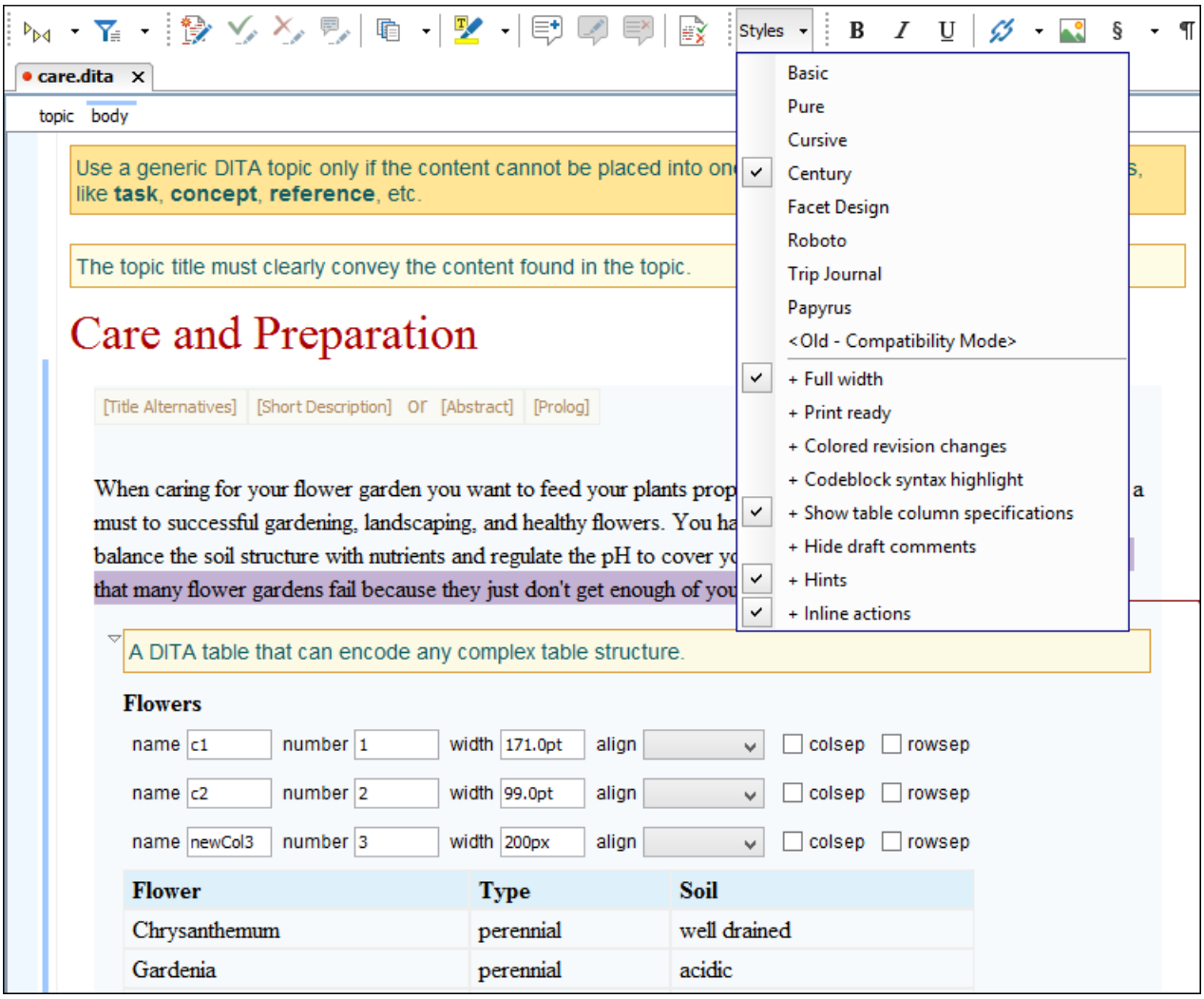


Figure 144: Styles Drop-down Menu in a DITA Document

Related Information:

Associating a Schema to XML Documents on page 464
Configuring and Managing Multiple CSS Styles on page 1256

Navigating the Document Content in Author Mode

Oxygen XML Editor includes some useful features to help you navigate XML documents.

Using the Keyboard

Oxygen XML Editor allows you to quickly navigate through a document using the **Tab** key to move the cursor to the next XML node and **Shift + Tab** to go to the previous one. If you encounter a *space-preserved element* when you navigate through a document and you do not press another key, pressing the **Tab** key will continue the navigation. However, if the cursor is positioned in a *space-preserved element* and you press another key or you position the cursor inside such an element using the mouse, the **Tab** key can be used to arrange the text.

To navigate one word forward or backwards, use **Ctrl + RightArrow (Command + RightArrow on OS X)**, and **Ctrl + LeftArrow (Command + LeftArrow on OS X)**, respectively. Entities and hidden elements are skipped. To position the cursor at the beginning or at the end of the document you can use **Ctrl + Home (Command + Home on OS X)**, and **Ctrl + End (Command + End on OS X)**, respectively.

Navigation Buttons

Oxygen XML Editor includes some actions that help you to quickly navigate to a particular modification. These navigation buttons are available in the main toolbar and the actions can also be accessed from the **Find** menu. The three actions include:

- **Last Modification** - Moves the cursor to the last modification in any opened document.
- **Back** - Moves the cursor to the previous position.
- **Forward** - Moves the cursor to the next position after the **Back** button has been used at least once.

Navigating with the Outline View

Oxygen XML Editor includes a very useful ***Outline*** *view* that displays a hierarchical tag overview of the currently edited XML Document.

You can use this view to quickly navigate through the current document by selecting nodes in the outline tree. It is synchronized with the editor area, so when you make a selection in the **Outline** view, the corresponding nodes are highlighted in the editor area.

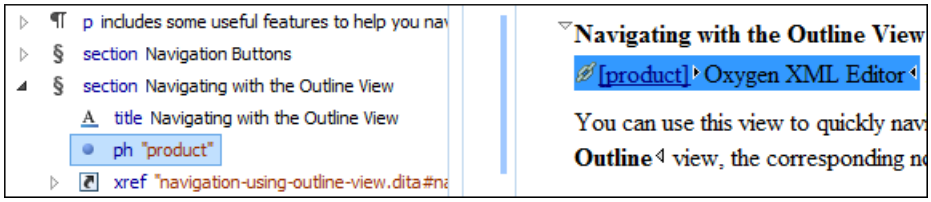


Figure 145: Outline View Navigation in Author Mode

Using the Breadcrumb to Navigate

A *breadcrumb* on the top stripe indicates the path from document root to the current element. It can also be used as a helpful tool to navigate to specific elements throughout the structure of the document.

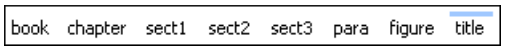


Figure 146: Breadcrumb in Author Mode

The last element listed in the *breadcrumb* is the element at the current cursor position. The last element is also highlighted by a thin light blue bar for easier identification. Clicking an element from the *breadcrumb* selects the entire element and navigates to it in the editor area.

Using the Linking Support

When working on multiple documents that reference each other (references, external entities, XInclude, DITA conref, etc), the **linking support** is useful for navigating between the documents. In the predefined customizations that are bundled with Oxygen XML Editor, links are marked with an icon representing a chain link (). When hovering over the icon, the mouse pointer changes its shape to indicate that the link can be accessed and a tooltip presents the destination location. Click the link to open the referenced resource in the editor or system browser. The same effect can be obtained by using the **Document** > **File** > **Open file at cursor (Ctrl + Enter (Command + Enter on OS X))** action when the cursor is inside a link element.

Note: Depending on the referenced file type, the target link will either be opened in the Oxygen XML Editor or in the default system application. If the target file does not exist, Oxygen XML Editor prompts you to create it.

Navigating with Bookmarks

A position in a document can be marked with a *bookmark*. You can then quickly go to the marked position with a keyboard shortcut or a menu action. This is useful when navigating large documents or working on multiple documents where the cursor needs to move between several marked positions. The *bookmarks* are displayed with a small icon on the vertical strip to the left of the editor. You can place up to nine distinct *bookmarks* in any

document. Shortcut keys are available to place the *bookmarks* or to return to any of the marked positions. You can configure these shortcut keys in the ***Options > Menu Shortcut Keys*** menu.

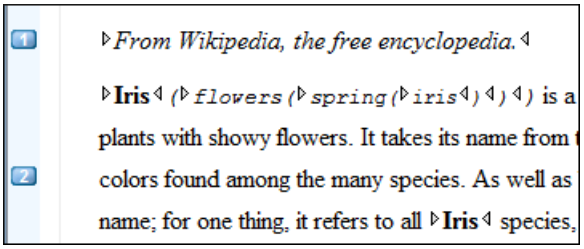


Figure 147: Editor Bookmarks

A *bookmark* can be inserted in **Author** mode by doing one of the following:

- Click in the vertical stripe on the left side of the editor.
- Select the **Create Bookmark (F9)** action from the **Edit** > **Bookmarks** menu.

A *bookmark* can be removed by right-clicking its icon on the vertical stripe and select **Remove** or **Remove all (Ctrl +F7 (Command+F7 on OS X))**.

You can navigate the *bookmarks* by using one of the actions available on the **Edit** > **Bookmarks** > **Go to** menu or by using the shortcut keys that are listed in that menu.

Displaying the Markup

You can control the amount of markup that is displayed in the **Author** mode with various levels of tag modes for both *block* and *in-line* elements.

The following dedicated tag modes are available from the **Tags Display Mode** drop-down menu (available on the toolbar):

Full Tags with Attributes
: Displays full tag names with attributes for both *block* and *inline* elements.

Full Tags
: Displays full tag names without attributes for both *block* and *inline* elements.

Block Tags
: Displays full tag names for *block elements* and simple tags without names for *inline elements*.

Inline Tags
: Displays full tag names for *inline elements*, while *block elements* are not displayed.

Partial Tags
: Displays simple tags without names for *inline elements*, while *block elements* are not displayed.

No Tags
: No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Configure Tags Display Mode
: Opens the ***Author*** *preferences page* where you can configure options in regards to *tags*, such as the default **Tags Display Mode**, **Tags Background Color**, **Tags Foreground Color**, **Tags Font**, and whether or not Oxygen XML Editor will use a **Compact Tag Layout** for displaying the tags (this option tries to group consecutive block tags on the same line).

Note: The associated CSS information is used to determine whether a tag should be considered *inline* or *block*. If the current document does not have an associated CSS stylesheet, then the **Full Tags** mode will be used.

Editing Content in Author Mode

The **Author** mode includes a large variety of user-friendly authoring features to help you work with XML content, including numerous toolbar, menu, and shortcut actions and some specialized content editing features.

Undo/Redo Actions

The typical undo and redo actions are available with shortcuts or in the **Edit** menu:

Undo (Ctrl + Z (Command + Z on OS X))

Reverses a maximum of 200 editing actions (configurable with the ***Undo history size*** *option* in the **Editor** preferences page) to return to the preceding state.

Note: Complex operations such as **Replace All** or **Indent selection** count as single undo events.

Redo (Ctrl + Y (Command + Shift + Z on OS X, Ctrl + Shift + Z on Linux/Unix))

Recreates a maximum of 100 editing actions that were undone by the **Undo** function.

Copy and Paste Actions

The typical copying and pasting actions are available with shortcuts or in the contextual menu (or the **Edit** menu):

Cut (Ctrl + X (Command + X on OS X))

Removes the current selected content from the document and places it in the clipboard.

Copy (Ctrl + C (Command + C on OS X))

Places a copy of the current selected content in the clipboard.

Paste (Ctrl + V (Command + V on OS X))

Inserts the current clipboard content into the document at the cursor position.

Select All (Ctrl + A (Command + A on OS X))

Selects the entire content of the current document.

Entering Text in Elements

By default, you can only enter text in elements that accept text content. If the element is declared as *empty* or *element only* in the associated schema, you are not allowed to insert text in it. Instead, a warning message is displayed.

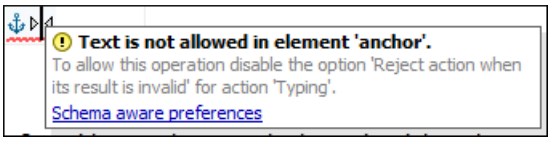


Figure 148: Editing in empty element warning

To allow text to be inserted in these instances, go to the **Schema-Aware** preferences page and deselect the ***Reject action when its result is invalid*** *option in the* ***Typing*** *actions section*.

Editing Text Content Without Modifying the XML Markup

You can use the options in the ***Tags Display Mode*** *drop-down menu* (available on the toolbar) to control how tags are displayed in **Author** mode. This can help you to clearly see where the current cursor position is within the tag structure so that you can avoid making unintended modifications to the XML markup. You can also switch to the ***Grid*** *editing mode* to modify text content without affecting the XML tags.

Changing the Font Size (Zoom)

The font size of the editor panel can be changed with the following actions that are available with shortcuts or in the **Document** > **Font size** menu:

Increase editor font (Ctrl + NumPad+ (Command + NumPad+ on OS X) or Ctrl + MouseWheelForward (Windows/Linux)

Increases the font size (zooms in) with one point for each execution of the action.

Note: For Mac OS X, if you activate the ***Enable mouse-wheel zooming*** *option* in the **Editor** preferences page, you can use **Command + MouseWheelForward** to increase the font size (zoom in). It is disabled by default due to the way inertia affects the mouse wheel on Mac OS X.

Decrease editor font (Ctrl + NumPad- (Command + NumPad- on OS X) or Ctrl + MouseWheelBackwards (Windows/Linux)

Decreases the font size (zooms out) with one point for each execution of the action.

Note: For Mac OS X, if you activate the ***Enable mouse-wheel zooming*** *option* in the **Editor** preferences page, you can use **Command + MouseWheelBackwards** to decrease the font size (zoom out). It is disabled by default due to the way inertia affects the mouse wheel on Mac OS X.

Normal editor font (Ctrl + 0 (Command + 0 on OS X))

Resets the font size to *the value of the editor font set in the* ***Fonts*** *preferences page*.

Related Information:

Editing XML Markup in Author Mode on page 336
Drag and Drop in Author Mode on page 343
Smart Paste Mechanism on page 343
Content Completion Assistant in Author Mode on page 345
Contextual Menu Actions in Author Mode on page 437
Frequently Used Shortcut Keys on page 18

Editing XML Markup in Author Mode

Oxygen XML Editor includes some useful actions that allow you to easily edit XML markup in **Author** mode. Most of these actions are available in the contextual menu and some of them have simple keyboard shortcuts.

Selecting XML Markup in Author Mode

Selecting XML tags in Oxygen XML Editor is very simple with several methods for selecting entire elements:

- **Breadcrumb** - Click the element (XML tag) on the *breadcrumb* displayed at the top of the editing window.
- **Outline View** - Click the element name in the ***Outline*** *view*.
- **Full Tags Mode** - While editing in ***Full Tags*** *mode*, click the start or end tag of the element in the editor.
- **Mouse Selection** - While editing in ***Full Tags*** *mode*, click before the start tag of the element, drag the selection, and release the mouse button after the end tag.
- **Shift + Arrow Keys** - While editing in ***Full Tags*** *mode*, place the cursor before the start tag of the element, press and hold **Shift**, and use the arrow keys to make the selection (including the end tag).

Note: If the selection does not include the entire element (for example you do not include the end tag of the element), Oxygen XML Editor will automatically close the appropriate tags when pasting the copied selection. This ensures that the pasted content will always result in *well-formed XML*.

Using the Breadcrumb in Author Mode

A *breadcrumb* on the top stripe indicates the path from document root to the current element. It can also be used as a helpful tool to insert and edit specific elements in the document structure.

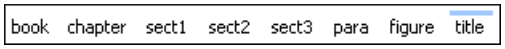


Figure 149: Breadcrumb in Author Mode

The last element listed in the *breadcrumb* is the element at the current cursor position. The last element is also highlighted by a thin light blue bar for easier identification. Clicking an element from the *breadcrumb* selects the entire element in the editor area and each element provides a contextual menu with access to the following actions:

Edit Attributes

Opens the *in-place attributes editor* that allows you to easily edit the attributes of an element.

Edit Profiling Attributes

Allows you to select the *profiling attributes* that apply to a certain element.

Append child

Opens a content completion list that allows you to select an element to be inserted as a child of the selected element.

Insert before

Opens a content completion list that allows you to select an element to be inserted (as a sibling) before the selected element.

Insert after

Opens a content completion list that allows you to select an element to be inserted (as a sibling) after the selected element.

Cut

Removes the selected element and copies it to the clipboard, while preserving the styles of the content.

Copy

Copies the selected element to the clipboard, while preserving the styles of the copied content.

Paste

Pastes a well-formed element from the clipboard at currently selected position in the breadcrumb.

Paste before

Insert a well-formed element (from the clipboard) before the currently selected element.

Paste after

Insert a well-formed element (from the clipboard) after the currently selected element.

Paste as XML

Inserts clipboard content that is considered to be well-formed XML content, preserving its XML structure.

Delete

Deletes the currently selected element.

Toggle Comment

Encloses the currently selected element in an XML comment, if the element is not commented, or removes the comment if it is commented.

Rename Element

Opens the **Rename** dialog box that allows you to rename the currently selected element and other elements with the same name.

Tip: The tag names displayed in the *breadcrumb* can be customized with an **Author** mode extension class that implements the `AuthorBreadCrumbCustomizer` API. See the *Oxygen SDK* for more details.

Move Nodes

You can move XML nodes in the current document by using the following actions in the **Refactoring** submenu of the contextual menu (or from the **Document** > **Markup** menu):

Move Up (Alt + UpArrow)

Moves the current node or selected nodes in front of the previous node.

Move Down (Alt + DownArrow)

Moves the current node or selected nodes after the subsequent node.

Tip: The easiest way to move nodes is to use the **Alt + UpArrow** and **Alt + DownArrow** shortcut keys.

Promote/Demote List Item Nodes

You can easily promote or demote selected list item nodes within ordered lists or unordered lists by using the following keyboard shortcuts:

Promote (Shift + Tab)

Promotes an entirely selected list item node to be a sibling of its parent node (the list item is moved to the left). It also works for selections of multiple list item nodes as long as all the selected nodes are siblings (on the same hierarchical level).

Demote (Tab)

Demotes an entirely selected list item node (the list item is moved to the right). It also works for selections of multiple list item nodes as long as all the selected nodes are siblings (on the same hierarchical level).

Join or Split Elements

You can join or split elements in the current document by using the following actions in the **Refactoring** submenu of the contextual menu (or from the **Document** > **Markup** menu):

Join Elements

Joins two adjacent *block elements* that have the same name. The action is available only when the cursor position is between the two adjacent *block elements*. Also, joining two *block elements* can be done by pressing the **Delete** or **Backspace** keys and the cursor is positioned between the boundaries of these two elements.

Tip: Specifically, the **Delete** or **Backspace** keys can be used to join *block elements* in the following situations:

- The cursor is located before the end position of the first element and **(Delete)** key is pressed.
- The cursor is located after the end position of the first element and **(Backspace)** key is pressed.
- The cursor is located before the start position of the second element and **(Delete)** key is pressed.
- The cursor is located after the start position of the second element and **(Backspace)** key is pressed.

If the element has no sibling or the sibling element has a different name, an **Unwrap** operation will be performed.

Split Element (Alt + Shift + D (Ctrl + Alt + D on OS X))

Splits the content of the closest element that contains the position of the cursor. Thus, if the cursor is positioned at the beginning or at the end of the element, the newly created sibling will be empty.

Rename Elements

You can rename elements by using the following action in the **Refactoring** submenu of the contextual menu (or from the **Document** > **Markup** menu):

Rename Element

The element from the cursor position, and any elements with the same name, can be renamed according with the options from the **Rename** dialog box.

Surround Content with Tags (Wrap)

You can surround a selection of content with tags (*wrap* the content) by using the following action in the **Refactoring** submenu of the contextual menu (or from the **Document** > **Markup** menu):

Surround with Tags (Ctrl + E (Command + E on OS X))

Allows you to choose a tag to enclose a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.

- If the ***Position cursor between tags*** *option* is selected in the **Content Completion** preferences page, the cursor is placed between the start and end tag.
- If the ***Position cursor between tags*** *option* is not selected in the **Content Completion** preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

Surround with '[tag]' (Ctrl + ForwardSlash (Command + ForwardSlash on OS X))

Surround the selected content with the last tag used.

Unwrap the Content of Elements

You can unwrap the content of an element by using the following action in the **Refactoring** submenu of the contextual menu (or from the **Document** > **Markup** menu):

Delete Element Tags

Deletes the tags of the closest element that contains the position of the cursor. This operation is also executed if the start or end tags of an element are deleted by pressing the **Delete** or **Backspace** keys.

Tip: Specifically, the **Delete** or **Backspace** keys can be used to unwrap the content of an element in the following situations:

- The cursor is located before the start position of the element and **(Delete)** key is pressed.
- The cursor is located after the start position of the element and **(Backspace)** key is pressed.
- The cursor is located before the end position of the element and **(Delete)** key is pressed.
- The cursor is located after the end position of the element and **(Backspace)** key is pressed.

If the element has no sibling or the sibling element has a different name, an **Unwrap** operation will be performed.

Remove Markup from Blocks of Content

You can remove the markup from the current element by highlighting the appropriate block of content and using the following action in the **Refactoring** submenu of the contextual menu (or from the **Document** > **Markup** menu):

Remove All Markup

Removes all the XML markup inside the selected block of content and keeps only the text content.

Tip: You can use the **(Delete)** or **(Backspace)** keys to remove markup, in which case the elements in the selected block will be unwrapped or joined with their sibling, or if the current element is empty, the element tags will be deleted.

Remove Text from Selected Markup

You can remove the text from elements by highlighting the appropriate block of content and using the following action in the **Refactoring** submenu of the contextual menu (or from the **Document** > **Markup** menu):

Remove Text

Removes the text content of the selected block of content and keeps the markup in tact with empty elements.

Other Refactoring Actions

You can also manage the structure of the markup by using the other specific XML refactoring actions that are available in the **Refactoring** submenu of the contextual menu:

Attributes submenu

Contains predefined XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

Add/Change attribute

Allows you to change the value of an attribute or insert a new one.

Delete attribute

Allows you to remove one or more attributes.

Rename attribute

Allows you to rename an attribute.

Replace in attribute value

Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

Comments submenu

Contains predefined XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

Delete comments

Allows you to delete comments found inside one or more elements.

DITA submenu

Contains predefined XML refactoring operations that pertain to DITA documents with some of the information preconfigured based upon the current context.

Change topic ID to file name

Use this operation to change the ID of a topic to be the same as its file name.

Convert conrefs to conkeyrefs

Use this operation to convert `conref` attributes to `conkeyref` attributes.

Convert simple tables to CALS tables

Use this operation to convert DITA simple tables to CALS tables.

Convert to Concept

Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept).

Convert to Reference

Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference).

Convert to Task

Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task).

Convert to Topic

Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic).

Convert to Troubleshooting

Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting).

Elements submenu

Contains predefined XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

Delete element

Allows you to delete elements.

Delete element content

Allows you to delete the content of elements.

Insert element

Allows you to insert new elements.

Rename element

Allows you to rename elements.

Unwrap element

Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

Wrap element

Allows you to surround elements with element tags.

Wrap element content

Allows you to surround the content of elements with element tags.

Fragments submenu

Contains predefined XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

Insert XML fragment

Allows you to insert an XML fragment.

Replace element content with XML fragment

Allows you to replace the content of elements with an XML fragment.

Replace element with XML fragment

Allows you to replace elements with an XML fragment.

JATSKit submenu

Contains predefined XML refactoring operations that pertain to JATS documents with some of the information preconfigured based upon the current context.

Add BITS DOCTYPE - NLM/NCBI Book Interchange 2.0

Adds an NLM 'BITS' 2.0 DOCTYPE declaration.

Add Blue DOCTYPE - NISO JATS Publishing 1.1

Adds a JATS 'Blue' 1.1 DOCTYPE declaration.

Normalize IDs

Assigned IDs are normalized and IDs are assigned to some elements that are missing them.

Related Information:

Editing Content in Author Mode on page 334
Displaying the Markup on page 234
Refactoring XML Documents on page 491
Selecting Content in Author Mode on page 344
Content Completion Assistant in Author Mode on page 345
Contextual Menu Actions in Author Mode on page 437
Frequently Used Shortcut Keys on page 18

Editing Attributes in Author Mode

You can easily edit attributes in **Author** mode by using the ***Attributes View*** and Oxygen XML Editor also allows you to edit attribute and element values in-place, directly in the **Author** mode, using an in-place attribute editor.

In-place Attributes Editor

Oxygen XML Editor includes an in-place attributes editor in **Author** mode. To edit the attributes of an XML element in-place, do one of the following:

- Select an element or place the cursor inside it and then press the **Alt + Enter** keyboard shortcut.
- Double-click any named start tag when the document is edited in one of the following *display modes.*: **Full Tags with Attributes**, **Full Tags**, **Block Tags**, or **Inline Tags**.

This opens an in-place attributes editor that contains the same content as the **Attributes** view. By default, this editor presents the **Name** and **Value** fields, with the list of all the possible attributes collapsed.

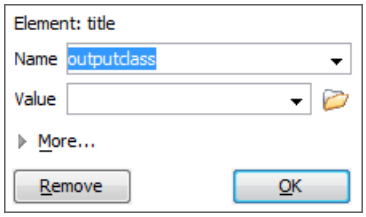


Figure 150: In-place Attributes Editor

Name Combo Box

Use this combo box to select an attribute. The drop-down list displays the list of possible attributes allowed by the schema of the document, as in the **Attributes** view.

Value Combo Box

Use this combo box to add, edit, or select the value of an attribute. If the selected attribute has predefined values in the schema, the drop-down list displays those possible values.

If you click **More** while in the collapsed version, it is expanded to the full version of the in-place attribute editor.

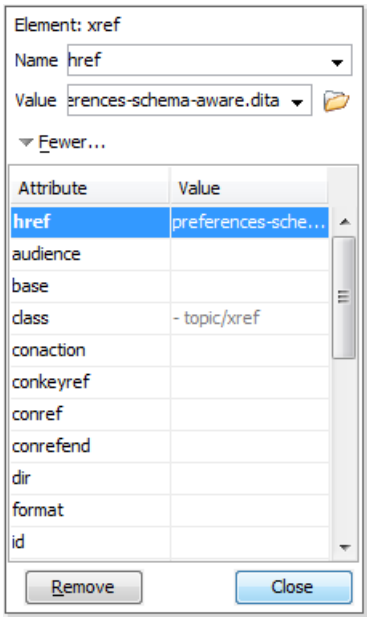


Figure 151: In-place Attributes Editor (Full Version)

The full version includes a table grid, similar to the **Atributes** view, that presents all the attributes for the selected element.

Note: If the cursor is located inside read-only content, the attribute names and values are faded and you cannot add, edit, or remove values.

Related Information:

Attributes View in Author Mode on page 224

Folding XML Elements in Author Mode

When working with a large document, the *folding* support in Oxygen XML Editor can be used to collapse some element content leaving only the parts that you need to edit in focus. Expanding and collapsing works on individual elements. Expanding an element leaves the child elements unchanged.

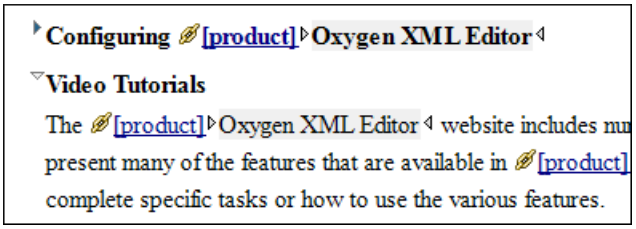


Figure 152: Folding of XML Elements in Author Mode

The fact that the folds are persistent is a unique feature of Oxygen XML Editor, meaning the next time you open the document the folds are restored to its last state.

Folding Actions in Author Mode

Foldable elements are marked with a small triangle (/) on the left side of the editor panel. If you hover over that arrow, the entire content of the element is highlighted by a dotted border for quick identification of the *foldable* area. To toggle the fold, simply click the icon. Also, the following actions are available in the **Folding** sub-menu of the contextual menu or from the **Document** > **Folding** menu:

Toggle Fold (or you can simply click on the / arrow)

Toggles the state of the current fold.

Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on OS X))

Folds all the elements except the current element.

Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on OS X))

Folds the child elements that are indented one level inside the current element.

Expand Child Folds

Unfolds all child elements of the currently selected element.

Expand All (Ctrl + NumPad* (Command + NumPad* on OS X))

Unfolds all elements in the current document.

To watch our video demonstration about the *folding* support in Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/FoldingSupport.html*.

Drag and Drop in Author Mode

The Oxygen XML Editor **Author** mode includes support for dragging and dropping content in XML documents.

When editing content in **Author** mode, entire sections or chunks of data can be moved or copied by using the drag and drop feature. The following situations can be encountered:

- When both of the drag and drop sources are from the **Author** mode editor, a well-formed XML fragment is transferred. The section is balanced before dropping it by adding matching tags when needed.
- When the drag source is from the **Author** mode editor but the drop target is a text-based editor, only the text inside the selection is transferred as it is.
- The text dropped from another text editor or another application into the **Author** mode editor is inserted without changes.

Related Information:

Smart Paste Mechanism on page 343

Smart Paste Mechanism

The **Author** editing mode includes a *Smart Paste* feature that preserves certain style and structure information when copying content and pasting it into document types that support the feature. You can copy content from various sources, including web pages, external applications (such as Office-type applications), or other document types within Oxygen XML Editor, and then paste it into DITA, TEI, DocBook, JATS, and XHTML documents. Oxygen XML Editor preserves the original text styling (such as bold, italics, underline) and formatting (such as lists, tables, paragraphs) and considers various pasting solutions to keep the resulting document valid.

The styles and general layout of the pasted content are converted to the equivalent XML markup for the target document type while preserving certain style and structure information. For example, if you copy content that includes multiple paragraphs and then paste it in **Author** mode, the multiple paragraph structure is preserved. If you paste the content in a location where the resulting XML would not be valid, Oxygen XML Editor will attempt to place it in a valid location, and may prompt you with one or more choices for where to place it.

Smart Paste Options

By default, the *Smart Paste* feature is enabled in Oxygen XML Editor. There are several options in the ***Schema Aware*** *preferences page* that control the *Smart Paste* mechanism:

- ***Smart paste and drag and drop*** - This option determines whether or not Oxygen XML Editor will try to find an appropriate insert position when the current location is not valid for the pasted content. This option is selected by default.
- ***Reject action when its result is invalid*** - If you select this option, Oxygen XML Editor will not let you paste content into a position where it would be invalid. This option is deselected by default.
- ***Convert external content on paste*** - This option determines whether or not Oxygen XML Editor will convert the styling and formatting of copied content from external sources when pasting it into a document type that supports the feature. This option is selected by default.

- ***Convert even when pasting inside space-preserve elements*** - If you select this option, the *Smart Paste* feature will also work when pasting external content into a *space-preserve* element (such as a `codeblock`). This option is deselected by default.

Smart Paste Supported Document Types

The *Smart Paste* feature is supported for the following document types (*frameworks*):

- DITA
- DocBook 4
- DocBook 5
- TEI
- XHTML
- JATS

To watch our video demonstration about the *Smart Paste* support, go to *https://www.oxygenxml.com/demo/Smart_Paste_Copy_Paste_from_Web_Office_Documents_to_DITA_DocBook_TEI_XHTML_Documents.html*.

Related Information:

Customizing Smart Paste Support on page 1211

Selecting Content in Author Mode

Oxygen XML Editor includes a variety of keyboard shortcuts that allow you to easily select content in **Author** mode.

Selection Shortcuts in Author Mode

Ctrl + A (Meta + A on Mac OS X)

Selects all content in the document.

Shift + Left/Right Arrow Keys

Begins a continuous selection at the cursor position and extends it one character at a time in the direction that you press the arrow keys.

Shift + Up/Down Arrow Keys

Begins a continuous selection at the cursor position and extends it one line at a time in the direction that you press the arrow keys.

Ctrl + Shift + Left/Right Arrow Keys (Meta + Shift + Left/Right Arrow Keys on Mac OS X)

Begins a continuous selection at the cursor position and extends it one word at a time in the direction that you press the arrow keys.

Shift + Home

Begins a continuous selection at the cursor position and extends it to the beginning of the current line (on Mac OS X, it extends to the beginning of the document).

Shift + End

Begins a continuous selection at the cursor position and extends it to the end of the current line (on Mac OS X, it extends to the end of the document).

Ctrl + Shift + Home

Begins a continuous selection at the cursor position and extends it to the beginning of the document.

Ctrl + Shift + End

Begins a continuous selection at the cursor position and extends it to the end of the document.

Shift + PageUp

Begins a continuous selection at the cursor position and extends it up one screen page.

Shift + PageDown

Begins a continuous selection at the cursor position and extends it down one screen page.

Double-Click

Selects the word at the cursor position.

Triple-Click

Selects the node at the cursor position.

Right-Click > Select > Element

Selects the entire element at the current cursor position.

Right-Click > Select > Content

Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

Right-Click > Select > Parent

Selects the entire parent element at the current cursor position.

Content Completion Assistant in Author Mode

One of the most useful features in **Author** mode is the *Content Completion Assistant*. It offers a list of elements, attributes, attribute values, and other options that are valid in the current editing context.

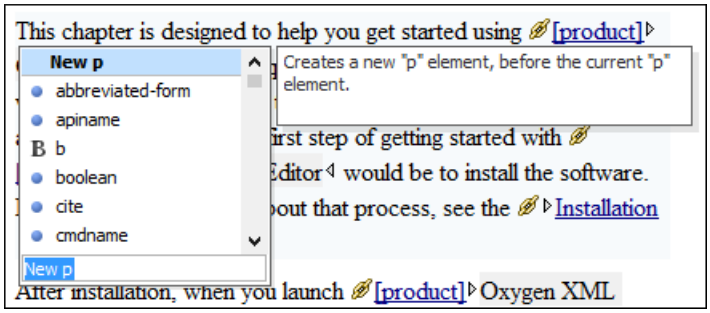


Figure 153: Content Completion Assistant in Author Mode

The *Content Completion Assistant* is enabled by default. To disable it, *open the **Preferences** dialog box (**Options > Preferences**)*, go to **Editor** > **Content Completion**, and deselect the ***Enable content completion** option*.

Using the Content Completion Assistant in Author Mode

To activate the feature in **Author** mode, use any of the following shortcut keys:

- **Enter**
- **Ctrl + Space (Command + Space on OS X)**
- **Alt + ForwardSlash (Command + Alt + ForwardSlash on OS X)**

You can navigate through the list of proposals by using the **Up** and **Down** keys on your keyboard. For each selected item in the list, the *Content Completion Assistant* displays a documentation window. You can customize the size of the documentation window by dragging its top, right, and bottom borders.

To insert the selected content in **Author** mode, simply press **Enter**.

Types of Proposals Listed in the Content Completion Assistant

The *Content Completion Assistant* offers the following types of proposed actions depending on the current context:

- Insert allowed elements for the current context schema and the list of proposals contains elements depending on the elements inserted both before and after the cursor position.
- Insert element values if such values are specified in the schema for the current context.
- Insert new undeclared elements by entering their name in the text field.
- Insert CDATA sections, comments, processing instructions.
- Insert *code templates*.
- If invoked on a selection that only contains an element start or end tag (remember that you can see all element tags while working in ***Full Tags** mode*), it will allow you to rename the element.

- If invoked on a selection of multiple elements or other content, it will allow you to surround the content with certain tags.
- If invoked on an empty list item that is the last element of the list, it will allow you to convert the list item to a paragraph.
- If the ***Show all possible elements in the content completion list*** *option from the* ***Schema-Aware*** *preferences page* is selected, the content completion pop-up window will present all the elements defined by the schema. When choosing an element from this section, the insertion will be performed using the schema-aware smart editing features.

  Note: By default, you are not allowed to insert element names that are not defined by the schema. This can be changed by deselecting the ***Allow only insertion of valid elements and attributes*** *check box from the* ***Schema-Aware*** *preferences page*.

Examples of How the Content Completion Assistant Works

To illustrate how the feature works, consider the following examples of invoking the *Content Completion Assistant* in certain contexts:

- If the cursor is positioned at the beginning or at the end of the element, the first item offered in the *Content Completion Assistant* is a **New <Element>** item. Selecting this item will insert an empty element.

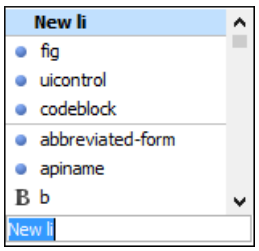


Figure 154: Example (New [Element Name])

- If the cursor is positioned somewhere inside the element, the first entry in the *Content Completion Assistant* is a **Split <Element>** item. In most cases, you can only split the closest *block element* to the cursor position, but if it is inside a list item, the list item will also be proposed for split. Selecting **Split <Element>** splits the content of the specified element around the cursor position.

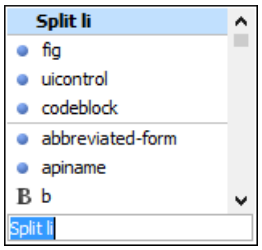


Figure 155: Example (Split [Element Name])

- If the cursor is positioned inside a *space-preserved element* (for example, a *codeblock*), the first choice in the *Content Completion Assistant* is **Enter**, which will insert a new line in the content of the element, followed by **New <Element>**.

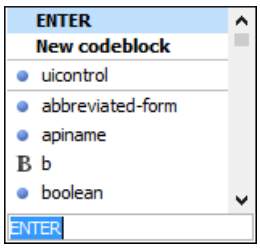


Figure 156: Example ('ENTER' New Line)

- If invoked on a selection that only contains an element start or end tag (remember that you can see all element tags while working in ***Full Tags*** *mode*), it will allow you to rename the element.

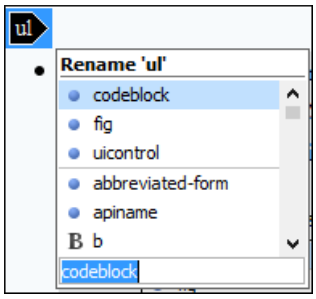


Figure 157: Example (Rename)

- If invoked on a selection of multiple elements or other content, it will allow you to surround the content with certain tags.

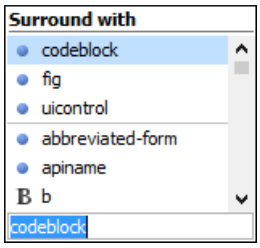


Figure 158: Example (Surround)

Related Information:

Customizing the Content Completion Assistant on page 1241

Set the Schema to be Used for Content Completion

The proposals that are presented in the *Content Completion Assistant* depend on the associated schemas. The DTD, XML Schema, Relax NG, or NVDL schema used to populate the *Content Completion Assistant* is specified in the following methods, in the order of their precedence:

- The *schema specified explicitly in the document*. In this case, Oxygen XML Editor reads the beginning of the document and resolves the location of the DTD, XML Schema, Relax NG schema, or NVDL schema.
- The *default schema declared* in the ***Schema*** *tab of the* ***Document Type*** *configuration dialog box* for the particular document type.

Schema Annotations in Author Mode

A *schema annotation* is a documentation snippet associated with the definition of an element or attribute in a schema. If such a schema is associated with an XML document, the annotations are displayed in the *Content Completion Assistant*.

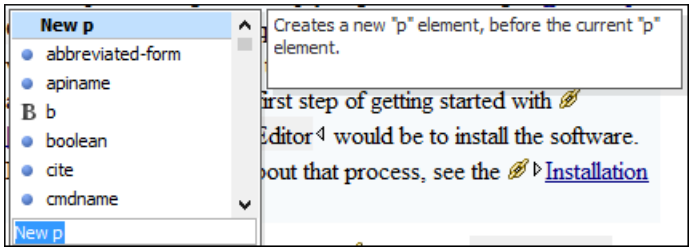


Figure 159: Schema Annotation in the Content Completion Assistant

The schema annotations support is available if the schema type is one of the following:

- XML Schema
- Relax NG
- NVDL schema
- DTD

This feature is enabled by default, but you can disable it by deselecting the ***Show annotations in Content Completion Assistant*** option in the **Annotations** preferences page.

Styling Annotations with HTML

You can use HTML format in the annotations you add in an XML Schema or Relax NG schema. This improves the visual appearance and readability of the documentation window displayed when editing XML documents validated against such a schema. An annotation is recognized and displayed as HTML if it contains at least one HTML element (such as `div`, `body`, p, `br`, `table`, `ul`, or `ol`).

The HTML rendering is controlled by the ***Show annotations using HTML format, if possible*** option in the **Annotations** preferences page. When this options is deselected, the annotations are converted and displayed as plain text and if the annotation contains one or more HTML tags (p, `br`, `ul`, `li`), they are rendered as an HTML document loaded in a web browser. For example, p begins a new paragraph, `br` breaks the current line, `ul` encloses a list of items, and `li` encloses an item of the list.

Collecting Annotations from XML Schemas

In an XML Schema, the annotations are specified in an `<xs:annotation>` element like this:

```
<xs:annotation>
  <xs:documentation>
            Description of the element.
  </xs:documentation>
</xs:annotation>
```

If an element or attribute does not have a specific annotation, then Oxygen XML Editor looks for an annotation in the type definition of that element or attribute.

Collecting Annotations from Relax NG Schemas

For Relax NG schema, element and attribute annotations are made using the `<documentation>` element from the `http://relaxng.org/ns/compatibility/annotations/1.0` namespace. However, any element outside the Relax NG namespace (`http://relaxng.org/ns/structure/1.0`) is handled as annotation and the text content is displayed in the annotation window. To activate this behavior, select the ***Use all Relax NG annotations as documentation*** option in the **Annotations** preferences page.

Collecting Annotation from DTDs

For DTD, Oxygen XML Editor defines a custom mechanism for annotations using comments enabled by the ***Prefer DTD comments that start with "doc:" as annotations*** option in the **Annotations** preferences page. The following is an example of a DTD annotation:

```
<!--doc:Description of the element. -->
```

Related Information:

Customizing the Rendering of Elements on page 1248
Annotations for XML Elements and Attributes in Content Completion Assistant on page 1250

Content Completion Helper Views

Information about the current element being edited is also available in various *dockable* views, such as the ***Model*** *view*, ***Attributes*** *view*, ***Elements*** *view*, and ***Entities*** *view*. By default, they are located on the right-hand side of the main editor window. These views, along with the powerful ***Outline*** *view*, provide spatial and insight information about the edited document and the current element. If any particular view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Model View

The **Model** view presents the structure of the currently selected tag, and its documentation, defined as annotation in the schema of the current document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

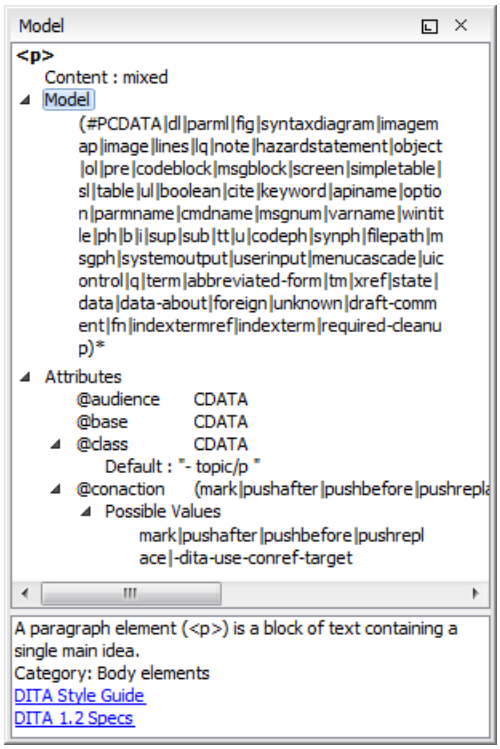


Figure 160: Model View

The **Model** view is comprised of two sections, an element structure panel and an annotations panel.

Element Structure Panel

The element structure panel displays the structure of the currently edited or selected tag in a tree-like format. The information includes the name, model, and attributes of the current tag. The allowed attributes are shown along with imposed restrictions, if any.

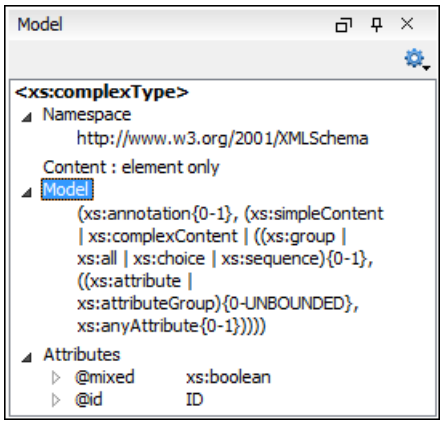


Figure 161: Element Structure Panel

Annotation Panel

The **Annotation** panel displays the annotation information for the currently selected element. This information is collected from the XML schema.

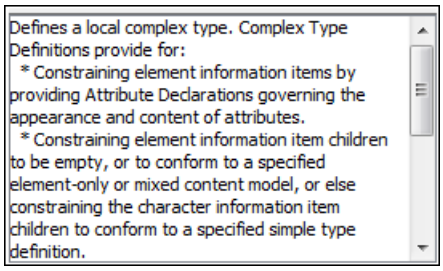


Figure 162: Annotation panel

Attributes View in Author Mode

The **Attributes** view presents all the attributes of the current element determined by the schema of the document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened from the **Window** > **Show View** menu.

You can use this view to edit or add attribute values. The attributes of an element are editable if any one of the following is true:

- The CSS stylesheet associated with the document does not specify a **false** value for the *-oxy-editable* property associated with the element.
- The element is entirely included in a deleted *Track Changes* marker.
- The element is part of a content fragment that is referenced in **Author** mode from another document.

The attributes are rendered differently depending on their state:

- The names of the attributes are rendered with a bold font, and their values with a plain font.
- Default values are rendered with a plain font, painted gray.
- Empty values display the text "[empty]", painted gray.
- Invalid attributes and values are painted red.

To edit the value of the corresponding attribute, double-click a cell in the **Value** column . If the possible values of the attribute are specified as `list` in the schema of the edited document, the **Value** column acts as a combo box that allows you to either select the value from a list or manually enter it.

Note: If the cursor is located inside read-only content, the attribute names and values are faded and you cannot add, edit, or remove values.

You can sort the attributes table by clicking the **Attribute** column header. The table contents can be sorted as follows:

- By attribute name in ascending order.
- By attribute name in descending order.
- Custom order, where the used attributes are displayed at the beginning of the table sorted in ascending order, followed by the rest of the allowed elements sorted in ascending order.

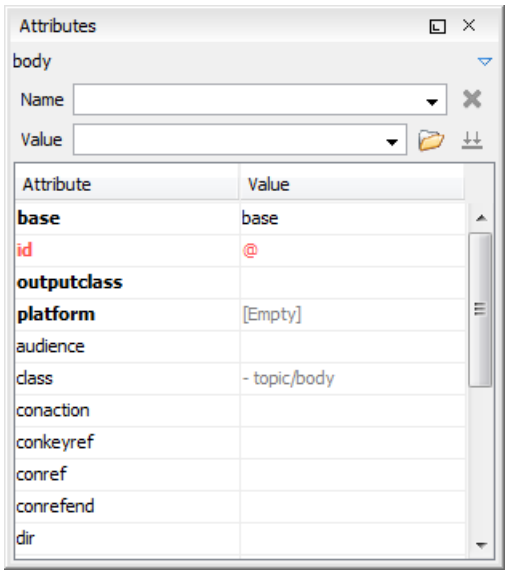


Figure 163: Attributes View

A drop-down list located in the upper part of the view allows you to select the current element or its ancestors.

Expand/Collapse Button

There is an **Expand/** **Collapse** button at the top-right of the view. When expanded, this presents the following additional combo boxes:

Name Combo Box

Use this combo box to select an attribute. The drop-down list displays the list of possible attributes allowed by the schema of the document, as in the **Attributes** view. You can use the **Remove** button to delete an attribute and its value from the selected element.

Value Combo Box

Use this combo box to add, edit, or select the value of an attribute. If the selected attribute has predefined values in the schema, the drop-down list displays those possible values. You can use the **Browse** button to select a URL for the value of an attribute. After you have entered or selected a value, use the **Update** button (or press **Enter**) to add the value to the attribute.

Contextual Menu Actions in the Attributes View

The following actions are available in the contextual menu of the **Attributes** view when editing in **Author** mode:

Set empty value

Specifies the current attribute value as empty.

Remove

Removes the attribute (action available only if the attribute is specified). You can invoke this action by pressing the **(Delete)** or **(Backspace)** keys.

Copy

Copies the `attrName="attrValue"` pair to the clipboard. The `attrValue` can be:

- The value of the attribute.
- The value of the default attribute, if the attribute does not appear in the edited document.
- Empty, if the attribute does not appear in the edited document and has no default value set.

Paste

Depending on the content of the clipboard, the following cases are possible:

- If the clipboard contains an attribute and its value, both of them are introduced in the **Attributes** view. The attribute is selected and its value is changed if they exist in the **Attributes** view.
- If the clipboard contains an attribute name with an empty value, the attribute is introduced in the **Attributes** view and you can start editing it. The attribute is selected and you can start editing it if it exists in the **Attributes** view.
- If the clipboard only contains text, the value of the selected attribute is modified.

In-place Attributes Editor

Oxygen XML Editor includes an in-place attributes editor in **Author** mode. To edit the attributes of an XML element in-place, do one of the following:

- Select an element or place the cursor inside it and then press the **Alt + Enter** keyboard shortcut.
- Double-click any named start tag when the document is edited in one of the following *display modes.*: **Full Tags with Attributes**, **Full Tags**, **Block Tags**, or **Inline Tags**.

This opens an in-place attributes editor that contains the same content as the **Attributes** view. By default, this editor presents the **Name** and **Value** fields, with the list of all the possible attributes collapsed.

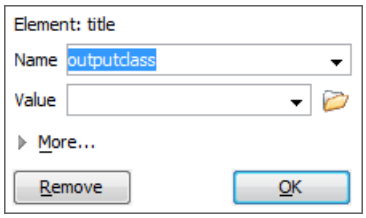


Figure 164: In-place Attributes Editor

Name Combo Box

Use this combo box to select an attribute. The drop-down list displays the list of possible attributes allowed by the schema of the document, as in the **Attributes** view.

Value Combo Box

Use this combo box to add, edit, or select the value of an attribute. If the selected attribute has predefined values in the schema, the drop-down list displays those possible values.

If you click **More** while in the collapsed version, it is expanded to the full version of the in-place attribute editor.

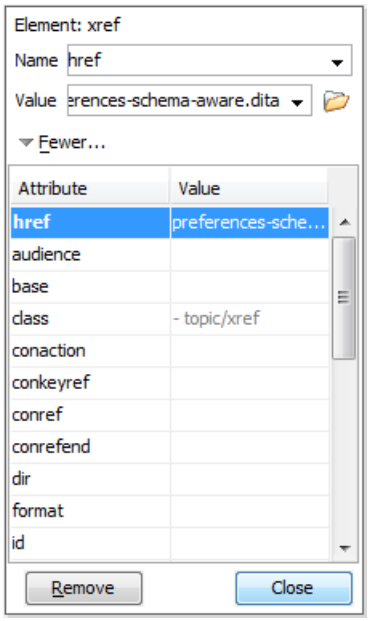


Figure 165: In-place Attributes Editor (Full Version)

The full version includes a table grid, similar to the **Atributes** view, that presents all the attributes for the selected element.

Elements View in Author Mode

The **Elements** view presents a list of all defined elements that are valid at the current cursor position according to the schema associated to the document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

The upper part of the view features a combo box that contains the ordered ancestors of the current element. Selecting a new element in this combo box updates the list of the allowed elements. By default, only the elements that are allowed at the current cursor position are listed. However, if the ***Show only items allowed at cursor position*** *option* is not selected in the ***Views*** *preferences page*, two additional tabs (**Before** and **After**) will be displayed at the bottom of the view and they list elements that are allowed before or after the element at the current cursor position.

Double-clicking any of the listed elements inserts that element into the edited document and its position depends on the tab.

- **Cursor** tab - Double-clicking an element inserts it at the current cursor position.
- **Before** tab - Double-clicking an element inserts it before the element at the cursor position.
- **After** tab - Double-clicking an element inserts it after the element at the cursor position.

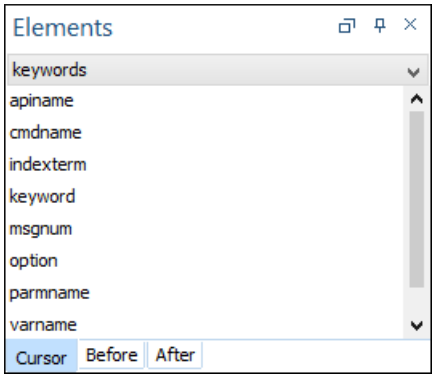


Figure 166: Elements View in Author Mode

Entities View

Entities provide abbreviated entries that can be used in XML files when there is a need of repeatedly inserting certain characters or large blocks of information. An *entity* is defined using the `ENTITY` statement either in the DOCTYPE declaration or in a DTD file associated with the current XML file.

There are three types of entities:

- **Built-in** or **Predefined** - Entities that are part of the predefined XML markup (`<`, `>`, `&`, `'`, `"`).
- **Internal** - Defined in the DOCTYPE declaration header of the current XML.
- **External** - Defined in an external DTD module included in the DTD referenced in the XML DOCTYPE declaration.

Note: If you want to add internal entities, you would need to switch to the Text editing mode and manually modify the DOCTYPE declaration. If you want to add external entities, you need to open the DTD module file and modify it directly.

The **Entities** view displays a list with all entities declared in the current document, as well as built-in ones. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Double-clicking one of the entities will insert it at the current cursor position in the XML document. You can also sort entities by name and value by clicking the column headers.

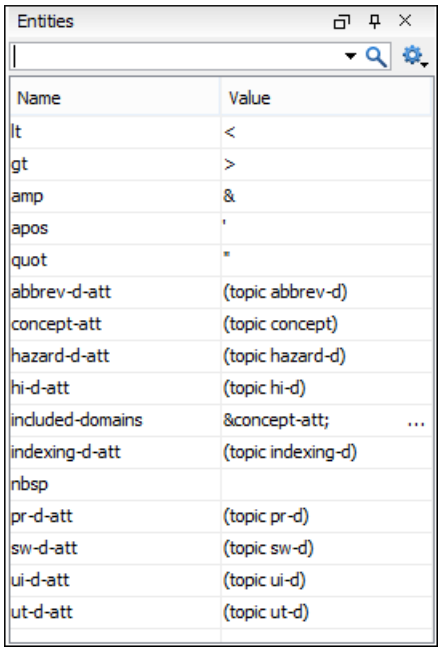


Figure 167: Entities View

The view features a filtering capability that allows you to search an entity by name, value, or both. Also, you can choose to display the internal or external entities.

Note: When entering filters, you can use the *?* and *** wildcards. Also, you can enter multiple filters by separating them with a comma.

Code Templates

Code templates are code fragments that can be inserted quickly at the current editing position . Oxygen XML Editor includes a set of built-in code templates for CSS, LESS, Schematron, XSL, XQuery, and XML Schema document types. You can also *define your own code templates for any type of file and share them with others*.

To get a complete list of available code templates, press **Ctrl + Shift + Space** in **Text** mode. To enter the code template, select it from the list or type its code and press **Enter**. If a shortcut key has been assigned to the code template, you can also use the shortcut key to enter it. Code templates are displayed with a symbol in the content completion list.

When the *Content Completion Assistant* is invoked (**Ctrl + Space (Command + Space on OS X)** in **Text** mode or **Enter** in **Author** mode), it also presents a list of code templates specific to the type of the active editor.

To watch our video demonstration about code templates, go to *https://www.oxygenxml.com/demo/Code_Templates.html*.

Outline View in Author Mode

The **Outline** view in **Author** mode displays a general tag overview of the currently edited XML Document. When you edit a document, the **Outline** view dynamically follows the changes that you make, displaying the node that you modify. This functionality gives you great insight on the location of your modifications in the current document. It also shows the correct hierarchical dependencies between elements. This makes it easy for you to be aware of the document structure and the way element tags are nested.

Outline View Features

The **Outline** view allows you to:

- Quickly navigate through the document by selecting nodes in the **Outline** tree.
- Insert or delete nodes using contextual menu actions.
- Move elements by dragging them to a new position in the tree structure.

- Highlight elements in the editor area. It is synchronized with the editor area, so when you make a selection in the editor area, the corresponding nodes are highlighted in the **Outline** view, and vice versa.
- View document errors, as they are highlighted in the **Outline** view. A tooltip also provides more information about the nature of the error when you hover over the faulted element.

Outline View Interface

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (*, ?) and separate multiple patterns with commas.

It also includes a **Settings** menu in the top-right corner that presents a variety of options to help you filter the view even further.

Drop and Drop Actions in the Outline View

Entire XML elements can be moved or copied in the edited document using only the mouse in the **Outline** view with drag-and-drop operations. Several drag and drop actions are possible:

- If you drag an XML element in the **Outline** view and drop it on another node, then the dragged element will be moved after the drop target element.
- If you hold the mouse pointer over the drop target for a short time before the drop then the drop target element will be expanded first and the dragged element will be moved inside the drop target element after its opening tag.
- You can also drop an element before or after another element if you hold the mouse pointer towards the upper or lower part of the targeted element. A marker will indicate whether the drop will be performed before or after the target element.
- If you hold down the **(Ctrl (Command on OS X))** key after dragging, a copy operation will be performed instead of a move.

Tip: You can select and drag multiple nodes in the **Outline** view when editing in **Author** mode.

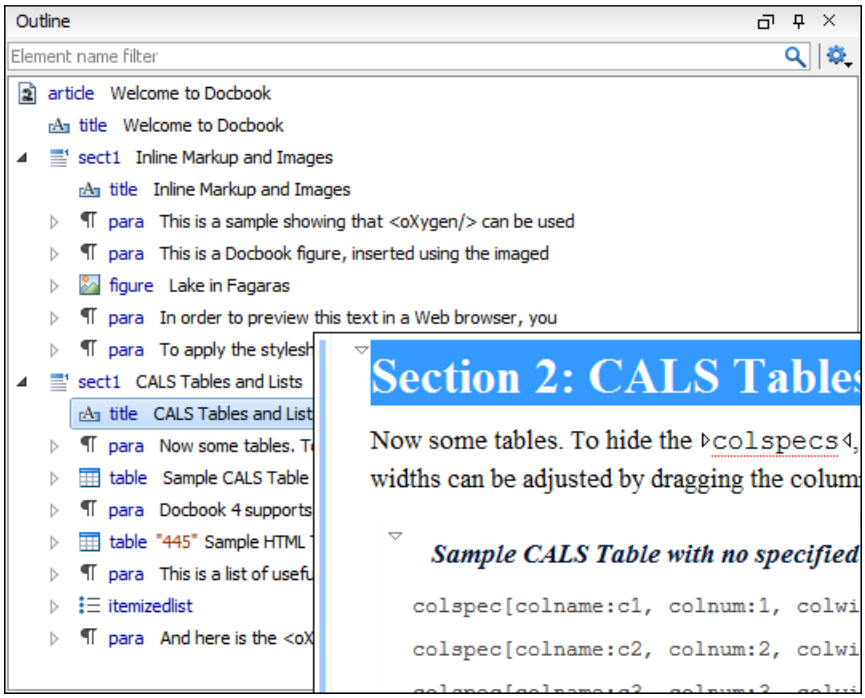


Figure 168: Outline View

Outline View Filters in Author Mode

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (*, ?) and separate multiple patterns with commas.

The following actions are available in the **Settings** menu of the **Outline** view when editing in **Author** mode:

Filter returns exact matches

The text filter of the **Outline** view returns only exact matches.

Flat presentation mode of the filtered results

When active, the application flattens the filtered result elements to a single level.

Show comments and processing instructions

Show/hide comments and processing instructions in the **Outline** view.

Show element name

Show/hide element name.

Show text

Show/hide additional text content for the displayed elements.

Show attributes

Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from *the Outline preferences panel*.

Configure displayed attributes

Displays the *XML Structured Outline preferences page*.

Outline View Contextual Menu Actions in Author Mode

The contextual menu of the **Outline** view in **Author** mode contains the following actions:

Edit Attributes

Displays an *in-place attributes editor* that allows you to edit the attributes of a selected node.

Edit Profiling Attributes

Allows you to change the *profiling attributes* defined on all selected elements.

Append Child

Invokes a content completion list with the names of all the elements that are allowed by the associated schema and inserts your selection as a child of the current element.

Insert Before

Invokes a content completion list with the names of all the elements that are allowed by the associated schema and inserts your selection immediately before the current element, as a sibling.

Insert After

Invokes a content completion list with the names of all the elements that are allowed by the associated schema and inserts your selection immediately after the current element, as a sibling.

Cut, Copy, Paste, Delete editing actions

Executes the typical editing actions on the currently selected elements. The **Cut** and **Copy** operations preserve the styles of the copied content. The **Paste before** and **Paste after** actions allow you to insert a well-formed element before or after the currently selected element. The **Paste as XML** action pastes copied content that is considered to be valid XML, preserving its XML structure.

Toggle Comment

Encloses the currently selected element in an XML comment, if the element is not already commented. If it is already commented, this action will remove the comment.

Rename Element

Invokes a **Rename** dialog box that allows you to rename the currently selected element, siblings with the same name, or all elements with the same name.

Expand More

Expands the structure tree of the currently selected element.

Collapse All

Collapses all of the structure tree of the currently selected node.

Tip: You can copy, cut or delete multiple nodes in the **Outline** by using the contextual menu after selecting multiple nodes in the tree.

Related Information:

Attributes View in Author Mode on page 224

Finding and Replacing Text

You can search for a specific word or string of characters using the following features:

- ***Find/Replace*** *dialog box*
- ***Find/Replace in Files*** *dialog box*
- ***Quick Find*** *toolbar*
- ***Find all Elements*** *dialog box*

Complex search operations may take some time to complete. If a search operation takes more than 5 seconds, you are prompted to decide whether you want to continue the operation or stop it.

Reviewing Documents

Oxygen XML Editor includes a variety of helpful review tools that improve your ability to collaborate with other members of your team, track changes, mark content for various reasons, add comments in your content, and to manage the review features.

Tracking Document Changes

The *Track Changes feature* is a way to keep track of the changes you make in a document. The *Track Changes* feature highlights changes that you make to the content in a document, as well as changes to attributes. Changes can be tracked for insertions and deletions. When the *Track Changes feature is activated*, insertions are rendered in **Author** mode with an underline while deletions are rendered with a strike through.

The *tracked changes* are also displayed in the ***Review*** *view* and you can also choose to present the changes in *callouts* by selecting the ***Track Changes Deletions*** and ***Track Changes Insertions*** options in the ***Callouts*** *preferences page*.

Adding Comments in Documents

You can associate a comment to a selected area of content. Comments can highlight virtually any content from your document, with the exception of *read-only* text. The difference between using comments and *change tracking* is that a comment can be associated to an area of text without modifying or deleting the text.

Comments are presented in *callouts* with persistent highlights and a colored background. The background color is assigned automatically by the application, but it can also be customized from the ***Review*** *preferences page*.

Highlighting Content

Oxygen XML Editor includes a highlighting feature that allows you to create digital markers to emphasise important fragments of your documents. This is especially useful when you want to mark content that needs additional work or the attention of others.

Using the Review View

Oxygen XML Editor includes a ***Review*** *view* that provides a simplified way of monitoring all the insertions, deletions, comments, and highlights in an XML document. This handy tool is especially useful for large teams that need to gather and manage all the edits from all team members who are working on the same project.

The **Review** view is also useful for managing *tracked changes* and comments in a single panel. In this view, the changes and comments are presented in a compact form, in the order they appear in the document, and they are synchronized with the changes and comments in the main editing area.

You can use this view to quickly navigate through changes, accept or reject them, or to view and manage comments or highlights. You can also search for specific changes or comments and it includes some filtering options (for example, you can filter it to only show certain types of edits or to only show edits for a particular author).

Printing Review Information

When you print a document from **Author** mode, whatever review information is shown in the main editing area will be included in the printed output. For example, *tracked changes* will be included and as long as the ***Comments*** *option* is selected in the ***Callouts*** *preferences page*, comment callouts will also be included (same with *tracked change* callouts if their corresponding options are selected in the ***Callouts*** *preferences page*.

Managing Tracked Changes

Oxygen XML Editor includes a *Track Changes feature* that allows you to review changes that you or other authors have made and then accept or reject them. You can also manage the visualization mode of the tracked changes, add comments to changes, and mark them as being done. These actions are easily accessible from contextual menus, the toolbar, or the ***Review*** *view*.

The *Track Changes* feature is also able to keep track of changes you make to attributes in a document and the changes are presented in the ***Review*** *view* and ***Attributes*** *view*.

To watch our video demonstration about the *Track Changes* support, go to *https://www.oxygenxml.com/demo/Change_Tracking.html*.

Types of Tracked Changes

The types of *tracked changes* include:

- Inserting, deleting content (text or elements)
- Drag and drop content (text or elements)
- Cutting or pasting content (text or elements)
- Inserting, deleting, and changing the structure of tables
- Inserting and editing lists and their content
- Inserting and deleting entities
- Inserting and deleting element tags
- Editing attributes
- Performing a **Split** operation
- Performing a **Surround with** operation
- Changes in referenced content (for example, XInclude fragments or DITA conrefs)

Important: If you copy content in **Author** mode that contains *tracked changes*, the changes will automatically be accepted prior to the content being copied to the clipboard. This filtering is performed only if the selection is not entirely inside a tracked change.

Activating the Change Tracking Feature

To activate the *Track Changes* feature for the current document, use any of the following methods:

- Click the **Track Changes** button on the toolbar.
- Select **Track Changes** from the **Review** submenu of the contextual menu in the main editing area in **Author** mode.

- Select **Track Changes** from the **Edit** > **Review** menu.

To activate the *Track Changes* feature globally for all documents that you open in Oxygen XML Editor, change the ***Initial State*** *option to* ***Always On*** in the ***Review*** *preferences page*.

Rendering Tracked Changes in Author Mode

When *Track Changes* is enabled, your modifications are highlighted using a distinctive color. The colors can be customized from the ***Review*** *preferences page*, along with the name of the author and the initial state of the feature when you open a document. Insertions are rendered with an underline while deletions are rendered with a strike through.

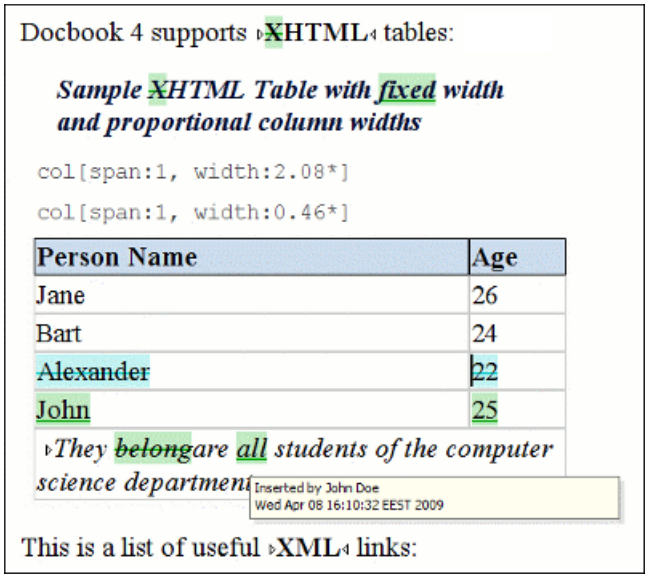


Figure 169: Change Tracking in Author Mode

When hovering over a change a tooltip displays information about the author and modification time.

Change Tracking Contextual Menu Actions

You can right-click any change in **Author** mode to access the following contextual menu actions:

Accept Change(s)

Accepts the *Tracked Change* located at the cursor position or all of the changes in a selection. If you select a part of a *deletion* or *insertion* change, only the selected content is accepted. If you select multiple changes, all of them are accepted. For an *insertion* change, it keeps the inserted text and for a *deletion* change, it removes the content from the document.

Reject Change(s)

Rejects the *Tracked Change* located at the cursor position or all of the changes in a selection. If you select a part of a *deletion* or *insertion* change, only the selected content is rejected. If you select multiple changes, all of them are rejected. For an *insertion* change, it removes the inserted text and for a *deletion* change, it preserves the original content.

Comment Change

Opens a dialog box that allows you to add a comment to an existing *Tracked Change*. The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

Change Tracking Toolbar Actions

By default, the toolbar includes the following actions and options for reviewing or *tracking changes* (similar actions are also available in the **Edit** > **Review** menu and the **Review** submenu of the contextual menu):

Track Changes

Enables or disables the *Track Changes* support for the current document.

Accept Change(s) combo box

This combo box is both a button and a drop-down menu that includes the following actions (when you select an action from the drop-down menu, that action becomes the default action for the combo box button):

- **Accept Change(s) and Move to Next** - Accepts the *Tracked Change* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a *deletion* or *insertion* change, only the selected content is accepted.
- **Accept Change(s)** - Accepts the *Tracked Change* located at the cursor position or all of the changes in a selection.
- **Accept All Changes** - Accepts all *Tracked Changes* in the current document.

Tip: You can *assign shortcut keys* for these actions to make it easier to access them.

Reject Change(s) combo box

This combo box is both a button and a drop-down menu that includes the following actions (when you select an action from the drop-down menu, that action becomes the default action for the combo box button):

- **Reject Change(s) and Move to Next** - Rejects the *Tracked Change* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a *deletion* or *insertion* change, only the selected content is rejected.
- **Reject Change(s)** - Rejects the *Tracked Change* located at the cursor position or all of the changes in a selection.
- **Reject All Changes** - Rejects all *Tracked Changes* in the current document.

Tip: You can *assign shortcut keys* for these actions to make it easier to access them.

Comment Change

Opens a dialog box that allows you to add a comment to an existing *Tracked Change*. The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

Track Changes Visualization Modes Drop-Down Menu

This drop-down menu includes specialized actions that allow you to switch between the following visualization modes:

- **View All Changes/Comments** - This mode is active by default. When you use this mode, all tracked changes are represented in the **Author** mode.
- **View only Changes/Comments by** - Only the tracked changes made by the author you select are presented.
- **View Final** - This mode offers a preview of the document as if all tracked changes (both inserted and deleted) were accepted.
- **View Original** - This mode offers a preview of the document as if all tracked changes (both inserted and deleted) were rejected. If you attempt to edit the document in this mode, the view mode will switch to **View All Changes/Comments**.

Note: If you use **View Final** mode and **View Original** mode, *callouts* are not displayed for comments or changes. To display callouts, use the **View All Changes/Comments** mode.

Highlight

Enables or disables the ***Highlight** tool*. Use the **Highlight** drop-down menu to select a new color.

Add Comment

Inserts a comment at the cursor position. The comment appears in a callout box and a tooltip (when hovering over the change).

Show/Edit Comment

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Manage Reviews

Opens the ***Review*** *view*.

Tracked Change Callouts

You can also choose to display insertion and deletion changes in *callouts* in **Author** mode. By default, tracked changes are not displayed in callouts, but you can change this behavior by selecting the ***Track Changes Deletions*** and ***Track Changes Insertions*** options in the ***Callouts*** *preferences page*. You can also choose to display the actual content of the deletion or insertion.

By displaying the changes in callouts, you then have access to even more actions, such as the ability to reply or mark them as being done. For more information, see *Author Callouts* on page 367.

Tracked Changes in the Review View

The ***Review*** *view* is also useful for managing tracked changes and comments. In this view, the edits are presented in a compact form, in the order they appear in the document and each edit is marked with a type-specific icon. You can use this view to quickly navigate through changes, accept or reject them, or to add and manage comments for the changes. You can also search for specific changes and it includes some filtering options (for example, you can filter it to only show certain types of changes or to only show changes for a particular author).

For more information, see *Review View* on page 371.

Tracked Changes XML Source Code

The changes are stored in the document source code as processing instructions and they do not interfere with validations or transformations. For each change, the author name and the modification time are preserved.

Example - Insertion Change: The following processing instruction is an example of how an *insertion* change is stored in a document:

```
<?oxy_insert_start author="John Doe" timestamp="20090408T164459+0300"?>all<?
oxy_insert_end?>
```

Example - Deletion Change: The following processing instruction is an example of how a *deletion* change is stored in a document:

```
<?oxy_delete author="John Doe" timestamp="20090508T164459+0300" content="belong"?>
```

Related Information:

Managing Comments on page 364
Author Callouts on page 367
Review View on page 371

Tracked Changes Behavior

The behavior of the *Track Changes feature* depends on the context, the type of change, and whether or not it is activated.

Inserting Content

If the *Track Changes* feature is disabled and you insert content, the following behavior is possible:

- Making an insertion in a **Delete** change results in the change being split in two and the content is inserted without being marked as change.
- Making an insertion in an **Insert** change results in the change being split in two and the content is inserted without being marked as change.
- Making an insertion in regular content results in a regular insertion.

If the *Track Changes* feature is enabled and you insert content, the following behavior is possible:

- Making an insertion in a **Delete** change results in the change being split in two and the current inserted content appears marked as an INSERT.
- Making an insertion in an **Insert** change results in the following:
  - If the original insertion was made by another user, the change is split in two and the current inserted content appears marked as an INSERT by the current author.
  - If the original **Insert** change was made by the same user, the change is just expanded to contain the inserted content. The creation time-stamp of the previous insert is preserved.
- If we insert in regular content, the current inserted content appears marked as an **Insert** change.

Surrounding Content

If the *Track Changes* feature is enabled and you surround content in a new XML element, the following behavior is possible:

- Making a surround in a **Delete** change results in nothing happening.
- Making a surround in an **Insert** change results in the following:
  - If the original insertion was made by another user, the change is split in two and the surround operation appears marked as being performed by the current author.
  - If the original **Insert** change was made by the same user, the existing change is just expanded to contain the surrounded content.
- Making a surround in regular content results in the operation being marked as a surround change.

Deleting Characters

If the *Track Changes* feature is disabled and you delete content character by character, the following behavior is possible:

- Deleting content in an existing **Delete** change results in nothing happening.
- Deleting content in an existing **Insert** change results in the content being deleted without being marked as a deletion and the INSERT change shrinks accordingly.
- Deleting in regular content results in a regular deletion.

If the *Track Changes* feature is enabled and you delete content character by character, the following behavior is possible:

- Deleting content in an existing **Delete** change results in the following:
  - If the same author created the **Delete** change, the previous change is marked as deleted by the current author.
  - If another author created the **Delete** change, nothing happens.
- Deleting content in an existing **Insert** change results in the following:
  - If the same author created the **Insert** change, the content is deleted and the **Insert** change shrinks accordingly.
  - If another author created the **Insert** change, the **Insert** change is split in two and the deleted content appears marked as a **Delete** change by the current author.
- Deleting in regular content results in the content being marked as a **Delete** change by the current author.

Deleting Selections of Content

If the *Track Changes* feature is disabled and you delete a selection of content, the following behavior is possible:

- If the selection contains an entire **Delete** change, the change disappears and the content is deleted.
- If the selection intersects with a **Delete** change (starts or ends in one), it results in nothing happening.
- If the selection contains an entire **Insert** change, the change disappears and the content is deleted.
- If the selection intersects with an **Insert** change (starts or ends in one), the **Insert** change is shrunk and the content is deleted.

If the *Track Changes* feature is enabled and you delete a selection of content, the following behavior is possible:

- If the selection contains an entire **Delete** change, the change is considered as rejected and then marked as deleted by the current author, along with the other selected content.
- If the selection intersects a **Delete** change (starts or ends in one), the change is considered as rejected and marked as deleted by the current author, along with the other selected content.
- If the selection contains an entire **Insert** change, the following is possible:
  - If the **Insert** is made by the same author, the change disappears and the content is deleted.
  - If the **Insert** is made by another author, the change is considered as accepted and then marked as deleted by the current author, along with the other selected content.
- If the selection intersects an **Insert** change (starts or ends in one), the **Insert** change shrinks and the part of the **Insert** change that intersects with the selection is deleted.

Deleting Tags

Assuming you are using any of the *Tag Display Modes* other than **No Tags** and the *Track Changes* feature is disabled, the following behavior is possible:

- If you delete a start or end tag, both the start and end tag will be removed, while any content that was inside the element is preserved.

Assuming you are using any of the *Tag Display Modes* other than **No Tags** and the *Track Changes* feature is enabled, the following behavior is possible:

- If you delete a start tag of an *inline element*, both the start and end tag are marked as a **Delete** change by the current author, while any content that was inside the element is preserved.

Copying Content

If the *Track Changes* feature is disabled and you copy content, the following behavior is possible:

- If the copied area contains **Insert** or **Delete** changes (or attribute edits), these are also copied to the clipboard.

If the *Track Changes* feature is enabled and you copy content, the following behavior is possible:

- If the copied area contains **Insert** or **Delete** changes (or attribute edits), these are all accepted in the content of the clipboard (the changes will no longer be in the clipboard).

Pasting Content

If the *Track Changes* feature is disabled and you paste content, the following behavior is possible:

- If the clipboard content contains **Insert** or **Delete** changes (or attribute edits), they will be preserved on paste.

If the *Track Changes* feature is enabled and you paste content, the following behavior is possible:

- If the clipboard content contains **Insert** or **Delete** changes (or attribute edits), all the changes are accepted and then the paste operation proceeds according to the insertion rules.

Tracked Changes Limitations

There are some inherent limitations to the *Change Tracking* feature. These limitations include the following:

- **Limitations to rejected changes** - Recording changes has limitations and there is no guarantee that rejecting all changes will return the document exactly to its original state.

- **Limitations to hierarchical changes** - Recorded changes are not hierarchical, a change cannot contain other changes inside. For example, if you delete an insertion made by another user, then reject the deletion, the information about the author who made the previous insertion is not preserved.
- **Limitations to using certain actions** - Some actions cannot be implemented with the *Track Changes feature* enabled. For example, some of the table-related actions (**Delete Row(s)**, **Delete Column(s)**, **Join Cells**, **Split Cell**) ignore the *Track Changes* feature when executing the action.

Tracked Changes XML Markup

Depending on the type of edits, the following markup appears in the document source code when you activate the *Track Changes feature*:

Edit Type	Processing Instruction Start Marker	Processing Instruction End Marker	Attributes
Insertion	`<?oxy_insert_start?>`	`<?oxy_insert_end?>`	`author`, `timestamp`
Split	`<?oxy_insert_start?>`	`<?oxy_insert_end?>`	`author`, `timestamp`, `type="split"`
Surround	`<?oxy_insert_start?>`	`<?oxy_insert_end?>`	`author`, `timestamp`, `type="surround"`
Deletion	`<?oxy_delete?>`	_	`author`, `timestamp`, `content`
Comment	`<?oxy_comment_start?>`	`<?oxy_comment_end?>`	`author`, `timestamp`, `comment`, `mid`
Attribute Change	`<?oxy_attributes?>`	_	`id`, `type`, `oldValue`, `author`, `timestamp`

If a comment intersects another, the `mid` attribute is used to correctly identify start and end processing instruction markers.

Managing Comments

You can add comments to any selected area of content within XML documents, with the exception of *read-only* content. The difference between using comments and *tracked changes* is that a comment is associated to a selection without modifying or deleting the content.

By default, when you annotate your XML documents, the comments are displayed in the **Author** mode as *callouts* (balloons) and they are rendered with a unique name and background for each user. If comments are not currently displayed in callouts, select the ***Comments** option* in the ***Callouts** preferences page*. Comments are also displayed in the ***Review** view*.

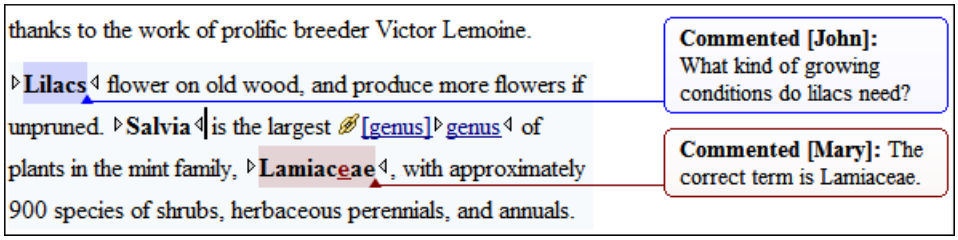


Figure 170: Comments in Author Mode

Managing Comments in the Main Editor

You can insert and manage comments directly in the main editing area in **Author** mode.

Add Comment

To insert a comment at the cursor position or on a specific selection of content, select the **Add Comment** action from the toolbar (or in the **Review** submenu of the contextual menu).

Show/Edit Comments

To edit an existing comment that you have added in the main editing area in **Author** mode, select the **Show/Edit Comments** action from the toolbar (or in the **Review** submenu of the contextual menu). The action opens a dialog box that allows you to see and edit your comment at the cursor position. Note that you cannot edit a comment that was added by another user, so in that case, the dialog box just displays the comment without the possibility of editing it.

Remove Comments

To remove a comment at the cursor position or multiple comments in a selection, select **Remove Comment(s)** from the toolbar (or in the **Review** submenu of the contextual menu).

Tip: When adding or editing a comment, you can use **Enter** to insert line breaks and Oxygen XML Editor will take the line breaks into account when presenting the callout. You can also use **Ctrl + Enter** to accept your changes and close the dialog box.

Managing Comments in Callouts

As long as the ***Comments*** *option* is selected in the ***Callouts*** *preferences page*, comments are also displayed in *callouts*. By displaying the comments in callouts, you then have access to even more actions, such as the ability to reply or mark them as being done. When you right-click a specific comment in its callout, the contextual menu includes the following actions.

Reply

Opens a dialog box that allows you to add a reply to a comment or *Tracked Changes*. When replying to a comment, the dialog box shows the entire conversation in the comment thread, starting with the first comment added in the particular thread, followed by all the replies. After replies are added to a comment thread, they are displayed with an indentation in the callouts and ***Review*** *view*.

Mark as Done

A toggle action that marks or unmarks a comment or comment thread as being done. It is also available for *Tracked Changes* that are displayed in a callout. When a comment or change is marked as done, the callout is grayed out and cannot be edited unless the action is toggled to the unmarked state. The action applies to the particular comment and all of its descendents. This is useful for marking comments or changes that have been addressed, leaving only those that still need some attention.

Show/Edit Comment

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Callouts Options

Select this option to open the ***Callouts*** *preference page* where you can configure various callout options.

Tip: When adding, editing, or replying to a comment, you can use **Enter** to insert line breaks and Oxygen XML Editor will take the line breaks into account when presenting the callout. You can also use **Ctrl + Enter** to accept your changes and close the dialog box.

Managing Comments in the Review View

The ***Review*** *view* is also useful for managing comments. In this view, comments are presented in a compact form, in the order they appear in the document, along with tracked changes. You can also use this view to search for specific comments and it includes some filtering options (for example, you can filter it to only show comments for a particular author). When you right-click a specific comment in the **Review** view, the contextual menu includes the following actions.

Reply

Opens a dialog box that allows you to add a reply to a comment or *Tracked Changes*. When replying to a comment, the dialog box shows the entire conversation in the comment thread, starting with the first comment added in the particular thread, followed by all the replies. After replies are added to a comment thread, they are displayed with an indentation in the callouts and ***Review*** *view*.

Mark as Done

A toggle action that marks or unmarks a comment or comment thread as being done. It is also available for *Tracked Changes* that are displayed in a callout. When a comment or change is marked as done, the callout is grayed out and cannot be edited unless the action is toggled to the unmarked state. The action applies to the particular comment and all of its descendents. This is useful for marking comments or changes that have been addressed, leaving only those that still need some attention.

Show/Edit Comment

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Show only reviews by '<author name>'

Filters the comments to only show comments for the particular author.

Remove all Comments

Removes all comments from the document.

Comments XML Source Code

The comments are stored in the document source code as processing instructions that contain information about the author name and the comment time:

```
<?oxy_comment_start author="John Doe" timestamp="20090508T164459+0300"
 comment="Do not change this content"?>
    Important content
<?oxy_comment_end?>
```

Replies to comments are stored in the document source code as a comment (with information about the author name and time), but with a `parentID` attribute and its value is the same as the `id` value of the parent comment.

```
<?oxy_comment_start author="Tom" timestamp="20160217T102630+0200"
comment="We should not forget about recycling the oil and oil filter!"
parentID="vws_x4l_1v" mid="4"?>
```

Related Information:

Author Callouts on page 367

Review View on page 371

Managing Highlights

Use the **Highlight** tool to mark fragments in your document using various colors. This is especially useful when you want to mark sections that needs additional editing or to draw the attention of others to particular content.

To watch our video demonstration about using the **Highlight** tool, go to *https://www.oxygenxml.com/demo/Highlight_Tool.html*.

Using the Highlight Tool

You can find the **Highlight** action on the main toolbar, in the **Edit** > **Review** menu, or in the **Review** submenu of the contextual menu of a document. You can also choose the color to use for the highlight or choose to **Stop highlighting** from the same menus.

To highlight content, follow these steps:

1. Click the **Highlight** icon on the toolbar.

   Step Result: The highlighting mode is on and the cursor changes to a dedicated symbol.
2. Click the small arrow next to the **Highlight** icon and select the color that you want to use for the highlighting.
3. Select the content you want to highlight. To mark multiple parts of a document, press and hold **Ctrl (Meta on Mac OS)** and select the parts you want to highlight.
4. To exit the highlighting mode, press **Esc**, click the **Highlight** icon, or start editing the document.

To remove highlighting from a document, follow these steps:

1. Either select the text you want to remove highlighting from using your cursor, or press **Ctrl + A (Command + A on OS X)** if you want to select all of the text.
2. Click the small arrow next to the **Highlight** icon and select **No color (erase)**, or right-click the highlighted content and select **Remove highlight(s)**.
3. To exit the highlighting mode, press **Esc**, click the **Highlight** icon, or start editing the document.

Note: Oxygen XML Editor preserves the highlighting of a document between working sessions.

Review View

The ***Review*** *view* is also useful for managing highlights. In this view, the highlights are presented in a compact form, in the order they appear in the document, along with *tracked changes* and comments. The following actions are available in the contextual menu of each highlight in the **Review** view:

Change Color

Allows you to change the color of an existing highlight by selecting the new color from this menu.

Remove Highlight

Removes the selected highlight.

Remove Highlights with the Same Color

Removes the selected highlight and all others that have the same color.

Remove All Highlights

Removes all highlights from the document.

Highlights XML Source Code

The highlights are stored in the document source code as processing instructions that contain information about the color:

```
<?oxy_custom_start type="oxy_content_highlight" color="0,128,255"?>
  The highlights are stored<?oxy_custom_end?>
```

Related Information:

Review View on page 371

Author Callouts

Oxygen XML Editor uses *callouts* to present comments and *tracked change* modifications that you or other members of your team have added to the document.

To watch our video demonstration about the Callouts support, go to *https://www.oxygenxml.com/demo/CalloutsSupport.html*.

Displaying Callouts in Author Mode

The callouts are displayed in the right side of the editing area in **Author** mode. They are decorated with a colored border and also have a colored background. The background color is assigned automatically by the application depending on the user who is editing the document and the type of change, but it can also be customized from

the ***Review** preferences page*. This preferences page allows you to configure the colors for tracked change insertions or deletions, and for comments.

You can also choose to use the same color for all changes of that particular type of change, regardless of who makes the change. To do this, select the **Fixed** option for the particular type of change and choose a color from the color box. If the **Automatic** option is selected, Oxygen XML Editor automatically assigns a color based upon the ***Colors for automatic assignment** list*.

The horizontal line that connects the callouts to their corresponding text fragments has the same color as the border. If this horizontal line is not visible, select the ***Show all connecting lines** option* in the **Callouts** preferences page. If you hover over a callout, it is highlighted and a tooltip is displayed that contains additional information.

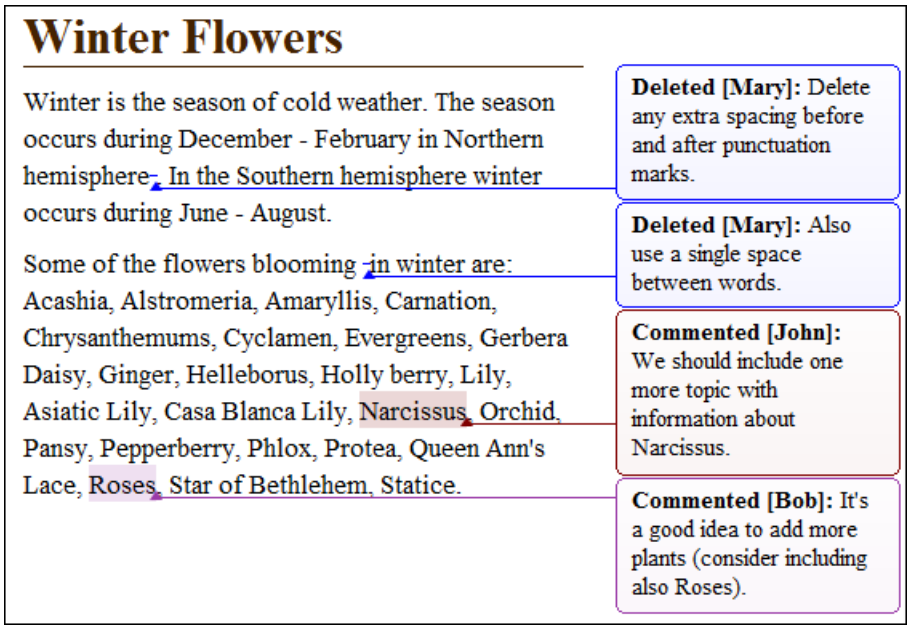


Figure 171: Multiple Author Callouts

Note: Oxygen XML Editor displays callouts only if **View All Changes/Comments** or **View Only Changes/ Comments by** is selected in the **Track Changes Visualization Modes** drop-down menu. Oxygen XML Editor does not display callouts in **View Final** and **View Original** modes.

In some cases, the text you are editing can span into the callouts area. For example, this situation can appear for callouts associated with wide images or *space-preserved elements* that contain long fragments (such as a DITA *codeblock* element or `programlisting` in DocBook). To help you view the text under the covered area, Oxygen XML Editor applies transparency to these callouts. When the cursor is located under a callout, the transparency is enhanced, allowing you to both edit the covered content and access the contextual menu of the editing area.

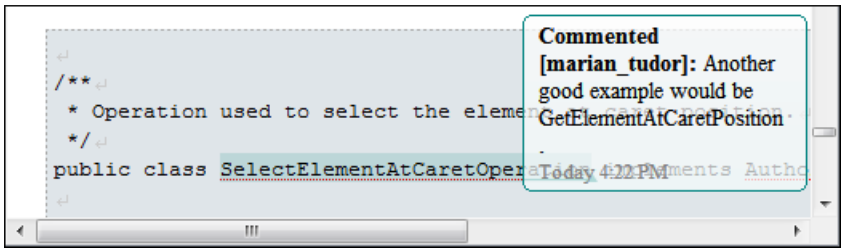


Figure 172: Transparent Callout

Adjusting Callout Width

To display more of the content in all the callouts in the current document, you can adjust the width by dragging the left side of any of the callouts. This will adjust the width for all comments in the current document. When you end the current editing session, the width of all callouts will revert back to the default value, which is the value of the ***Initial Width** option* in the **Callouts** preferences page.

You can also adjust the maximum number of lines to be shown in the callouts using the ***Text Lines Count Limit** option*. Note that this does not limit the number of lines in the actual comment. It only limits the number of lines shown without opening or editing it.

Type of Callouts in Oxygen XML Editor

Oxygen XML Editor uses callouts to display comments and *Tracked Changes* that you associate with fragments of the document you are editing. You can choose which types of edits will be shown in callouts by configuring the options in the ***Callouts** preferences page*. You can choose to enable the following types of review callouts:

- **Comment Callouts** - As long as the ***Comments** option* is selected in the ***Callouts** preferences page*, comments are displayed in callouts. A comment callout contains the name of the author who inserts the callout and the comment itself. You can also select the ***Show review time** option* to include timestamp information in the comment callouts.

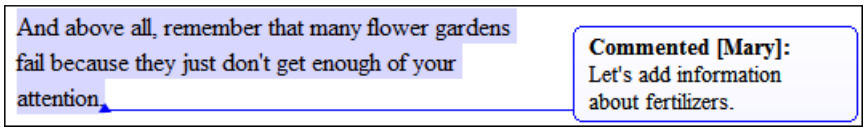
  

  Figure 173: Comment Callouts

  There are several types of comments that can be added in **Author** mode:

  - *Author Review Comments* - Comments that you associate with specific content. To insert this type of comment, select the content and use the **Add Comment** action that is available on the toolbar (or in the **Review** submenu of the contextual menu).
  - *Comments Added to Tracked Changes* - Comments that you add to an already existing *tracked change* insertion or deletion. To insert this type of comment, right-click the change in the main editor or its callout and select **Comment Change**.
  - *Replies to Comments* - You can use this type of comment to create discussion threads. To insert this type of comment, right-click the change in the its callout and select **Reply**. A single callout is presented for a root comment or change and its replies. The replies are displayed with an indentation in the callouts and those that are on the same level are sorted depending on the timestamp.

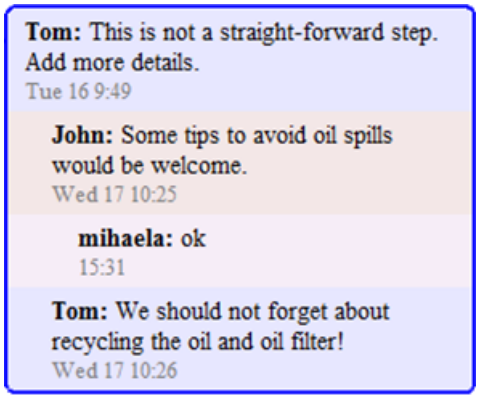
    

    Figure 174: Callout for a Comment with Replies

    Tip: When adding, editing, or replying to a comment, you can use **Enter** to insert line breaks and Oxygen XML Editor will take the line breaks into account when presenting the callout. You can also use **Ctrl + Enter** to accept your changes and close the dialog box.
- **Tracked Change Deletion Callouts** - As long as the ***Track Changes Deletions** option* is selected in the ***Callouts** preferences page*, deletions that are made while the *Track Changes* feature is enabled are displayed in callouts. A deletion callout contains the type of callout (**Deleted**) and the name of the author that made the deletion. You can also select the ***Show deleted content in callout** option* to display the actual deleted content in the callout. Additionally, you can select the ***Show review time** option* to include timestamp information in the deletion callouts.

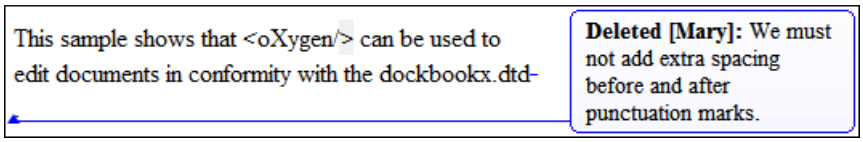


Figure 175: Deletion Callouts

- **Tracked Change Insertion Callouts** - As long as the ***Track Changes Insertions*** *option* is selected in the ***Callouts*** *preferences page*, insertions that are done while the *Track Changes* feature is enabled are displayed in callouts. An insertion callout contains the type of callout (**Inserted**) and the name of the author that inserted the content. You can also select the ***Show inserted content in callout*** *option* to display the actual deleted content in the callout. Additionally, you can select the ***Show review time*** *option* to include timestamp information in the deletion callouts.

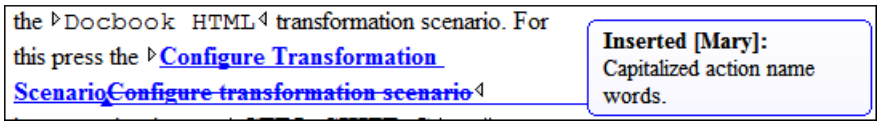


Figure 176: Insertion Callouts

Callout Contextual Menu Actions

Some useful actions are available when the contextual menu is invoked on a callout. The actions depend on the type of callout.

Insertion or Deletion Callout Actions

The following actions are available in the contextual menu of an insertion or deletion callout:

Reply

Opens a dialog box that allows you to add a reply to a comment or *Tracked Changes*. When replying to a comment, the dialog box shows the entire conversation in the comment thread, starting with the first comment added in the particular thread, followed by all the replies. After replies are added to a comment thread, they are displayed with an indentation in the callouts and ***Review*** *view*.

Mark as Done

A toggle action that marks or unmarks a comment or comment thread as being done. It is also available for *Tracked Changes* that are displayed in a callout. When a comment or change is marked as done, the callout is grayed out and cannot be edited unless the action is toggled to the unmarked state. The action applies to the particular comment and all of its descendents. This is useful for marking comments or changes that have been addressed, leaving only those that still need some attention.

Accept Change

Accepts the tracked change, removes the callout, and moves to the next change. For an *insertion* change, it keeps the inserted text and for a *deletion* change, it removes the content from the document.

Reject Change

Rejects the tracked change, removes the callout, and moves to the next change. For an *insertion* change, it removes the inserted text and for a *deletion* change, it preserves the original content.

Comment Change

Opens a dialog box that allows you to add a comment to an existing *Tracked Change*. The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

Edit Reference

If the fragment that contains a callout is a reference, use this option to go to the reference and edit the callout.

Callouts Options

Select this option to open the ***Callouts*** *preference page* where you can configure various callout options.

Comment Callout Actions

The following options are available in the contextual menu of a comment callout:

Reply

Opens a dialog box that allows you to add a reply to a comment or *Tracked Changes*. When replying to a comment, the dialog box shows the entire conversation in the comment thread, starting with the first comment added in the particular thread, followed by all the replies. After replies are added to a comment thread, they are displayed with an indentation in the callouts and ***Review** view*.

Mark as Done

A toggle action that marks or unmarks a comment or comment thread as being done. It is also available for *Tracked Changes* that are displayed in a callout. When a comment or change is marked as done, the callout is grayed out and cannot be edited unless the action is toggled to the unmarked state. The action applies to the particular comment and all of its descendents. This is useful for marking comments or changes that have been addressed, leaving only those that still need some attention.

Show/Edit Comment

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Edit Reference

If the fragment that contains a callout is a reference, use this option to go to the reference and edit the callout.

Callouts Options

Select this option to open the ***Callouts** preference page* where you can configure various callout options.

Printing Callouts

When you print a document from **Author** mode, all callouts that you or other authors have added to the document are printed. For a preview of the document and its callouts, go to **File** > **Print Preview**.

Review View

The ***Review** view* is also useful for managing the information in callouts. In this view, changes and comments are presented in a compact form, in the order they appear in the document, and they are synchronized with the changes in the callouts. You can also search for specific changes or comments and it includes some filtering options (for example, you can filter it to only show certain types of edits or to only show edits for a particular author).

For more information, see *Review View* on page 371.

Related Information:

Managing Tracked Changes on page 358
Managing Comments on page 364
Review View on page 371

Review View

The **Review** view is an independent panel, available both for built-in and custom XML document *frameworks*. It is designed to offer an enhanced way of monitoring all the changes that you make to a document. This means you can view and manage highlights, comments, and *tracked changes* using a single view.

The **Review** view is useful when you are working with documents that contain large number of edits. The edits are presented in a compact form, in the order they appear in the document. Each type of edit is marked with a specific icon. This view and the editing area are synchronized. When you select an edit listed in the **Review** view,

its corresponding fragment of text is highlighted in the editing area and the reverse is also true. For example, when you place the cursor inside an area of text marked as inserted, its corresponding edit is selected in the list.

You can use this view to quickly navigate through changes and it includes some useful hover actions and contextual menu actions to help you manage changes, comments, and highlights. You can also search for specific changes or comments and it includes some filtering options (for example, you can filter it to only show certain types of edits or to only show edits for a particular author).

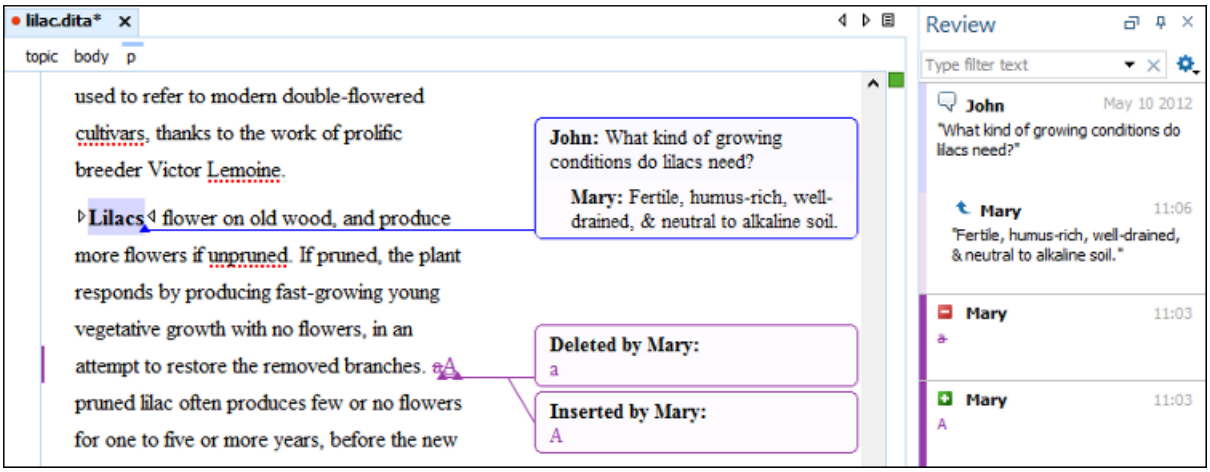


Figure 177: Review View

To watch our video demonstration about the **Review** view, go to *https://www.oxygenxml.com/demo/Review_Panel.html*.

Activating the Review View

To activate the **Review** view, do one of the following:

- Click the **Manage reviews** button on the toolbar.
- Right-click anywhere in a document and select **Review** > **Manage reviews**.
- Open it from the **Window** > **Show View** menu.

Review View Settings

The upper part of the view contains a filtering area that allows you to search for specific edits. Use the small arrow symbol from the right side of the search field to display the search history.

The **Settings** menu includes the following options:

- **Show highlights** - Controls whether or not the **Review** view displays the highlighting in your document.
- **Show comments** - Controls whether or not the **Review** view displays the comments in the document you are editing.
- **Show track changes** - Controls whether or not the **Review** view displays the inserted and deleted content in your document.
- **Show review time** - Displays the time when the edits from the **Review** view were made.
- **Configure review options** - Opens the ***Review*** *preferences page* where you can configure various options for review information.

Hover Actions in the Review View

You can use this view to easily manage changes, highlights, and comments that have been added by you or other users. The following actions are available when you hover over the changes in the **Review** view:

Remove

Available for highlights and comments presented in the **Review** view and it removes the particular highlight or comment from your document and moves to the next change.

Accept

Available for inserted and deleted content presented in the **Review** view and it accepts the particular change in your document and moves to the next change.

Reject

Available for inserted and deleted content presented in the **Review** view and it rejects the particular change in your document and moves to the next change.

Contextual Menu Actions in the Review View

Depending on the type of an edit, the following additional actions are available in the contextual menu of the **Review** view:

Reply

Opens the **Reply** dialog box where you can add a reply to comment or change. The replies are displayed with an indentation in this view.

Mark as Done

This toggles the comment or change as being done and grays it out. You can mark a whole discussion thread as being done by selecting the action on the first (parent) comment in the thread.

Show Comment

Available for comments added by other users and you can use this option to view it in a **Show comment** dialog box.

Edit Comment

Available for comments you have added and you can use this action to edit a comment.

Remove Comment

Use this action to remove the selected comment.

Show only Reviews by '<author name>'

Use this action to filter the edits to only show them for a certain author.

Remove All Comments

Use this action to remove all the comments that appear in the edited document.

Change Color

Available for highlights and it opens a palette that allows you to choose a new color for the highlighted content.

Remove Highlight

Use this action to remove the selected highlight.

Remove Highlights with the Same Color

Use this action to remove all the highlights with the same color from the entire document.

Remove All Highlights

Use this action to remove all the highlights in your document.

Accept Change

Accepts the selected change and moves to the next change.

Reject change

Rejects the selected change and moves to the next change.

Comment change

Available for insertions or deletions and you can use this option to add a comment for the particular change.

Accept all changes

Accepts all the changes in the current document.

Reject all changes

Rejects all the changes in the current document.

Related Information:

Managing Tracked Changes on page 358
Managing Comments on page 364
Managing Highlights on page 366
Author Callouts on page 367

Profiling and Conditional Text

Profiling text is a way to mark blocks of text meant to appear in some renditions of the document but not in others. Conditional text differs from one variant of the document to another, while unconditional text appear in all document versions. For example, you can mark a section of a document that is to be included in a manual to be designated for *expert* users and another section for *novice* users, while unmarked sections are included in all renditions.

Profiling Attributes and Condition Sets

Oxygen XML Editor allows you to define values for the profiling attributes and they can be easily managed to filter content in the published output. You can switch between profile sets to see how the edited content looks like before publishing. You can also conditionally profile parts of a document so that certain parts are displayed when certain profiling conditions are set. You can even customize the colors and styling of how the profiling is displayed in **Author** mode.

You can use profiling and conditional text to help you create documentation for multiple output scenarios, including:

- Multiple outputs for a series of similar products.
- Multiple outputs for various releases of a product.
- Multiple outputs for various audiences.

This feature helps to reduce the effort for updating and translating your content and provides an easy way to customize the output for various audiences.

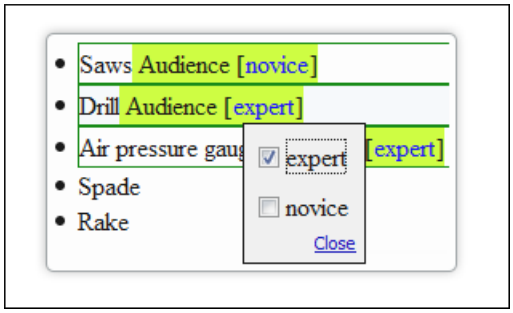


Figure 178: Example: Profiling Content

Oxygen XML Editor includes a preconfigured set of profiling attribute values for some of the most popular document types. These attributes can be redefined to match your specific needs in the ***Profiling/Conditional Text*** *preferences page*. You can also define your own profiling attributes and condition sets for each document type (*framework*) and your profiling configuration can be shared amongst content authors through the project file.

For information about creating and editing profiling attributes, see *Creating and Editing Profiling Attributes* on page 375 (for information about sharing them, see *Sharing Profiling Attribute Configurations* on page 377).

For information about creating and editing condition sets, see *Creating and Editing Profiling Condition Sets* on page 379 (for information about sharing them, see *Sharing Condition Set Configurations* on page 380).

Related Information:

Customizing Elements that Wrap Profiled Content on page 1215

Creating and Editing Profiling Attributes

Oxygen XML Editor includes support for defining your own profiling attributes, or modifying existing ones, for each particular document type (*framework*). You can then apply the profiling attributes to content in **Author** mode to see how the profiling will affect the output.

Create Profiling Attributes

To create custom profiling attributes for a specific document type, follow these steps:

1. Make sure the attribute is already defined in the document DTD or schema before continuing with the procedure.
2. *Open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Edit modes** > **Author** > **Profiling/ Conditional Text** .
3. In the ***Profiling Attributes** section*, press the **New** button.

   Step Result: The **Profiling Attribute** configuration dialog box is opened.

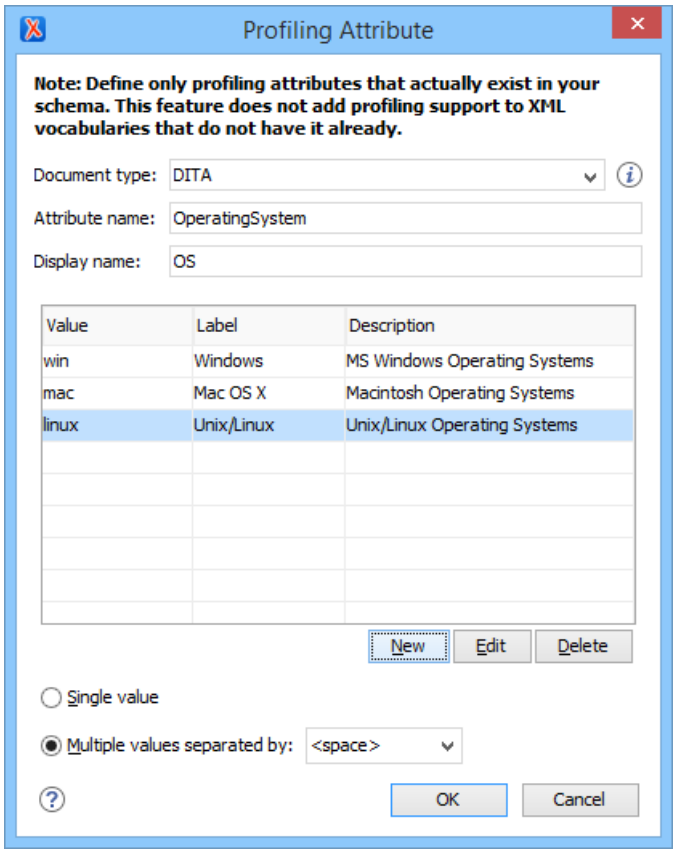


Figure 179: Profiling Attribute Dialog Box

4. Configure your profiling attributes as desired. The following options are available in this dialog box:

   Document type

   Select the document type (*framework*) for which you have defined profiling attributes.

   Tip: You can use the `*` or ? wildcards in this combo box. For example, `DITA*` would match any document type that starts with *"DITA"*. You can also specify multiple document types by using commas to separate them.

   Attribute name

   The name of the new profiling attribute.

   Display name

   This optional field is used for descriptive rendering in profiling dialog boxes.

Attribute Values Table

This table displays the values for the profiling attribute and allows you to configure them by using the following buttons at the bottom of the table:

New

Opens a dialog box that allows you to insert a new value. The fields that can be configured in this dialog box correspond to the columns in the table and are as follows:

- **Value** - The attribute value.
- **Label** - You can specify a label for the attribute value that will be rendered as its name in various components in **Author** mode (***Edit Profiling Attributes*** *dialog box*, ***Condition Set*** *dialog box*, ***Profiling*** *tab in the* ***Edit Properties*** *dialog box*, ***DITA Maps Manager***). If the **Label** is not specified, the **Value** will be used as its rendered name.
- **Description** - A description for the attribute value that will be displayed in this table.

Edit

Use this button or double-click an attribute value to modify it.

Delete

Removes the selected attributed value.

Single value

Select this option if you want the attribute to only accept a single value.

Multiple values separated by

Select this option if you want the attribute to accept multiple values, and you can choose the type of delimiter to use. You can choose between *space*, *comma*, and *semicolon*, or you can enter a custom delimiter in the text field. A custom delimiter must be supported by the specified document type. For example, the DITA document type only accepts spaces as delimiters for attribute values.

5. Click **OK** to confirm your selections and close the **Profiling Attributes** configuration dialog box.
6. Click **Apply** to save the profiling attribute.

Editing Existing Profiling Attributes

To modify an existing profiling attribute or its values, follow these steps:

1. *Open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Editor** > **Edit modes** > **Author** > **Profiling/ Conditional Text**.
2. In the ***Profiling Attributes*** *section*, press the **Edit** button to modify an existing condition set (you can also use **Delete** button to remove a profiling attribute or the **Up** and **Down** buttons to change their priority).

   Step Result: If you use the **Edit** button, the **Profiling Attributes** configuration dialog box is opened:
3. Modify your profiling attribute as desired.
4. To add or modify attribute values, use the **New**, **Edit**, or **Delete** buttons under the attribute values table.
5. Click **OK** to confirm your selections and close the **Profiling Attributes** configuration dialog box.
6. Click **Apply** to save your modifications.

Adding or Editing Profiling Attribute Values

There are several ways to add values to existing profiling attributes.

- Use the procedure in the *Editing Existing Profiling Attributes* section to edit an existing attribute and use the **Profiling Attribute** configuration dialog box to add, edit, or delete values for existing profiling attributes.
- You can add values directly to the existing profiling attributes in a document using the *In-Place Attributes Editor* in **Author** mode, the ***Attributes*** *view*, or in the source code in **Text** mode. However, this just adds them to the document and does not change the conditional text configuration. If you invoke the **Edit Profiling Attributes** action (from the contextual menu in **Author** mode) on the new value, the **Profiling Values Conflict** dialog box will appear and it includes an **Add these values to the configuration** action that will automatically add the new value to the particular profiling attribute. It also includes an **Edit the configuration** action that opens the ***Profiling / Conditional Text*** *preferences page* where you can edit the profiling configuration.

Note: If the ***Allow contributing extra profiling attribute values*** option is not selected in the **Profiling / Conditional Text** preferences page, the **Profiling Values Conflict** dialog box will never appear, so this second method will not be possible.

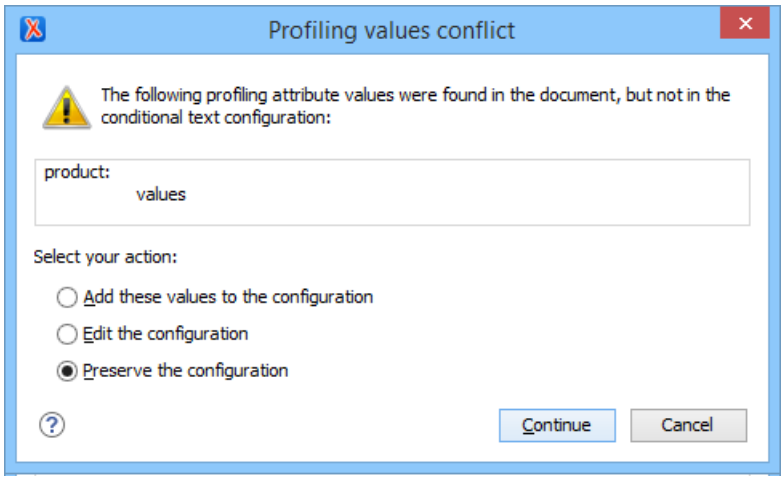


Figure 180: Profiling Values Conflict Dialog Box

Sharing Profiling Attribute Configurations

Your profiling configuration can be shared with other users through a project file. If you select ***Project Options*** at the bottom of the **Profiling/Conditional Text** preferences page, your configuration is stored in the project file and can be shared with others. For instance, if your project file is saved on a version control system (such as SVN, CVS, or Source Safe) or in a shared folder, your team will have the same option configuration that you stored in the project file.

For more information about sharing project files, see *Sharing a Project - Team Collaboration* on page 284.

Related Information:

Apply Profiling Attributes on page 377
Creating and Editing Profiling Condition Sets on page 379
Apply Profiling Condition Sets on page 380
Showing and Filtering Profiled Content in Author Mode on page 382

Apply Profiling Attributes

Profiling attributes are applied on element nodes. You can apply profiling attributes on a text fragment (it will automatically be wrapped into a phrase-type element), on a single element, or on multiple elements in the same time. If there is no selection in your document, the profiling attributes are applied on the element at the cursor position.

To apply a profiling attribute to content in **Author** mode, follow these steps:

1. To apply a profiling attribute to content in **Author** mode, highlight the content and select **Edit Profiling Attributes** from the contextual menu. To profile an entire element, position the cursor inside the element, right-click, and select **Edit Profiling Attributes** (you can also right-click the element in the *breadcrumb* or ***Outline*** view).

   Step Result: The **Edit Profiling Attributes** dialog box is displayed and shows all the profiling attributes and their values, as defined for the particular document type (*framework*). If you have a large list of profiling attributes, you can use the text filter field to search for attributes or values, and you can expand or collapse attributes by using the **Expand All/** **Collapse All** buttons to the right of the text filter or the arrow button to the left of the profiling attribute name.

   The attributes and values that appear in the dialog box are determined as follows:

   - If you have *defined profiling attribute values* for the DITA document type in the ***Profiling/Conditional Text preferences page*** and you store them at *project-level*, those values are displayed in the dialog box.

- If you have *defined profiling attribute values* for the DITA document type in the ***Profiling/Conditional Text preferences page*** and you store them at *global-level*, those values are displayed in the dialog box.
- Otherwise, a generic default set of profiling attributes and values are available.

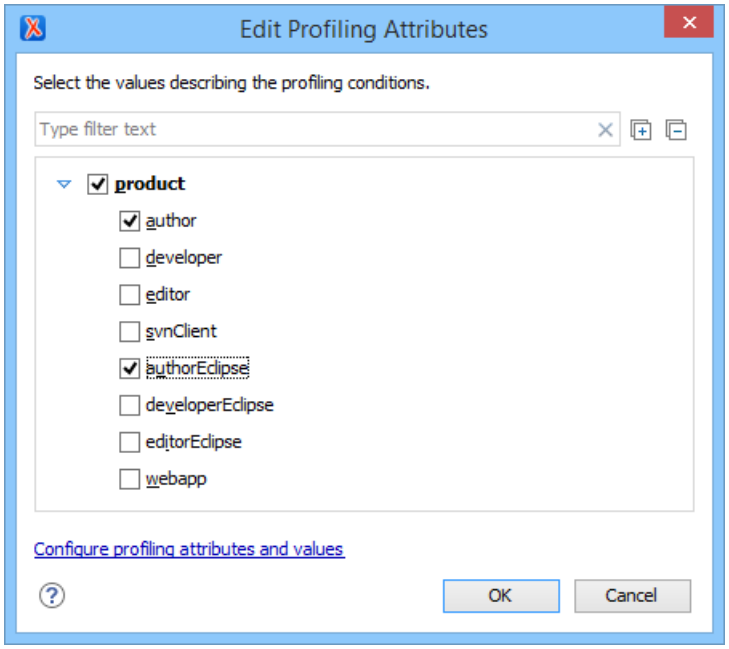


Figure 181: Edit Profiling Attributes Dialog Box

2. In the **Edit Profiling Attributes** dialog box, select the checkboxes that correspond to the attribute values you want to apply on the *document fragment*.
3. Click **OK** to finish the profiling configuration.

   Result: The attribute names and values selected in the **Edit Profiling Attributes** dialog box are set on the elements contained in the profiled fragment. If you only select a fragment of content (rather than the entire element), this fragment is wrapped in phrase-type elements in which the profiling attributes are set.

   If the ***Show Profiling Attributes*** *option* (available in the **Profiling / Conditional Text** toolbar menu) is selected, a green border is painted around profiled text in the **Author** mode and all profiling attributes set on the current element are listed at the end of the highlighted block. To edit the attributes of a profiled fragment, click one of the listed attribute values. A form control pops up and allows you to add or remove attribute values.

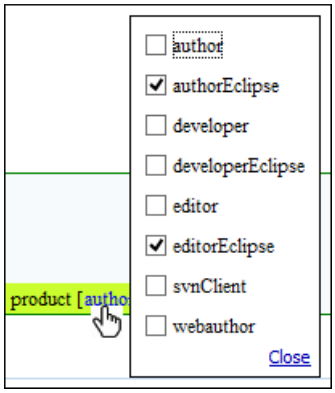


Figure 182: Profiling Attribute Value Form Control Pop Up

Related Information:

Creating and Editing Profiling Attributes on page 375
Creating and Editing Profiling Condition Sets on page 379
Apply Profiling Condition Sets on page 380
Showing and Filtering Profiled Content in Author Mode on page 382

Creating and Editing Profiling Condition Sets

Multiple profiling attributes can be aggregated into a profiling condition set that allows you to apply more complex filters on the document content. A *Profiling Condition Set* is a very powerful and convenient tool that can be used to preview the content that goes into the published output. For example, an installation manual available in both Windows and Linux variants can be profiled to highlight only the Linux procedures for more advanced users.

Create Profiling Condition Sets

To create a new profiling condition set, follow these steps:

1. *Open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Edit modes** > **Author** > **Profiling/ Conditional Text**.
2. In the ***Profiling Condition Sets** section*, press the **New** button.

   Step Result: The **Condition Set** configuration dialog box is opened.

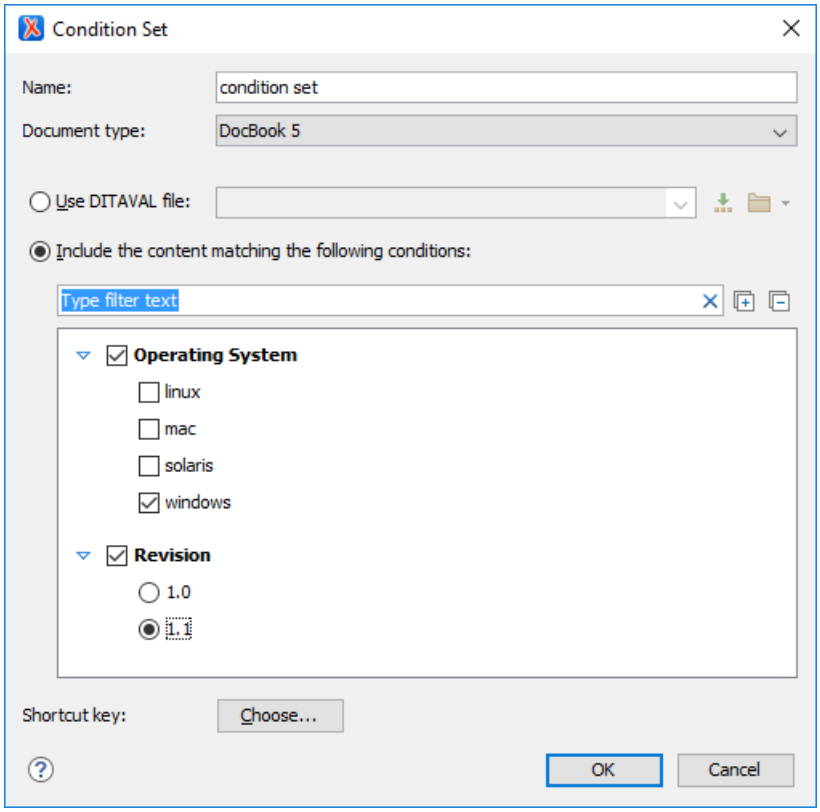


Figure 183: Condition Set Dialog Box

3. Configure your condition set as desired. The following options are available in this dialog box:

   Name

   The name of the new condition set.

   Document type

   Select the document type (*framework*) for which you have defined profiling attributes.

   Use DITAVAL file

   For DITA projects, select this option if you want the *Profiling Condition Set* to reference a *DITAVAL file*. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.

   Include the content matching the following conditions

   You can select this option to define the combination of attribute values for your condition set by selecting the appropriate checkboxes for the values you want to be included in this particular condition set. If you have defined a lot of profiling attributes, you can use the **filter** text field to search for specific conditions.

Shortcut key

You can click the **Choose** button to open a dialog box that allows you to define a shortcut key for this particular condition set. You can then use that shortcut key anytime you want to select this condition set to filter content.

4. Click **OK** to confirm your selections and close the **Condition Set** configuration dialog box.
5. Click **Apply** to save the condition set. All saved profiling condition sets are available in the ***Profiling / Conditional Text*** *toolbar drop-down menu*.

Editing Existing Profiling Condition Sets

To modify an existing profiling condition set, follow these steps:

1. *Open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Editor** > **Edit modes** > **Author** > **Profiling/ Conditional Text**.
2. In the ***Profiling Condition Sets*** *section*, press the **Edit** button to modify an existing condition set (you can also use **Delete** button to remove a condition set or the **Up** and **Down** buttons to change their priority).

   Step Result: If you use the **Edit** button, the **Condition Set** configuration dialog box is opened:
3. Modify your condition set as desired.
4. Click **OK** to confirm your selections and close the **Condition Set** configuration dialog box.
5. Click **Apply** to save your modifications.

Sharing Condition Set Configurations

Your condition set configuration can be shared with other users through a project file. If you select ***Project Options*** at the bottom of the **Profiling/Conditional Text** preferences page, your configuration is stored in the project file and can be shared with others. For instance, if your project file is saved on a version control system (such as SVN, CVS, or Source Safe) or in a shared folder, your team will have the same option configuration that you stored in the project file.

For more information about sharing project files, see *Sharing a Project - Team Collaboration* on page 284.

Related Information:

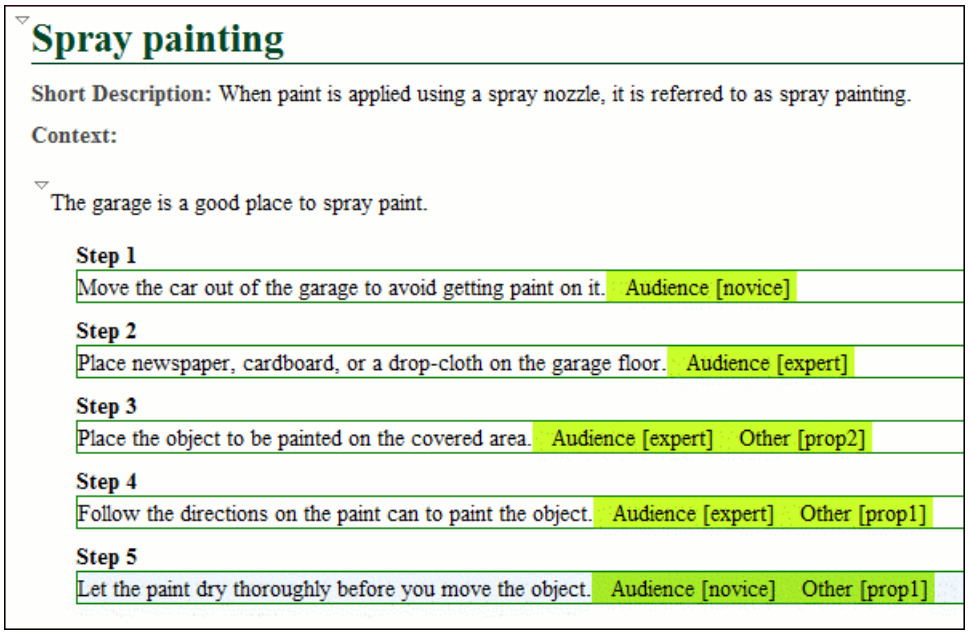
Apply Profiling Condition Sets on page 380
Creating and Editing Profiling Attributes on page 375
Apply Profiling Attributes on page 377
Showing and Filtering Profiled Content in Author Mode on page 382

Apply Profiling Condition Sets

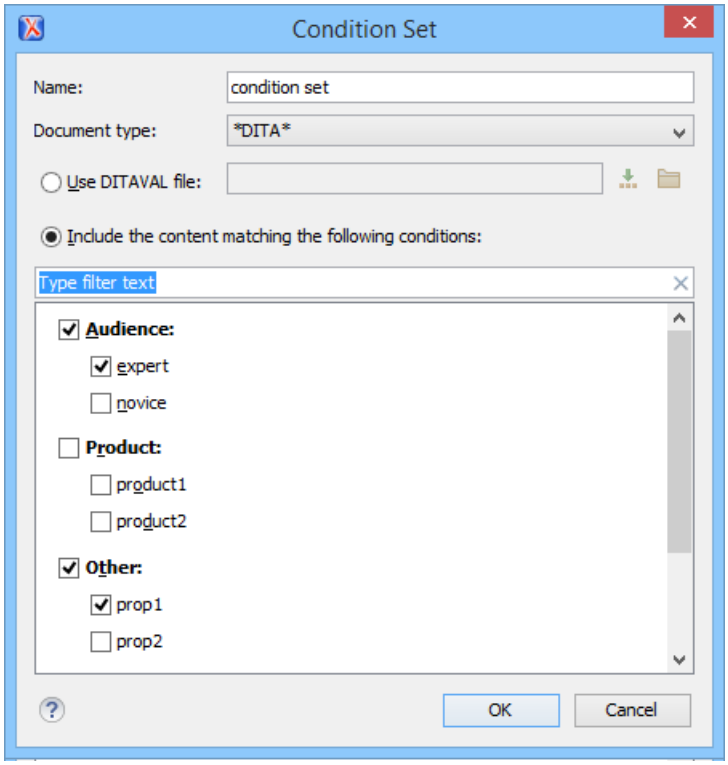
All defined *Profiling Condition Sets* are available as shortcuts in the ***Profiling / Conditional Text*** *toolbar menu*. Select a menu entry to apply the condition set. The filtered content is then grayed-out in the **Author** mode, ***Outline*** *view*, and ***DITA Maps Manager*** *view* (for DITA documents). An element is filtered-out when one of its attributes is part of the condition set and its value does not match any of the value covered by the condition set.

EXAMPLE:

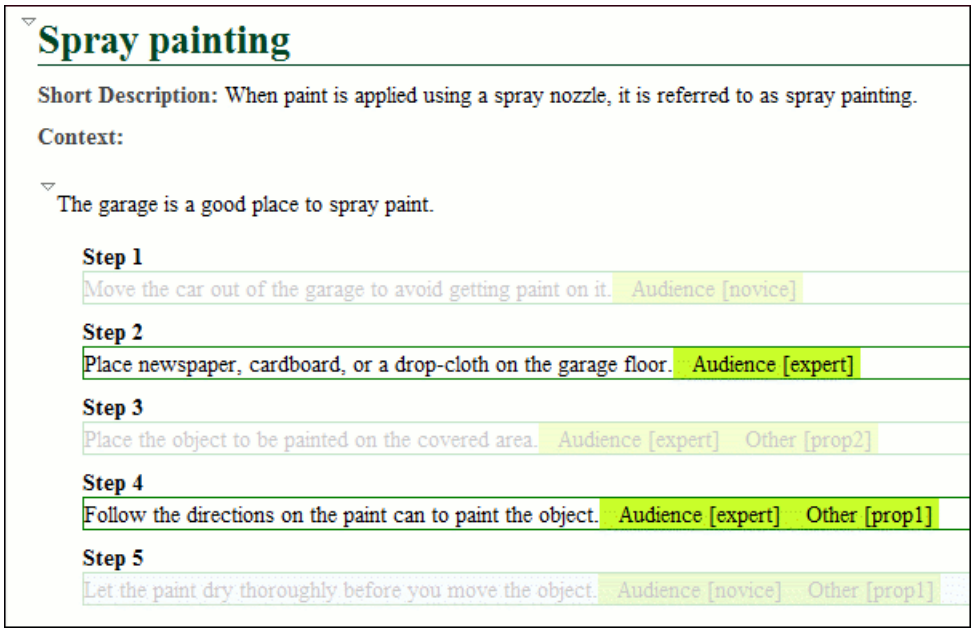
Suppose that you have the following document:



If you apply the following condition set, it means that you want to filter out the content to only include content written for an expert audience and content that has the *Other* attribute value of *prop1*.



This is how the document looks after you apply the condition set:



Related Information:

Creating and Editing Profiling Condition Sets on page 379
Creating and Editing Profiling Attributes on page 375
Apply Profiling Attributes on page 377
Showing and Filtering Profiled Content in Author Mode on page 382

Showing and Filtering Profiled Content in Author Mode

You can visualize the effects of profiled content in **Author** mode by using the options in the **Profiling/ Conditional Text** drop-down menu that is located on toolbar. This drop-down menu includes the following filtering options:

Show Profiling Colors and Styles

Select this option to show colors and styles for profiled content in **Author** mode. You can configure the colors and styles or specify whether or not this option is selected by default in the ***Profiling/Conditional Text*** *>* ***Colors and Styles*** *preferences page*.

Show Profiling Attributes

Select this option to display the values of the profiling attributes at the end of profiled content in **Author** mode. You can specify whether or not this option is selected by default in the ***Profiling/Conditional Text*** *>* ***Attributes Rendering*** *preferences page*.

Show Excluded Content

Controls whether the content filtered out by a particular condition set is hidden or grayed-out in **Author** mode and the ***Outline*** view. When this option is selected and a *condition set is selected in this drop-down menu*, the filtered content is grayed-out. If this option is not selected and a *condition set is selected in this drop-down menu*, the filtered content is hidden. You can specify whether or not this option is selected by default in the ***Profiling/Conditional Text*** *preferences page*.

Choose Condition Set (Available if more than 15 condition sets are defined)

This option is available if you have more than 15 conditions sets defined. It opens a dialog box that makes it easier to find and select condition sets that are not displayed in this drop-down menu.

List of Defined Condition Sets

Up to 15 defined condition sets are listed and you can toggle each one of them on to filter the content in **Author** mode to only show content that will appear in the output for that particular condition set. If there are more than 15 defined condition sets, the rest of them can be accessed in the **More** submenu or by using the ***Choose Condition Set*** *option* to access a dialog box that presents all of them.

Profiling Settings

Opens the ***Profiling/Conditional Text*** *preferences page* where you can add and edit profiling attributes and condition sets.

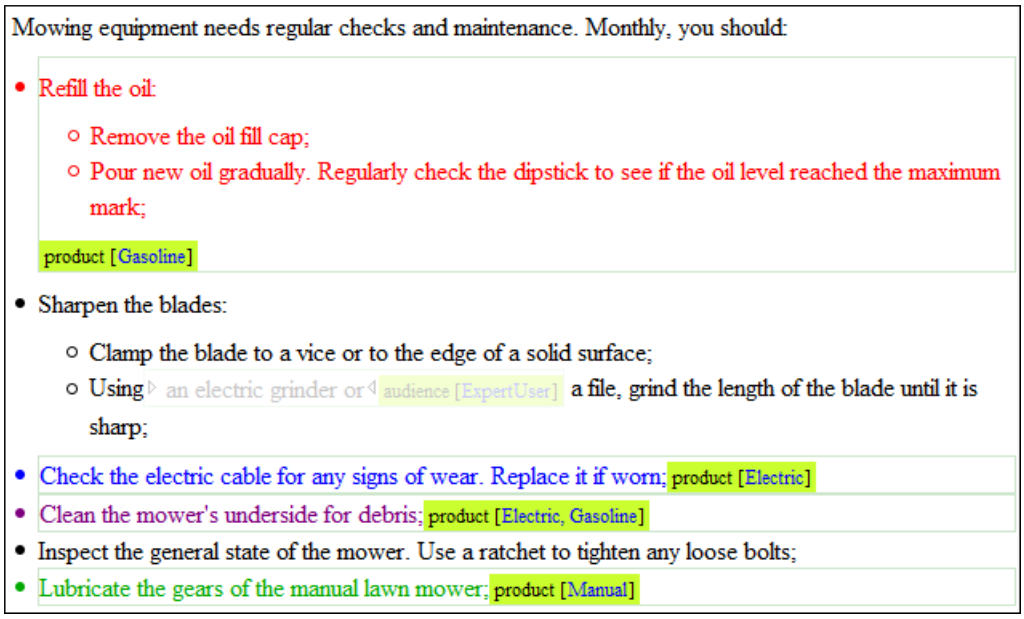


Figure 184: Example: Filtering Profiled Content in Author Mode

If the ***Show Profiling Attributes*** *option* is selected, a green border is painted around profiled text in the **Author** mode. Also, all profiling attributes set on the current element are listed at the end of the highlighted block and in its tooltip message. To edit the attributes of a profiled fragment, click one of the listed attribute values. A form control pops up and allows you to add or remove attribute values.

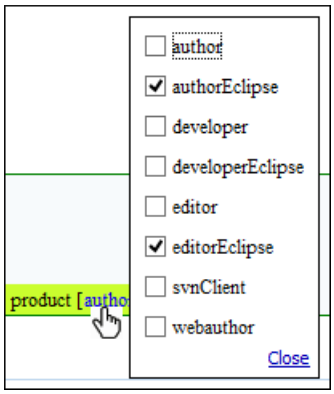


Figure 185: Profiling Attribute Value Form Control Pop Up

Related Information:

Creating and Editing Profiling Attributes on page 375
Apply Profiling Attributes on page 377
Creating and Editing Profiling Condition Sets on page 379
Apply Profiling Condition Sets on page 380

Customizing Colors and Styles for Rendering Profiling in Author Mode

By applying profiling colors and styles, you can mark profiled content in **Author** mode so that you can instantly spot differences between multiple variants of the output. This allows you to preview the content that will go into the published output. The excluded text is grayed-out or hidden in **Author** mode, allowing you to easily recognize the differences.

Mowing equipment needs regular checks and maintenance. Monthly, you should:

- Refill the oil:
  - Remove the oil fill cap;
  - Pour new oil gradually. Regularly check the dipstick to see if the oil level reached the maximum mark;

  product [Gasoline]
- Sharpen the blades:
  - Clamp the blade to a vice or to the edge of a solid surface;
  - Using an electric grinder or audience [ExpertUser] a file, grind the length of the blade until it is sharp;
- Check the electric cable for any signs of wear. Replace it if worn; product [Electric]
- Clean the mower's underside for debris; product [Electric, Gasoline]
- Inspect the general state of the mower. Use a ratchet to tighten any loose bolts;
- Lubricate the gears of the manual lawn mower; product [Manual]

Figure 186: Example: Profiling Colors and Styles

Choosing the right style for a specific profiling attribute is a matter of personal taste, but be aware of the following:

- If the same block of text is profiled with two or more profiling attributes, their associated styles combine. Depending on the styling, this might result in an excessively styled content that may prove difficult to read or work with.
- It is recommended that you only profile the differences. There is no need to profile common content, since excessive profiling can visually pollute the document.
- A mnemonic associated with a style will help you instantly spot differences in the types of content.

Styling Profiling Attribute Values

To set colors and styles for profiling attribute values, follow these steps:

1. Select the ***Show Profiling Colors and Styles*** *option* from the **Profiling / Conditional Text** toolbar drop-down menu.
2. Select ***Profiling Settings*** from the **Profiling / Conditional Text** toolbar drop-down menu. This is a shortcut to the ***Profiling/Conditional Text*** *preferences page*.
3. Go to the ***Colors and Styles*** *preferences page* to configure the colors and styling for the profiling attributes.
4. Go to the ***Attributes Rendering*** *preferences page* to configure how you want the profiling attributes to appear in Oxygen XML Editor.

Result: The styling is now applied in the **Author** editing mode and the ***Outline*** *view*. Also, to help you more easily identify the profiling you want to apply in the current context, the styling is applied in the ***Edit Profiling Attributes*** *dialog box* and in the inline form control pop up that allows you to quickly set the profiling attributes.

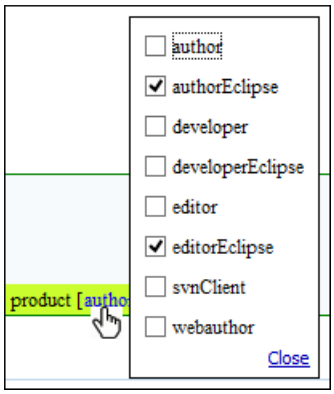


Figure 187: Profiling Attribute Value Form Control Pop Up

Related Information:

Creating and Editing Profiling Attributes on page 375
Apply Profiling Attributes on page 377
Creating and Editing Profiling Condition Sets on page 379
Apply Profiling Condition Sets on page 380
Showing and Filtering Profiled Content in Author Mode on page 382

Adding Tables in Author Mode

You can use the **Insert Table** action on the toolbar or from the contextual menu to add a table in various *frameworks* (DITA, DocBook, TEI, and XHTML). This opens the **Insert Table** dialog box. Each *framework* has a different set of options that are available in this dialog box for configuring the properties of the tables. In all cases, Oxygen XML Editor includes some general editing actions for configuring tables in **Author** mode.

This section explains those general actions and the various configuration options and layouts for tables that are inserted in the most commonly used document types.

Editing Tables in Author Mode

Oxygen XML Editor provides support for editing data in a tabular form. A variety of features and operations are available for editing tables in **Author** mode and they include the following:

Adjusting Column Width

To adjust the width of a column or table, drag the border of the column or table. The changes you make to a table are committed into the source document. You can also manage table width and column width specifications from the source document, and some types of tables include a **colspecs** section that appears above the table in **Author** mode that allows you to easily configure some column specifications (such as column width). These column width specifications are supported in fixed, dynamic, and proportional dimensions. The predefined DITA, DocBook, and XHTML *frameworks* support this feature. The layout of the tables for these document types takes into account the table width and the column width specifications particular to them.

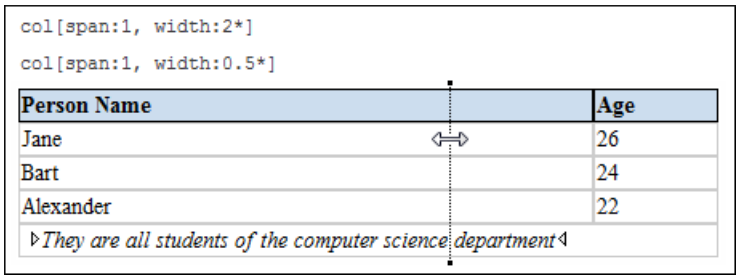


Figure 188: Resizing a Table Column in Author Mode

Selecting Columns and Rows

To select a row or a column of a table, place the mouse cursor above the column or in front of the row you want to select, then click. When hovering the mouse cursor in front of rows or above column headers, the cursor changes to for row selection and to for column selection and that specific row or column is highlighted.

You can use the **Ctrl** and **Shift** keys to select multiple rows.

Selecting Cells

To select a cell in a table, press and hold the **Ctrl** key and click anywhere inside the cell. You can also use the **Ctrl** and **Shift** keys to select multiple cells or to deselect cells from a selection. Alternatively, you can click the left corner of a cell (right corner if you are editing a *RTL document*) to select it. The cursor changes to when you hover over the corner of the cell.

You can also select multiple rectangular blocks of cells by using your mouse to select a cell and drag it to expand the selection.

Drag and Drop

You can use the drag and drop action to edit the content of a table. You can select a column and drag it to another location in the table you are editing. When you drag a column and hover the cursor over a valid drop position, Oxygen XML Editor decorates the target location with bold rectangles. The same drag and drop action is also available for entire rows or columns by hovering the mouse cursor in front of rows or above column headers, then selecting the row or column and dragging them to the desired location.

Copy/Cut and Paste

In Oxygen XML Editor, you can copy/cut entire rows or columns of the table you are editing and paste the copied columns or rows inside the same table or inside other tables. You can also use the copy or cut actions for tables located in other documents. If you paste a row or column into non-table content, Oxygen XML Editor introduces a new table that contains the fragments of the copied row or column content.

For copied columns, the fragments are introduced starting with the header of the column. Also, if the copied column is from a *CALS* table, Oxygen XML Editor preserves column width information. This information is then used when you paste the column into another *CALS* table.

For copied cells, when pasting them into another cell without a selection (the cursor is just placed in the new cell), the copied cells are pasted while preserving their initial order and spacing. If pasting them into a selection of cells, first the content of the selected cells is deleted, then the copied cells are pasted with their initial order and spacing preserved and if there are more cells in the selection than in the copied content, the pasting will repeat the copied cells until the end of the selection.

Deleting Content

To delete the content of a cell, select the cell and press the **Delete** or **Backspace** key on your keyboard. If you press **Delete** or **Backspace** again, the selected table structure will also be removed.

To delete an entire row or column, place the cursor inside the row or column (or select it) and use the **Delete Row(s)** or **Delete Column(s)** actions from the toolbar or contextual menu. This will delete both the content and the table structure for the current row or column.

To delete a selection of multiple rows or columns, select them and use the **Delete Row(s)** or **Delete Column(s)** actions from the toolbar or contextual menu. This will delete both the content and the table structure for all rows or columns that exist in the current selection.

Navigating Cells

Along with the normal mouse navigation, you can also navigate between cells by using the arrow keys on your keyboard. By default, when using the arrow keys to navigate between table cells, the cursor jumps from one cell to another. However, if the ***Quick navigation in tables*** *option* is not selected in the **Cursor Navigation** preferences

page, using the arrow keys to navigate between table cells will cause the cursor to navigate between XML nodes, rather than jumping from cell to cell.

Related Information:

Adding Tables in DocBook on page 387
Adding Tables in DITA Topics on page 394
Adding Tables in XHTML Documents on page 404

Adding Tables in DocBook

You can use the **Insert Table** action on the toolbar or from the contextual menu to add a table in a DocBook document.

DocBook supports two types of tables:

- **CALS** table model - This is used for more advanced functionality.
- **HTML** table model - This is used for inserting a formal (captioned) HTML table.

Inserting a CALS Table Model in DocBook

To insert a *CALS* table model in DocBook documents, select the **Insert Table** action on the toolbar or from the contextual menu. The **Insert Table** dialog box appears. Select **CALS** for the table **Model**. This model allows you to configure a few more properties than the *HTML* model.

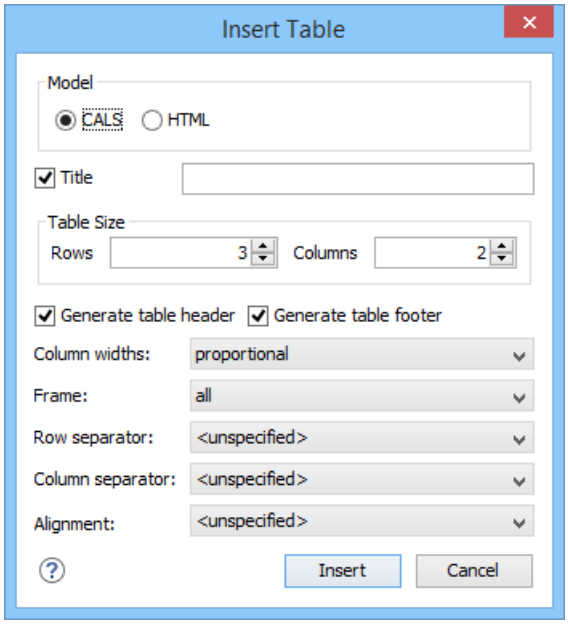


Figure 189: Insert Table Dialog Box - CALS Model

The dialog box allows you to configure the following options when you select the **CALS** table model:

Title

If this checkbox is selected, you can specify a title for your table in the adjacent text box.

Table Size

Allows you to choose the number of **Rows** and **Columns** for the table.

Generate table header

If selected, an extra row will be inserted at the top of the table to be used as the table header.

Generate table footer

If selected, an extra row will be inserted at the bottom of the table to be used as the table footer.

Column widths

Allows you to specify the type of properties for column widths (`colwidth` attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a `colwidth` attribute with the values listed as the number of shares followed by an asterisk. The value of the shares are totaled and rendered as a percent. For example, `colwidth="1* 2* 3*"` causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in **Author** mode, the values of the `colwidth` attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the `colwidth` attribute is `1*`.
- **dynamic** - If you choose this option, the columns are created without a specified width (`colwidth` attribute). Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in **Author** mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.
- **fixed** - The width is specified in fixed units. By default, the `pt` unit is inserted, but you can change the units in the **colspecs** (column specifications) section above the table or in **Text** mode. The following units are allowed: `pt` (points), `cm` (centimeters), `mm` (millimeters), `pi` (picas), `in` (inches).

Frame

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. There are a variety of allowed values, as specified in the *DocBook CALS table specifications*.

Row separator

Specifies whether or not to include row separators (`rowsep` attribute). The allowed values are: `0` (no separator) and 1 (include separators).

Column separator

Specifies whether or not to include column separators (`colsep` attribute). The allowed values are: `0` (no separator) and 1 (include separators).

Alignment

Specifies the alignment of the text within the table (`align` attribute). The allowed values are:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width.

  Note: The `justify` value cannot be rendered in **Author** mode, so you will only see it in the output.
- **char** - Aligns text to the leftmost occurrence of the value specified on the `char` attribute for alignment.

Note: The options in the **Insert Table** dialog box for DocBook documents are persistent, so changes made in one session will carry over to another.

When you click **Insert**, a CALS table is inserted into your document at the current cursor position.

When you insert a CALS table, you see a link for setting the `colspecs` (column specifications) of your table. Click the link to open the controls that allow you to adjust various column properties. Although they appear as part of the ***Author** mode*, the `colspecs` link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

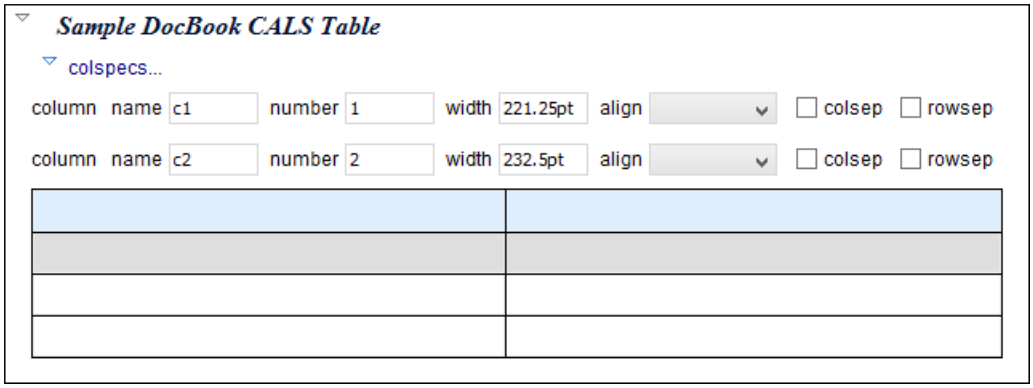


Figure 190: CALS Table in DocBook

Inserting an HTML Table Model

To insert an *HTML* table model in DocBook documents, select the **Insert Table** action on the toolbar or from the contextual menu. The **Insert Table** dialog box appears. Select **HTML** for the table **Model**.

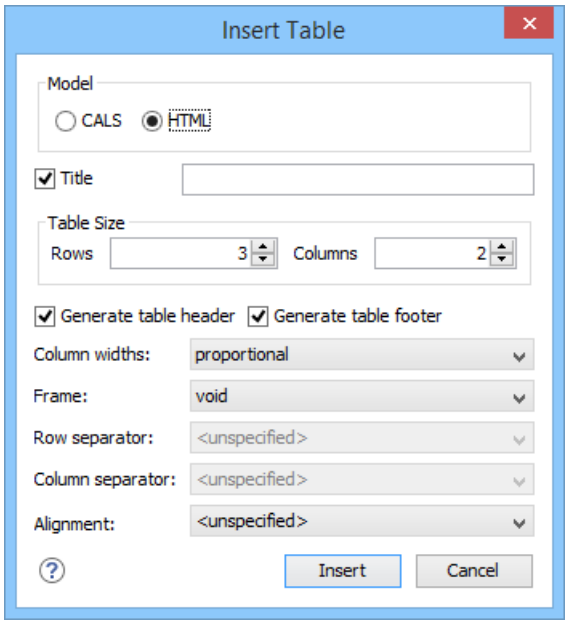


Figure 191: Insert Table Dialog Box - Simple Model

The dialog box allows you to configure the following options when you select the **HTML** table model:

Title

If this checkbox is selected, you can specify a title for your table in the adjacent text box.

Table Size

Allows you to choose the number of **Rows** and **Columns** for the table.

Generate table header

If selected, an extra row will be inserted at the top of the table to be used as the table header.

Generate table footer

If selected, an extra row will be inserted at the bottom of the table to be used as the table footer.

Column widths

Allows you to specify the type of properties for column widths (`width` attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a `width` attribute (in a `col` element) with the values listed as the number of shares followed by an asterisk. The value of the shares are totaled and rendered as a percent. For example, `width="1* 2* 3*"` causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in **Author** mode, the values of the `width` attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the `width` attribute is `1*`.
- **dynamic** - If you choose this option, the columns are created without a specified width. Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in **Author** mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.
- **fixed** - The width is specified in fixed units. By default, the `pt` unit is inserted, but you can change the units in the section above the table or in **Text** mode. In addition to the standard pixel, percentage, and relative values, this attribute also allows the special form “0*” (zero asterisk), which means that the width of the each column in the group should be the minimum width necessary to hold the contents.

Frame

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. There are a variety of allowed values, as specified in the *DocBook HTML table specifications*.

Alignment

Specifies the alignment of the text within the table (`align` attribute). The allowed values are:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width.

  Note: The `justify` value cannot be rendered in **Author** mode, so you will only see it in the output.
- **char** - Aligns text to the leftmost occurrence of the value specified on the `char` attribute for alignment.

Note: The options in the **Insert Table** dialog box for DocBook documents are persistent, so changes made in one session will carry over to another.

When you click **Insert**, an HTML style of table is inserted into your document at the current cursor position.

When you insert an HTML table, you see a section above the table that allows you to easily configure some properties without opening the **Table Properties** dialog box. Although this section appears as part of the ***Author*** *mode*, it will not appear in your output. It is just there to make it easier to adjust how the columns of your table are formatted.

Editing an Existing Table

You can edit the structure of an existing table using the table buttons on the toolbar (or in the contextual menu) to add or remove cells, rows, or columns, and to set basic table properties. Additional attributes can be used to fine-tune the formatting of your tables by using the ***Attributes*** *view* (**Window** > **Show View** > **Attributes**).

You can also use the ***Table Properties (Ctrl + T (Command + T on OS X))*** action from the toolbar or contextual menu to *modify many of the properties of the table*.

Also, remember that underneath the visual representation, both table models are really just XML. If necessary, you can edit the XML directly by switching to ***Text*** *mode*.

DocBook Table Layouts

The DocBook *framework* supports the following two table model layouts:

- *CALS table model*
- *HTML table model*

CALS Table Model Layout

The **CALS** table model allows for more flexibility and table customization than other models. When choosing a **CALS** table model from the **Insert Table** dialog box, you have access to more configurable properties. The layout of a **CALS** table includes a **colspecs** section that allows you to easily configure some properties without opening the **Table Properties** dialog box. For example, you can change the value of column widths (`colwidth` attribute) or the text alignment (`align` attribute). Although they appear as part of the ***Author*** *mode*, the `colspecs` link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

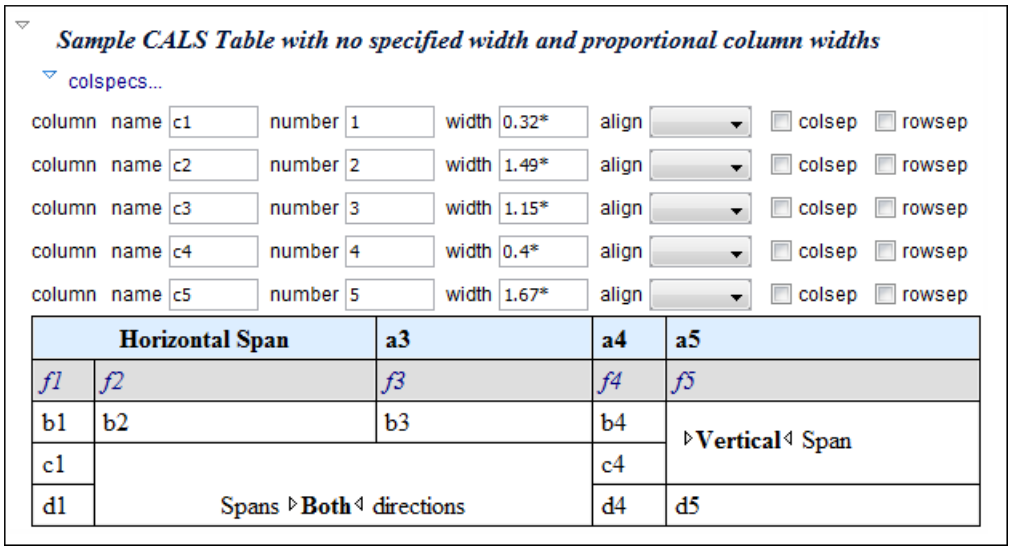


Figure 192: CALS Table in DocBook

HTML Table Model Layout

Choosing an **HTML** table model from the **Insert Table** dialog box in a DocBook document inserts a formal (captioned) HTML table. The layout of an *HTML* table includes a section above the table that allows you to easily configure some properties without opening the **Table Properties** dialog box. For example, you can change the value of column widths (`width` attribute) or the text alignment (`align` attribute). Although these properties appear as part of the ***Author*** *mode*, they will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

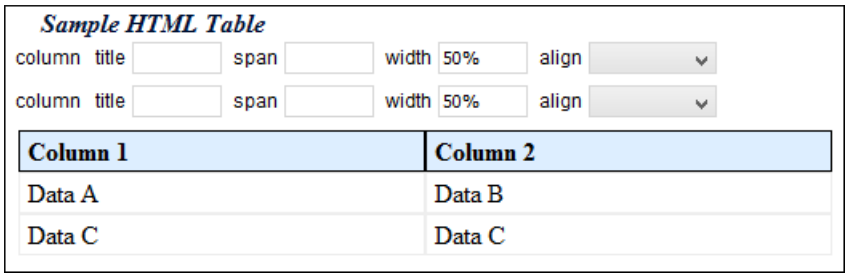


Figure 193: HTML Table in DocBook

Pasting Tables in DocBook

Tables that are pasted into a DocBook file are automatically converted to the *CALS* model. If you want to overwrite this behavior and instruct Oxygen XML Editor to convert them to *HTML* tables, set the `docbook.html.table` parameter to 1. You can find this parameter in the following stylesheet:

- *`[OXYGEN_INSTALL_DIR]`*`/frameworks/docbook/resources/xhtml2db5Driver.xsl` for DocBook 5
- *`[OXYGEN_INSTALL_DIR]`*`/frameworks/docbook/resources/xhtml2db4Driver.xsl` for DocBook 4

Table Validation in DocBook

Oxygen XML Editor reports table layout problems that are detected in manual or automatic validations. The types of errors that may be reported for DocBook table layout problems include:

CALS Tables

- A row has fewer cells than the number of columns detected from the table `cols` attribute.
- A row has more cells than the number of columns detected from the table `cols` attribute.
- A cell has a vertical span greater than the available rows count.
- The number of `colspecs` is different than the number of columns detected from the table `cols` attribute.

- The number of columns detected from the table `cols` attribute is different than the number of columns detected in the table structure.
- The value of the `cols`, `rowsep`, or `colsep` attributes are not numeric.
- The `namest`, `nameend`, or `colname` attributes point to an incorrect column name.

HTML Tables

- A row has fewer cells than the number of table columns.
- The value of the `colspan`, `rowspan`, or `span` attributes are not numeric.
- A cell has a vertical span greater than the available rows count.

Editing Table Properties in DocBook

You can edit the structure of an existing table using the table buttons on the toolbar (or from the contextual menu) to add or remove cells, rows, or columns, and to set basic table properties. Additional attributes can be used to fine-tune the formatting of your tables by using the ***Attributes*** *view* (**Window** > **Show View** > **Attributes**).

You can use the **Table Properties (Ctrl + T (Command + T on OS X))** action to modify many of the properties of the table. You can also adjust some of the properties in the specification section above the table.

The **Table properties** dialog box allows you to set specific properties to the table elements. The options that are available depend on the context and location within the table in which the action was invoked.

Note: Some properties allow the following special values, depending on the context and the current properties or values:

- **<not set>** - Use this value if you want to remove a property.
- **<preserve>** - If you select multiple elements that have the same property set to different values, you can choose this value to keep the values that are already set. In some cases it can also be used to keep the current non-standard value for a particular property.

Edit Table Properties for a CALS Table Model

For a **CALS** table model, the **Table properties** dialog box includes four tabs of options:

- **Table** tab - The options in this tab apply to the entire table.
- **Row** tab - The options in this tab apply to the current row or selection of multiple rows. A message at the bottom of the tab tells you how many rows will be affected.
- **Column** tab - The options in this tab apply to the current column or selection of multiple columns. A message at the bottom of the tab tells you how many columns will be affected.
- **Cell** tab - The options in this tab apply to the current cell or selection of multiple cells. A message at the bottom of the tab tells you how many cells will be affected.

The options in four tabs include a **Preview** pane that shows a representation of the modification.

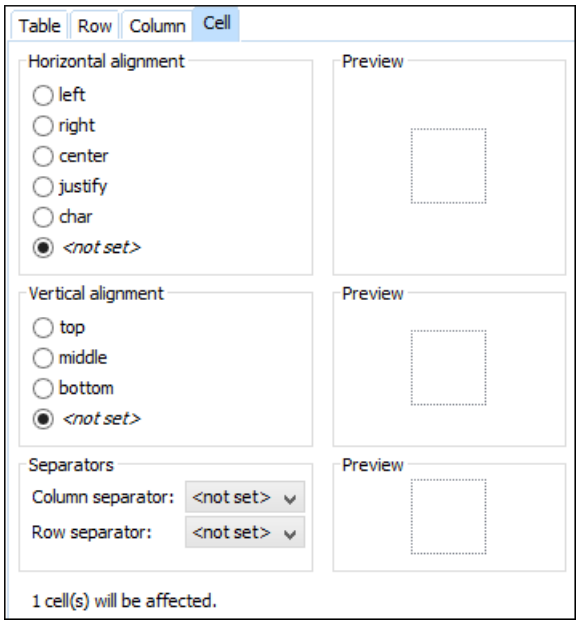


Figure 194: Table Properties Dialog Box with Cell Tab Selected (DocBook CALS Table Model)

The options in the four tabs include the following:

Horizontal alignment (Available in the Table, Column, and Cell tabs)

Specifies the horizontal alignment of text within the current table/column/cell or selection of multiple columns/cells (`align` attribute). The allowed values are as follows:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width.

  Note: The `justify` value cannot be rendered in **Author** mode, so you will only see it in the output.
- **char** - Aligns text to the leftmost occurrence of the value specified on the `char` attribute for alignment.

Vertical alignment (Available in the Row and Cell tabs)

Specifies the vertical alignment of text within the current row/cell or selection of multiple rows/cells (`valign` attribute). The allowed values are as follows:

- **top** - Aligns the text at the top of the cell.
- **middle** - Aligns the text in a vertically centered position.
- **bottom** - Aligns the text at the bottom of the cell.

Column separator (Available in the Table, Column, and Cell tabs)

Specifies whether or not to include column separators (borders/grid lines) in the form of the `colsep` attribute. The allowed values are: `0` (no separator) and 1 (include separators).

Row separator (Available in all four tabs)

Specifies whether or not to include row separators (borders/grid lines) in the form of the `rowsep` attribute. The allowed values are: `0` (no separator) and 1 (include separators).

Frame

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. There are a variety of allowed values, as specified in the *DocBook CALS table specifications*.

Row type (Available in the Row tab only)

Allows you change the row to a header, body, or footer type of row (within a `thead`, `tbody`, or `tfoot` attribute).

Edit Table Properties for an HTML Table Model

For an **HTML** table model, the **Table properties** dialog box includes four tabs of options (**Table**, **Row**, **Column**, and **Cell**) and the options include a **Preview** pane that shows a representation of the modification.

The options in the four tabs include the following:

Frame (Available only in the Table tab)

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. There are a variety of allowed values, as specified in the *DocBook HTML table specifications*.

Row type (Available in the Row tab only)

Allows you change the row to a header, body, or footer type of row (within a `thead`, `tbody`, or `tfoot` attribute).

Horizontal alignment (Available in the Row, Column, and Cell tabs)

Specifies the horizontal alignment for the text in the current row/column/cell or selection of multiple rows/columns/cells (`align` attribute). The allowed values are:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width.

  Note: The `justify` value cannot be rendered in **Author** mode, so you will only see it in the output.
- **char** - Aligns text to the leftmost occurrence of the value specified on the `char` attribute for alignment.

Vertical alignment (Available in the Row, Column, and Cell tabs)

Specifies the vertical alignment for the text in the current row/column/cell or selection of multiple rows/columns/cells (`valign` attribute). The allowed values are:

- **top** - Aligns the text at the top of the cell.
- **middle** - Aligns the text in a vertically centered position.
- **bottom** - Aligns the text at the bottom of the cell.
- **baseline** - Sets the row so that all the table data share the same baseline. This often has the same effect as the `bottom` value. However, if the fonts are different sizes, the `baseline` value often makes the table look better.

Related Information:

Editing Tables in Author Mode on page 385

Adding Tables in DITA Topics

You can use the **Insert Table** action on the toolbar or from the contextual menu to add a table in a DITA topic. By default, DITA supports four types of tables:

- *DITA Simple table model* - This is the most commonly used model for basic tables.
- *CALS table model (OASIS Exchange Table Model)* - This is used for more advanced functionality.
- *DITA Choice table model* - This is used within a `step` element in a DITA Task document to describe a series of optional choices that a user must make before proceeding.
- *DITA Properties table model* - This is used in DITA Reference documents to describe a property (for example, its type, value, and description).

If you are using a specialized DITA vocabulary, it may contain specialized versions of these table models.

Since DITA is a structured format, you can only insert a table in places in the structure of a topic where tables are allowed. The Oxygen XML Editor toolbar provides support for entering and editing tables. It also helps to indicate where you are allowed to insert a table or its components by disabling the appropriate buttons.

Inserting a Simple Table Model

To insert a **Simple** DITA table, select the **Insert Table** action on the toolbar or from the contextual menu (or the **Table** submenu from the **DITA** menu). The **Insert Table** dialog box appears. Select **Simple** for the table **Model**.

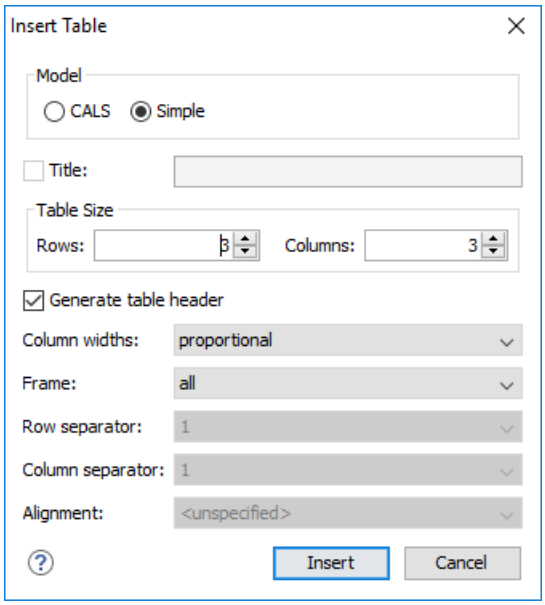


Figure 195: Insert Table Dialog Box - Simple Model

The dialog box allows you to configure the following options when you select the **Simple** table model:

Title

If this checkbox is selected, you can specify a title for your table in the adjacent text box.

Generate table header

If selected, an extra row will be inserted at the top of the table to be used as the table header.

Column widths

Allows you to specify the type of properties for column widths (`colwidth` attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a `relcolwidth` attribute with the values listed as the number of shares followed by an asterisk. The value of the shares are totaled and rendered as a percent. For example, `relcolwidth="1* 2* 3*"` causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in **Author** mode, the values of the `relcolwidth` attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the `relcolwidth` attribute is `1*`.
- **dynamic** - If you choose this option, the columns are created without a specified width. Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in **Author** mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.

Frame

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

Note: The options in the **Insert Table** dialog box for DITA documents are persistent, so changes made in one session will carry over to another.

When you click **Insert**, a simple table is inserted into your document at the current cursor position.

Inserting a CALS Table Model (OASIS Exchange Table)

To insert an OASIS Exchange Table (**CALS**), select the **Insert Table** action on the toolbar or from the contextual menu (or the **Table** submenu from the **DITA** menu). The **Insert Table** dialog box appears. Select **CALS** for the table **Model**. This model allows you to configure more properties than the *Simple* model.

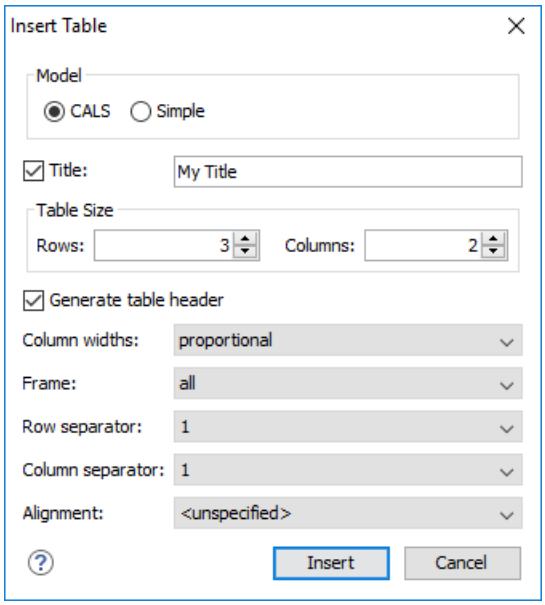


Figure 196: Insert Table Dialog Box - CALS Model

The dialog box allows you to configure the following options when you select the **CALS** table model:

Title

If this checkbox is selected, you can specify a title for your table in the adjacent text box.

Table Size

Allows you to choose the number of **Rows** and **Columns** for the table.

Generate table header

If selected, an extra row will be inserted at the top of the table to be used as the table header.

Column widths

Allows you to specify the type of properties for column widths (`colwidth` attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a `colwidth` attribute with the values listed as the number of shares followed by an asterisk. The value of the shares are totaled and rendered as a percent. For example, `colwidth="1* 2* 3*"` causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in **Author** mode, the values of the `colwidth` attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the `colwidth` attribute is `1*`.
- **dynamic** - If you choose this option, the columns are created without a specified width (`colwidth` attribute). Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in **Author** mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.
- **fixed** - The width is specified in fixed units. By default, the `pt` unit is inserted, but you can change the units in the **colspecs** (column specifications) section above the table or in **Text** mode. The following units are allowed: `pt` (points), `cm` (centimeters), `mm` (millimeters), `pi` (picas), `in` (inches).

Frame

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

Row separator

Specifies whether or not to include row separators (`rowsep` attribute). The allowed values are: `0` (no separator) and 1 (include separators).

Column separator

Specifies whether or not to include column separators (`colsep` attribute). The allowed values are: `0` (no separator) and 1 (include separators).

Alignment

Specifies the alignment of the text within the table (`align` attribute). The allowed values are:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width.

  Note: The `justify` value cannot be rendered in **Author** mode, so you will only see it in the output.
- **char** - Aligns text to the leftmost occurrence of the value specified on the `char` attribute for alignment.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

Note: The options in the **Insert Table** dialog box for DITA documents are persistent, so changes made in one session will carry over to another.

When you click **Insert**, a CALS table is inserted into your document at the current cursor position.

When you insert a CALS table, you see a link for setting the `colspecs` (column specifications) of your table. Click the link to open the controls that allow you to adjust various column properties. Although they appear as part of the ***Author** mode*, the `colspecs` link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

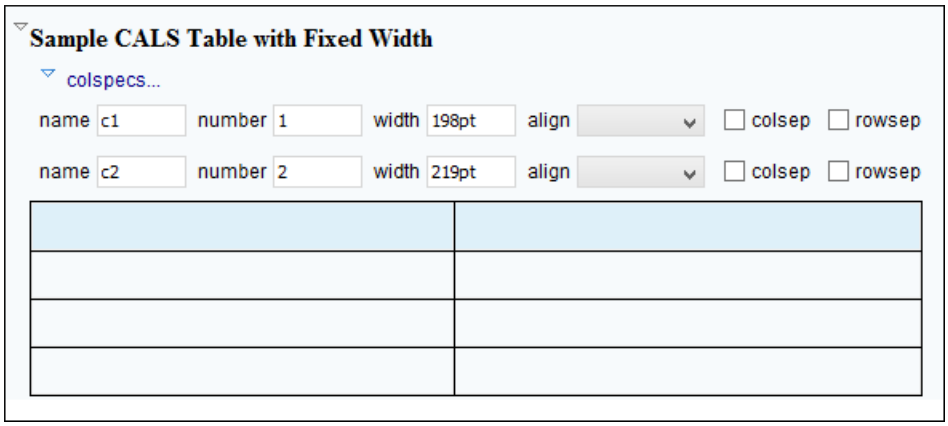


Figure 197: CALS Table in DITA

Inserting a Choice Table Model

To insert a **Choice** table within a `step` element in a DITA Task document, select the **Insert Table** action on the toolbar or in the **Insert** submenu from the contextual menu (or the **Table** submenu from the **DITA** menu), or select `choicetable` from the *Content Completion Assistant*. The **Insert Table** dialog box appears. Select **Simple** for the table **Model**.

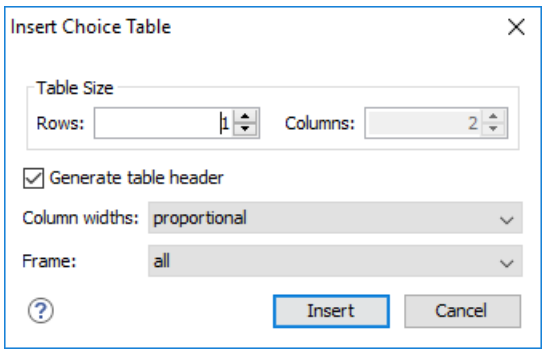


Figure 198: Insert Table Dialog Box - Choice Model

The dialog box allows you to configure the following options when you insert a *Choice* table model within a DITA Task:

Table Size

Allows you to choose the number of **Rows** and **Columns** for the table.

Generate table header

If selected, an extra row will be inserted at the top of the table to be used as the table header.

Column widths

Allows you to specify the type of properties for column widths (`colwidth` attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a `relcolwidth` attribute with the values listed as the number of shares followed by an asterisk. The value of the shares are totaled and rendered as a percent. For example, `relcolwidth="1* 2* 3*"` causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in **Author** mode, the values of the `relcolwidth` attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the `relcolwidth` attribute is `1*`.
- **dynamic** - If you choose this option, the columns are created without a specified width. Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in

Author mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.

Frame

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

When you click **Insert**, a *Choice* table is inserted into your DITA Task document at the current cursor position (within a `step` element).

Inserting a Properties Table Model

To insert a **Properties** table within a `refbody` element in a DITA Reference document, select the **Insert Table** action on the toolbar or in the **Insert** submenu from the contextual menu (or the **Table** submenu from the **DITA** menu), or select `properties(wizard)` from the *Content Completion Assistant*. The **Insert Table** dialog box appears. Select **Properties** for the table **Model**.

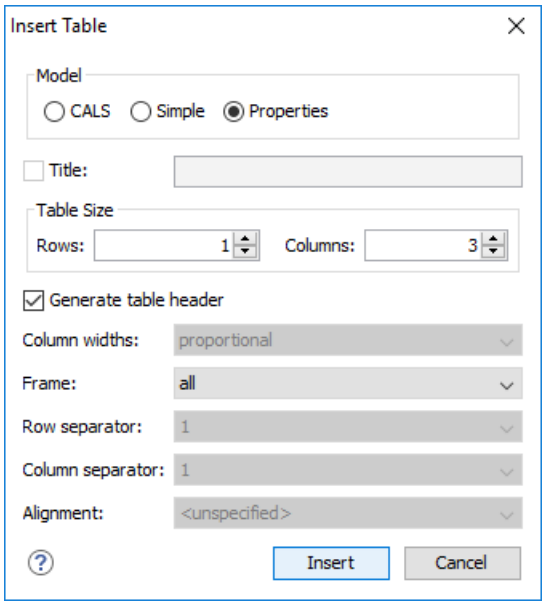


Figure 199: Insert Table Dialog Box - Properties Model

The dialog box allows you to configure the following options when you insert a *Properties* table model within a DITA Reference:

Table Size

Allows you to choose the number of **Rows** and **Columns** for the table.

Generate table header

If selected, an extra row will be inserted at the top of the table to be used as the table header.

Frame

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

When you click **Insert**, a *Properties* table is inserted into your DITA Reference document at the current cursor position (within a `refbody` element).

Editing an Existing Table

You can edit the structure of an existing table using the table buttons on the toolbar (or in the contextual menu) to add or remove cells, rows, or columns, and to set basic table properties. Additional attributes can be used to fine-tune the formatting of your tables by using the ***Attributes*** *view* (**Window** > **Show View** > **Attributes**). See the *DITA documentation* for a full explanation of these attributes.

You can also use the ***Table Properties (Ctrl + T (Command + T on OS X))*** action from the toolbar or contextual menu (or **DITA** menu) to *modify many of the properties of the table*.

Also, remember that underneath the visual representation, both table models are really just XML. If necessary, you can edit the XML directly by switching to ***Text*** *mode*.

Related Information:
Editing Tables in Author Mode on page 385

DITA Table Layouts

Depending on the context, DITA accepts the following table layouts:

- *CALS table model*
- *Simple table model*
- *Choice table model*
- *Properties table model*

CALS Table Model Layout

The **CALS** table model allows for more flexibility and table customization than other models. When choosing a **CALS** table model from the **Insert Table** dialog box, you have access to more configurable properties. The layout of a **CALS** table includes a **colspecs** section that allows you to easily configure some properties without opening the **Table Properties** dialog box. For example, you can change the value of column widths (`colwidth` attribute) or the text alignment (`align` attribute). Although they appear as part of the ***Author*** *mode*, the `colspecs` link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

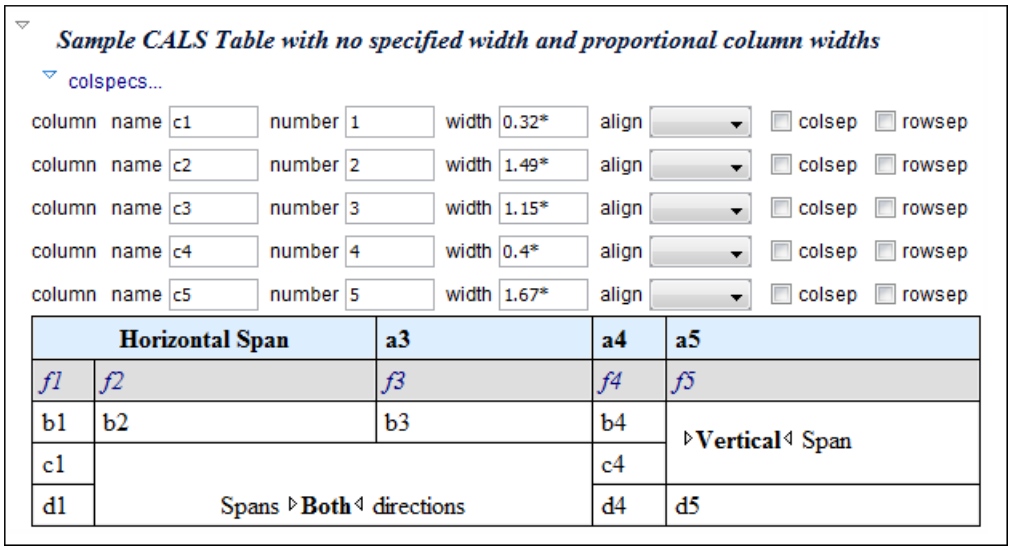


Figure 200: CALS Table in DITA

Simple Table Model Layout

When choosing a **Simple** table model from the **Insert Table** dialog box, you only have access to configure a few properties. For example, you can choose the number of rows and columns, specify values for frames, and choose from a few types of properties for the column width. The layout of this type of table is very simple, as the name suggests.

Header 1	Header 2
Column 1	Column 2

Figure 201: DITA Simple Table

Choice Table Model Layout

A **Choice** table model is used within a `step` element in a DITA Task document to describe a series of optional choices that a user must make before proceeding. The `choicetable` element is a useful device for documenting options within a single step of a task. You can insert *Choice* tables in DITA Task documents either by selecting `choicetable` from the *Content Completion Assistant* (within a `step` element) or by using the **Insert Table** action on the toolbar or from the contextual menu). The options and layout of a *Choice* table is similar to the *Simple* table model.

Option	Description
Opt A	
Opt B	
Opt C	

Figure 202: DITA Choice Table

Properties Table Model Layout

A **Properties** table model is used within a `refbody` element in a DITA Reference document to describe a property (for example, its type, value, and description). You can insert *Properties* tables in DITA Reference documents either by selecting `properties(wizard)` from the *Content Completion Assistant* (within a `refbody` element) or by using the **Insert Table** action on the toolbar (or from the contextual menu) and selecting **Properties** for the **Model**. The layout of a *Properties* table is very simple. It allows for a maximum of 3 columns (typically for property type, value, and description) and the only options available are for whether or not you want a header row and for specifying frames (borders).

Property Type	Property Value	Property Description
A	B	C

Figure 203: DITA Properties Table

Table Validation in DITA

Oxygen XML Editor reports table layout problems that are detected in manual or automatic validations. When you validate a *DITA map* with the **Validate and Check for Completeness** action, if the ***Report table layout problems*** *option is selected in the* ***DITA Map Completeness Check*** *dialog box*, table layout problems will be reported in the validation results. The types of errors that may be reported for DITA table layout problems include:

CALS Tables

- A row has fewer cells than the number of columns detected from the table `cols` attribute.
- A row has more cells than the number of columns detected from the table `cols` attribute.
- A cell has a vertical span greater than the available rows count.
- The number of `colspecs` is different than the number of columns detected from the table `cols` attribute.
- The number of columns detected from the table `cols` attribute is different than the number of columns detected in the table structure.
- The value of the `cols`, `rowsep`, or `colsep` attributes are not numeric.
- The `namest`, `nameend`, or `colname` attributes point to an incorrect column name.

Simple or Choice Tables

- A row has fewer cells than the number of table columns.

Editing Table Properties in DITA

To customize the look of a table in DITA, place the cursor anywhere in a table and invoke the **Table Properties (Ctrl + T (Command + T on OS X))** action from the toolbar or the **Table** submenu of the contextual menu (or **DITA** menu). This opens the **Table properties** dialog box.

The **Table properties** dialog box allows you to set specific properties to the table elements. The options that are available depend on the context and location within the table in which the action was invoked.

Note: Some properties allow the following special values, depending on the context and the current properties or values:

- **<not set>** - Use this value if you want to remove a property.
- **<preserve>** - If you select multiple elements that have the same property set to different values, you can choose this value to keep the values that are already set. In some cases it can also be used to keep the current non-standard value for a particular property.

Edit Table Properties for a CALS Table Model

For a **CALS** table model, the **Table properties** dialog box includes four tabs of options:

- **Table** tab - The options in this tab apply to the entire table.
- **Row** tab - The options in this tab apply to the current row or selection of multiple rows. A message at the bottom of the tab tells you how many rows will be affected.
- **Column** tab - The options in this tab apply to the current column or selection of multiple columns. A message at the bottom of the tab tells you how many columns will be affected.
- **Cell** tab - The options in this tab apply to the current cell or selection of multiple cells. A message at the bottom of the tab tells you how many cells will be affected.

The options in four tabs include a **Preview** pane that shows a representation of the modification.

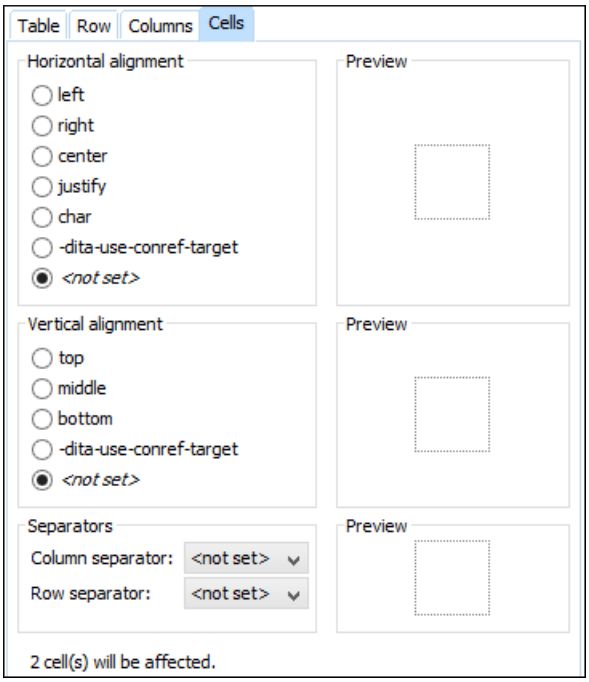


Figure 204: Table Properties Dialog Box with Cell Tab Selected (DITA CALS Table Model)

The options in the four tabs include the following:

Horizontal alignment (Available in the Table, Column, and Cell tabs)

Specifies the horizontal alignment of text within the current table/column/cell or selection of multiple columns/cells (`align` attribute). The allowed values are as follows:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width.

  Note: The `justify` value cannot be rendered in **Author** mode, so you will only see it in the output.
- **char** - Aligns text to the leftmost occurrence of the value specified on the `char` attribute for alignment.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

Vertical alignment (Available in the Row and Cell tabs)

Specifies the vertical alignment of text within the current row/cell or selection of multiple rows/cells (`valign` attribute). The allowed values are as follows:

- **top** - Aligns the text at the top of the cell.
- **middle** - Aligns the text in a vertically centered position.
- **bottom** - Aligns the text at the bottom of the cell.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

Column separator (Available in the Table, Column, and Cell tabs)

Specifies whether or not to include column separators (borders/grid lines) in the form of the `colsep` attribute. The allowed values are: `0` (no separator) and 1 (include separators).

Row separator (Available in all four tabs)

Specifies whether or not to include row separators (borders/grid lines) in the form of the `rowsep` attribute. The allowed values are: `0` (no separator) and 1 (include separators).

Frame (Available only in the Table tab)

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

Edit Table Properties for a Simple, Choice, or Properties Table Model

For a *Simple*, *Choice*, *Properties* table model, the **Table properties** dialog box only allows you to edit a few options.

Table tab

Frame

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

Row tab (not available for *Properties* tables)

Row type

Allows you change the row to a body or header type of row.

Related Information:

Adding Tables in DITA Topics on page 394
Editing Tables in Author Mode on page 385

Adding Tables in XHTML Documents

You can use the **Insert Table** action on the toolbar or from the contextual menu to add a table in an XHTML document. This action opens the **Insert Table** dialog box.

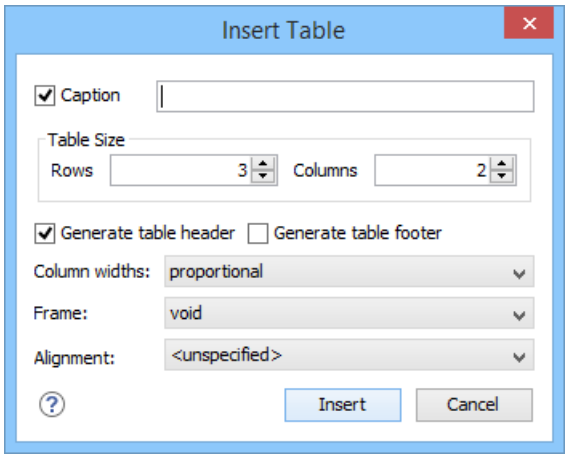


Figure 205: Insert Table Dialog Box in XHTML

The dialog box allows you to configure the following options:

Caption

If this checkbox is selected, you can specify a title (caption) for your table in the adjacent text box.

Table Size

Allows you to choose the number of **Rows** and **Columns** for the table.

Generate table header

If selected, an extra row will be inserted at the top of the table to be used as the table header.

Generate table footer

If selected, an extra row will be inserted at the bottom of the table to be used as the table footer.

Column widths

Allows you to specify the type of properties for column widths (`width` attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a `width` attribute (in a `col` element) with the values listed as the number of shares followed by an asterisk. The value of the shares are totaled and rendered as a percent. For example, `width="1* 2* 3*"` causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in **Author** mode, the values of the `width` attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the `width` attribute is `1*`.
- **dynamic** - If you choose this option, the columns are created without a specified width. Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in **Author** mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.
- **fixed** - The width is specified in fixed units. By default, the `pt` unit is inserted, but you can change the units in the section above the table or in **Text** mode. In addition to the standard pixel, percentage, and relative values, this attribute also allows the special form “0*” (zero asterisk), which means that the width of the each column in the group should be the minimum width necessary to hold the contents.

Frame

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. There are a variety of allowed values, as specified in HTML specifications.

Alignment

Specifies the alignment of the text within the table (`align` attribute). The allowed values are:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width.

Note: The `justify` value cannot be rendered in **Author** mode, so you will only see it in the output.

- **char** - Aligns text to the leftmost occurrence of the value specified on the `char` attribute for alignment.

Note: The options in the **Insert Table** dialog box for XHTML documents are persistent, so changes made in one session will carry over to another.

When you click **Insert**, an HTML style of table is inserted into your XHTML document at the current cursor position.

When you insert an HTML table, you see a link for setting the `colspecs` (column specifications) of your table. Click the link to open the controls that allow you to adjust various column properties. Although they appear as part of the ***Author*** *mode*, the `colspecs` link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

Editing an Existing Table

You can edit the structure of an existing table using the table buttons on the toolbar (or in the contextual menu) to add or remove cells, rows, or columns, and to set basic table properties. Additional attributes can be used to fine-tune the formatting of your tables by using the ***Attributes*** *view* (**Window** > **Show View** > **Attributes**). Also, remember that underneath the visual representation, the table is really just XML. If necessary, you can edit the XML directly by switching to ***Text*** *mode*.

XHTML Table Layout

When you insert a table in an XHTML document, an HTML type of table is added. The layout of an *XHTML* table includes a **colspecs** section that allows you to easily configure some properties. For example, you can change the value of column widths (`width` attribute) or the text alignment (`align` attribute). Although they appear as part of the ***Author*** *mode*, the `colspecs` link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

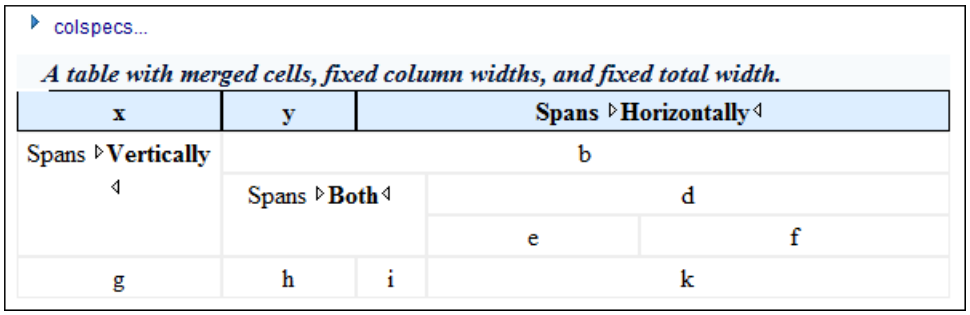


Figure 206: Table Layout in XHTML Documents

Table Validation in XHTML

Oxygen XML Editor reports table layout problems that are detected in manual or automatic validations. The types of errors that may be reported for XHTML table layout problems include:

HTML Tables

- A row has fewer cells than the number of table columns.
- The value of the `colspan`, `rowspan`, or `span` attributes are not numeric.
- A cell has a vertical span greater than the available rows count.

Adding Tables in TEI Documents

You can use the **Insert Table** action on the toolbar or from the contextual menu to add a table in a TEI document. This action opens the **Insert Table** dialog box.

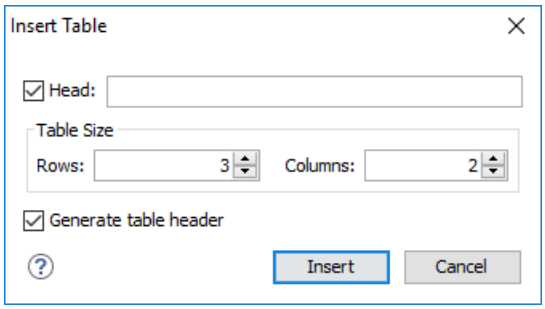


Figure 207: Insert Table Dialog Box in TEI

The dialog box allows you to configure the following options:

Head

If this checkbox is selected, you can specify a title for your table in the adjacent text box.

Table Size

Allows you to choose the number of **Rows** and **Columns** for the table.

Generate table header

If selected, an extra row will be inserted at the top of the table to be used as the table header.

Note: The options in the **Insert Table** dialog box for TEI documents are persistent, so changes made in one session will carry over to another.

When you click **Insert**, a simple table is inserted into your TEI document at the current cursor position.

Editing an Existing Table

You can edit the structure of an existing table using the table buttons on the toolbar (or in the contextual menu) to add or remove cells, rows, or columns. Additional attributes can be used to fine-tune the formatting of your tables by using the ***Attributes*** *view* (**Window** > **Show View** > **Attributes**). Also, remember that underneath the visual representation, the table is really just XML. If necessary, you can edit the XML directly by switching to ***Text*** *mode*.

Sorting Content in Tables and List Items

Oxygen XML Editor offers support for sorting the content of tables and list items of ordered and unordered lists.

What do you want to do?

- *Sort an entire table*.
- *Sort a selection of rows in a table*.
- *Sort a table that contains cells merged over multiple rows*.
- *Sort list items*.

Sorting a Table

To sort rows in a table, select the entire table (or specific rows) and use the **Sort** action from the main toolbar or the contextual menu. This opens the **Sort** dialog box.

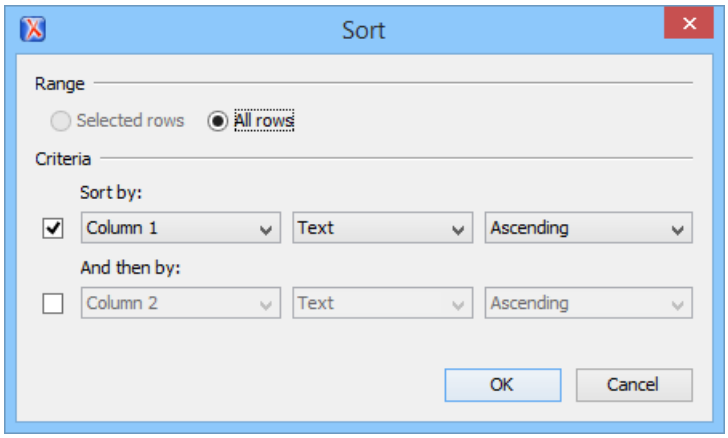


Figure 208: Sort Dialog Box

This dialog box sets the range that is sorted and the sorting criteria. The range is automatically selected depending on whether you sort an entire table or only a selection of its rows.

Note: When you invoke the sorting operation over an entire table, the **Selected rows** option is disabled.

The **Criteria** section specifies the sorting criteria (a maximum of three sorting criteria are available), defined by the following:

- A name, which is collected from the column heading.
- The type of the information that is sorted. You can choose between the following:
  - **Text** - Alphanumeric characters.
  - **Numeric** - Regular integer or floating point numbers are accepted.
  - **Date** - Default date and time formats from the local OS are accepted (such as *short*, *medium*, *long*, *full*, *xs:date*, and *xs:dateTime*).
- The sorting direction (either *ascending* or *descending*).

The sort criteria is automatically set to the column where the cursor is located at the time when the sorting operation is invoked.

After you finish configuring the options in the **Sort** dialog box, click **OK** to complete the sorting operation. If you want to revert to the initial order of your content, press **Ctrl + Z (Command + Z on OS X)** on your keyboard.

Note: The sorting support takes the value of the `xml:lang` attribute into account and sorts the content in a natural order.

Sorting a Selection of Rows

To sort a selection of rows in a table, select the rows that you want to sort and either right-click the selection and choose **Sort**, or click **Sort** on the main toolbar. This opens the **Sort** dialog box.

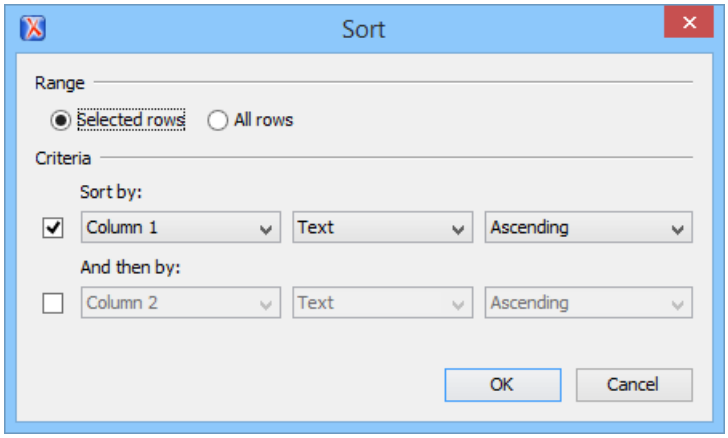


Figure 209: Sort Selected Rows

This dialog box sets the range that is sorted and the sorting criteria. The range is automatically selected depending on whether you sort an entire table or only a selection of its rows.

The **Sort** dialog box also allows you to apply the sorting operation to the entire table, using the **All rows** option.

The **Criteria** section specifies the sorting criteria (a maximum of three sorting criteria are available), defined by the following:

- A name, which is collected from the column heading.
- The type of the information that is sorted. You can choose between the following:
  - **Text** - Alphanumeric characters.
  - **Numeric** - Regular integer or floating point numbers are accepted.
  - **Date** - Default date and time formats from the local OS are accepted (such as *short*, *medium*, *long*, *full*, *xs:date*, and *xs:dateTime*).
- The sorting direction (either *ascending* or *descending*).

The sort criteria is automatically set to the column where the cursor is located at the time when the sorting operation is invoked.

After you finish configuring the options in the **Sort** dialog box, click **OK** to complete the sorting operation. If you want to revert to the initial order of your content, press **Ctrl + Z (Command + Z on OS X)** on your keyboard.

Note: The sorting support takes the value of the `xml:lang` attribute into account and sorts the content in a natural order.

Sort Using Multiple Criteria

You can also sort an entire table or a selection of its rows based on multiple sorting criteria. To do so, select the rest of boxes in the **Criteria** section of the **Sort** dialog box, configure the applicable items, and click **OK** to complete the sorting operation.

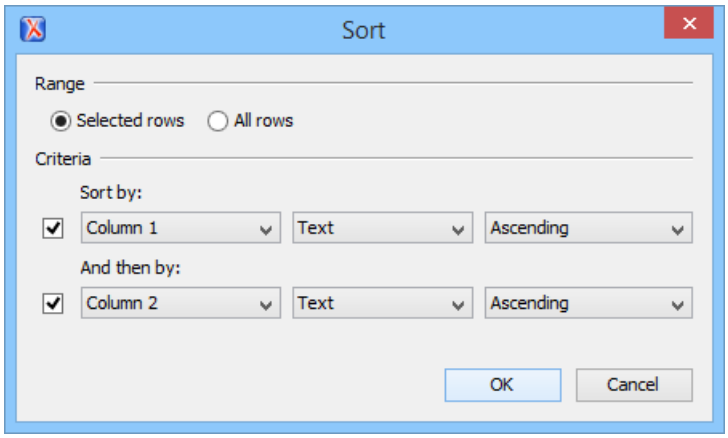


Figure 210: Sorting Based on Multiple Criteria

Sorting a Table that Contains Merged Cells

If a table contains cells that span over multiple rows, you can not perform the sorting operation over the entire table. Still, the sorting mechanism works over a selection of rows that do not contain *rowspans*.

Note: For this type of table, the **Sort** dialog box keeps the **All rows** option disabled even if you perform the sorting operation over a selection of rows.

Sorting List Items

A sorting operation can be performed on various types of lists and list items. Oxygen XML Editor provides support for sorting the following types of lists:

- Ordered list (`ol`)
- Unordered list (`ul`)
- Parameter list (`parml`)

- Simple list (`sl`)
- Required conditions (`reqconds`)
- Supplies list (`supplyli`)
- Spare parts list (`sparesli`)
- Safety conditions (`safety`)
- Definitions list (`dl`)

The sorting mechanism works on an entire list or on a selection of list items. To sort items in a list, select the items or list and use the **Sort** action from the main toolbar or the contextual menu. This opens the **Sort** dialog box.

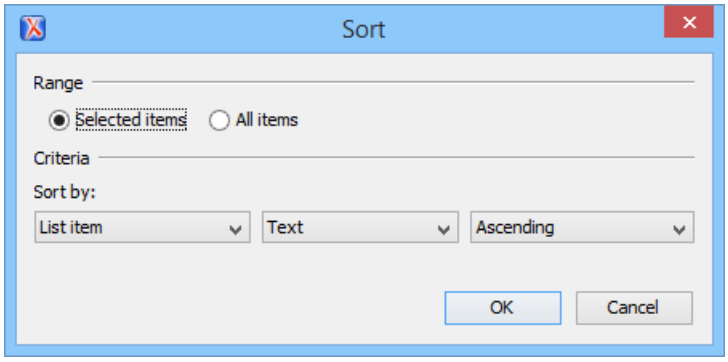


Figure 211: Sorting List Items

This dialog box sets the range that is sorted and the sorting criteria. The range is automatically selected depending on whether you sort an entire list or only a selection of its items.

Note: When you invoke the sorting operation over an entire list, the **Selected rows** option is disabled.

The **Criteria** section specifies the sorting criteria, defined by the following:

- The name of the type of item being sorted.
- The type of the information that is sorted. You can choose between the following:
  - **Text** - Alphanumeric characters.
  - **Numeric** - Regular integer or floating point numbers are accepted.
  - **Date** - Default date and time formats from the local OS are accepted (such as *short*, *medium*, *long*, *full*, *xs:date*, and *xs:dateTime*).
- The sorting direction (either *ascending* or *descending*).

After you finish configuring the options in the **Sort** dialog box, click **OK** to complete the sorting operation. If you want to revert to the initial order of your content, press **Ctrl + Z (Command + Z on OS X)** on your keyboard.

Note: The sorting support takes the value of the `xml:lang` attribute into account and sorts the content in a natural order.

Inserting Images

To insert an image in a document while editing in **Author** mode, use one of the following methods:

- Click the **Insert Image** action from the toolbar. This opens a dialog box that allows you to choose the image file you want to insert and configure some properties. Oxygen XML Editor tries to reference the image with a path that is relative to that of the document you are currently editing. For example, if you want to add the `file:/C:/project/xml/dir/img1.jpg` image into the `file:/C:/project/xml/doc1.xml` document, Oxygen XML Editor inserts a reference to `dir/img1.jpg`. This is useful when multiple users work on a common project and they have it stored in multiple locations.

  Note: The **Insert Image** action is available for the following document types: DocBook 4, DocBook 5, DITA, TEI, XHTML.

- Drag an image from other application and drop it in the **Author** editing mode. If it is an image file, it is inserted as a reference to the image file. For example, in a DITA topic the path of the image file is inserted as the value of the `href` attribute in an `image` element:

  ```
  <image href="../images/image_file.png"/>
  ```

  Note: To replace an image, just drag and drop a new image over the existing one. Oxygen XML Editor will automatically update the reference to the new image.
- Copy an image file from another document or another application (such as a system file browser or web browser) and paste it into your document. Oxygen XML Editor will insert it as a reference to the image file the same as the drag/drop method.
- Select an image (or part of an image) from another application (such as an image editor), copy it, and paste it into your document. Oxygen XML Editor will prompt you to save it. After saving the image, a reference to that file path is inserted at the paste position.

Related Information:

Image Map Editor on page 414
*Image Rendering in **Author** Mode* on page 411
Adding Video, Audio, and Embedded HTML Resources in DITA Topics on page 1626

Image Rendering in Author Mode

The **Author** mode and the output transformation process might render the images referenced in an XML document differently, since they use different rendering engines.

Table 8: Supported Image Formats

Image Type	Support	Additional Information
GIF	built-in	Animations not yet supported
JPG, JPEG	built-in	*JPEG images with CMYK color profiles* are properly rendered only if color profile is inside the image.
PNG	built-in	
SVG, SVGZ, WMF	built-in	Rendered using the open-source Apache Batik library that supports SVG 1.1.
BMP	built-in	
TIFF	built-in	Rendered using a part of the Java JAI Image library.
EPS	built-in	Renders the preview TIFF image inside the EPS.
AI	built-in	Renders the preview image inside the Adobe Illustrator file.
JPEG 2000, WBMP	plugin	Renders by *installing the Java Advanced Imaging (JAI) Image I/O Tools plug-in*.
CGM	plugin	Renders by *installing an additional library*.
PDF	plugin	Renders by *installing the Apache PDF Box library*.

When an image cannot be rendered, Oxygen XML Editor **Author** mode displays a warning message that contains the reason why this is happening. Possible causes include the following:

- The image is too large. Select the ***Show very large images** option*.
- The image format is not supported by default. It is recommended to *install the Java Advanced Imaging(JAI) Image I/O Tools plug-in*.

Scaling Images

Image dimension and scaling attributes are taken into account when an image is rendered. The following rules apply:

- If you specify only the width attribute of an image, the height of the image is proportionally applied.
- If you specify only the height attribute of an image, the width of the image is proportionally applied.
- If you specify width and height attributes of an image, both of them control the rendered image.
- If you want to scale both the width and height of an image proportionally, use the `scale` attribute.

Note: As a Java application, Oxygen XML Editor uses the *Java Advanced Imaging* API that provides a pluggable support for new image types. If you have an *ImageIO* library that supports additional image formats, just copy this library to the *[OXYGEN_INSTALL_DIR]*`/lib` directory.

Customize Oxygen XML Editor to Render CGM Images (Experimental Support)

Oxygen XML Editor provides experimental support for CGM 1.0 images.

Attention: Image hotspots are not supported.

Since this is an experimental support, some graphical elements might be missing from the rendered image.

The CGM rendering support is based on a third party library. In its free of charge variant it renders the images watermarked with the string `Demo`, painted across the panel. You can find more information about ordering the fully functioning version here: *http://www.bdaum.de/cgmpanel.htm*.

Follow this procedure to allow the rendering of CGM images in **Author** mode:

1. Download the `CGMPANEL.ZIP` from *http://www.bdaum.de/CGMPANEL.ZIP*.
2. Unpack the ZIP archive and copy the `cgmpanel.jar` into the *[OXYGEN_INSTALL_DIR]*`\lib` directory.
3. Restart the application.

Customize Oxygen XML Editor to Render PDF Images (Experimental Support)

Oxygen XML Editor provides experimental support for PDF images using the Apache PDFBox library.

To allow the rendering of PDF images in **Author** mode, follow this procedure:

1. Go to *http://pdfbox.apache.org/downloads.html* and download the pre-built PDFBox standalone binary *JAR* files: **`pdfbox-2.0.3.jar`**, **`fontbox-2.0.3.jar`**, and **`xmpbox-2.0.3.jar`**. Alternatively, you can use the `1.8.12` version of these files, as they have been tested and work properly.

   Note: It is not recommended to use `pdfbox-app-2.0.3.jar` file instead of the three specified files because it contains additional classes that may cause conflicts elsewhere in Oxygen XML Editor.
2. Create a subfolder called `pdfImageJars` in the *[OXYGEN_INSTALL_DIR]*`\lib` directory.
3. Copy the downloaded *JAR* libraries to that newly created subfolder.
4. Restart the application.

Customize Oxygen XML Editor to Render PSD Images

Oxygen XML Editor provides support for rendering PSD (Adobe Photoshop) images.

To allow the rendering of PSD images in **Author** mode, follow this procedure:

1. Download the following *JAR* files:
   - *http://search.maven.org/remotecontent?filepath=com/twelvemonkeys/common/common-lang/3.1.0/common-lang-3.1.0.jar*
   - *http://search.maven.org/remotecontent?filepath=com/twelvemonkeys/common/common-io/3.1.0/common-io-3.1.0.jar*
   - *http://search.maven.org/remotecontent?filepath=com/twelvemonkeys/common/common-image/3.1.0/common-image-3.1.0.jar*
   - *http://search.maven.org/remotecontent?filepath=com/twelvemonkeys/imageio/imageio-core/3.1.0/imageio-core-3.1.0.jar*

- *http://search.maven.org/remotecontent?filepath=com/twelvemonkeys/imageio/imageio-metadata/3.1.0/imageio-metadata-3.1.0.jar*
- *http://search.maven.org/remotecontent?filepath=com/twelvemonkeys/imageio/imageio-psd/3.1.0/imageio-psd-3.1.0.jar*

2. Copy the downloaded *JAR* libraries to the `[OXYGEN_INSTALL_DIR]\lib` directory.
3. Restart the application.

Customize Oxygen XML Editor to Render EPS and AI Images

Most EPS and AI image files include a preview picture of the content. Oxygen XML Editor tries to render this preview picture. The following scenarios are possible:

- The EPS or AI image does not include the preview picture. Oxygen XML Editor cannot render the image.
- The EPS image includes a TIFF preview picture.

  Note: Some newer versions of the TIFF picture preview are rendered in gray-scale.
- The AI image contains a JPEG preview picture. Oxygen XML Editor renders the image correctly.

Installing Java Advanced Imaging (JAI) Image I/O Tools Plugin

Certain special image types can be rendered in Oxygen XML Editor by using a Java Advanced Imaging (JAI) Image I/O Tools plugin.

How to Install JAI Image I/O Tools Plugin

To install this plug, follow this procedure:

1. Start Oxygen XML Editor and open the **Help** > **About** dialog box. Go to the **System properties** tab and look for the *java.runtime.name* and *java.home* properties. Keep their values for later use.
2. Download the JAI Image I/O kit corresponding to your operating system and Java distribution (found in the *java.runtime.name* property). A list of archived JAI distributions can be found at: *http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-java-client-419417.html*.

   Note: The JAI API is not the same thing as JAI Image I/O. Make sure you have installed the latter.
3. Run the installer. When the installation wizard displays the **Choose Destination Location** page, fill-in the **Destination Folder** field with the value of the *java.home* property. Continue with the installation procedure and follow the on-screen instructions.

OS X Workaround

There is no native implementation of the JAI Image I/O Tools plugin for OS X 10.5 and later. However, it has a Java implementation fallback that also works on OS X. Some of the image formats are not fully supported in this fallback mode, but at least the TIFF image format is known to be supported.

Use the following procedure for this OS X workaround:

1. Download a Linux (`tar.gz`) distribution of the JAI Image I/O Tools plugin. A list of archived JAI distributions can be found at: *http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-java-client-419417.html*.
2. In the `[OXYGEN_INSTALL_DIR]/lib` directory, create a directory named `endorsed` (`[OXYGEN_INSTALL_DIR]/lib/endorsed`).
3. Unpack the `tar.gz`. Copy the `clibwrapper_jiio.jar` and `jai_imageio.jar` files from its `lib` directory and paste them in the `[OXYGEN_INSTALL_DIR]/lib/endorsed` directory.
4. Restart the application and the JAI Image I/O support will be up and running.

Using Retina/HiDPI Images in Author Mode

Oxygen XML Editor provides support for Retina and HiDPI images through simple naming conventions. The higher resolution images are stored in the same images folder as the normal resolution images and they are identified by a scaling factor that is included in the name of the image files. For instance, images with a Retina scaling factor of 2 will include *@2x* in the name (for example, `myImage@2x.png`).

You can reference an image to style an element in a CSS by using the `url` function in the `content` property, as in the following example:

```
listItem:before{
   content: url('../img/myImage.png');
}
```

This would place the image that is loaded from the `myImage.png` file just before the `listItem` element. However, if you are using a Retina display (on a Mac), the icon looks a bit blurry as it automatically gets scaled, or if you are using an HiDPI display (on a Windows-based PC), the icon remains at the original size, thus it will look very small. To solve this rendering problem, you need to be able to reference both a *normal* DPI image and a *high* DPI image. However, referencing both of them from the CSS is not practical, as there is no standard way of doing this.

Starting with version 17, Oxygen XML Editor interprets the argument of the `url` function as key rather than a fixed URL. Therefore, when running on a system with a Retina or HiDPI display, Oxygen XML Editor will first try to find the image file that corresponds to the retina scaling factor. For instance, using the previous example, Oxygen XML Editor would first try to find `myImage@2x.png`. If this file is not found, it defaults back to the *normal* resolution image file (`myImage.png`).

Oxygen XML Editor also supports dark color themes. This means that the background of the editor area can be of a dark color and the foreground a lighter color. On a dark background, you may find it useful to invert the colors of images. Again, this can be done with simple naming conventions. If an image designed for a dark background is not found, the *normal* image is used.

Retina/HiDPI Naming Convention

Refer to the following table for examples of the Retina/HiDPI image naming convention that is used in Oxygen XML Editor:

Color Theme	Referred Image File	Double Density Image File	Triple Density Image File
normal	../img/myImage.png	../img/myImage@2x.png	../img/myImage@3x.png
dark	../img/ myImage_dark.png	../img/ myImage_dark@2x.png	../img/ myImage_dark@3x.png

Related Information:

Adding Retina/HiDPI Icons in a Framework on page 1196

Image Map Editor

Oxygen XML Editor includes an **Image Map Editor** that allows you to create hyperlinks in specific areas of an image that will link to various destinations. For example, an image that is a map of the seven continents may have a specific hyperlink for each continent that links to a resource that has information about the particular continent. The main purpose of an **image map** is to provide an easy way of linking various parts of an image without having to divide the image into separate image files.

The support for image maps in Oxygen XML Editor is available for images in DITA, DocBook, TEI, and XHTML document types (*frameworks*). To create an image map on an existing image and open the **Image Map Editor**, right-click the image and select **Image Map Editor**.

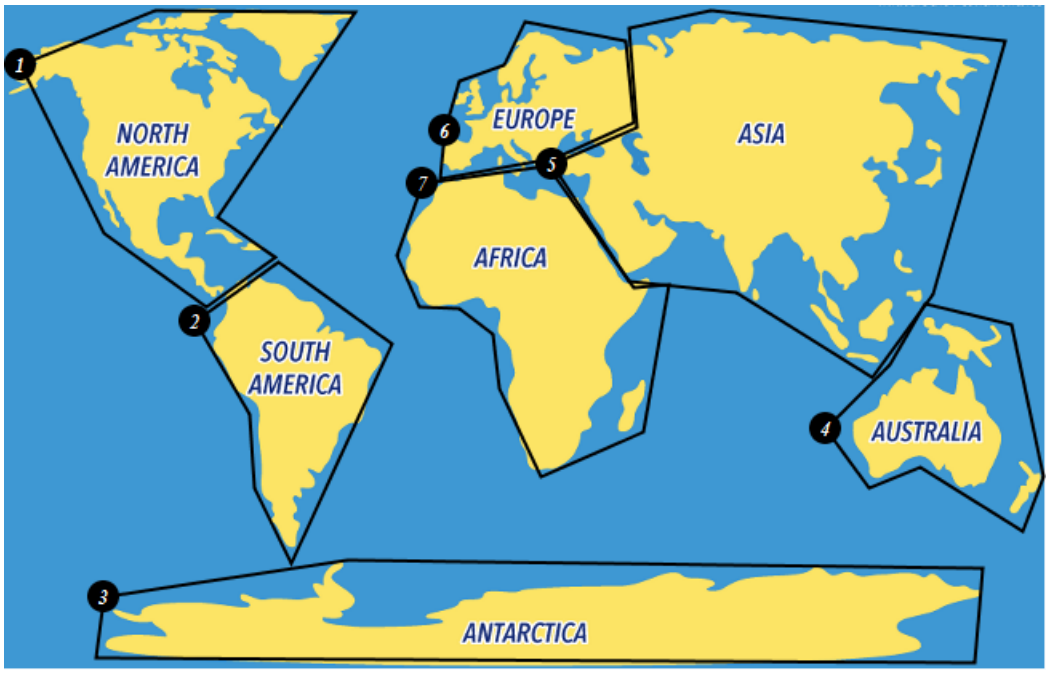


Figure 212: Image Map Rendered in Author Mode

Image Maps in DITA

Oxygen XML Editor includes support for **image maps** in DITA documents through the use of the `imagemap` element. This feature provides an easy way to create hyperlinks in various areas within an image without having to divide the image into separate image files. The visual **Author** editing mode includes an **Image Map Editor** that helps you to easily create and configure image maps.

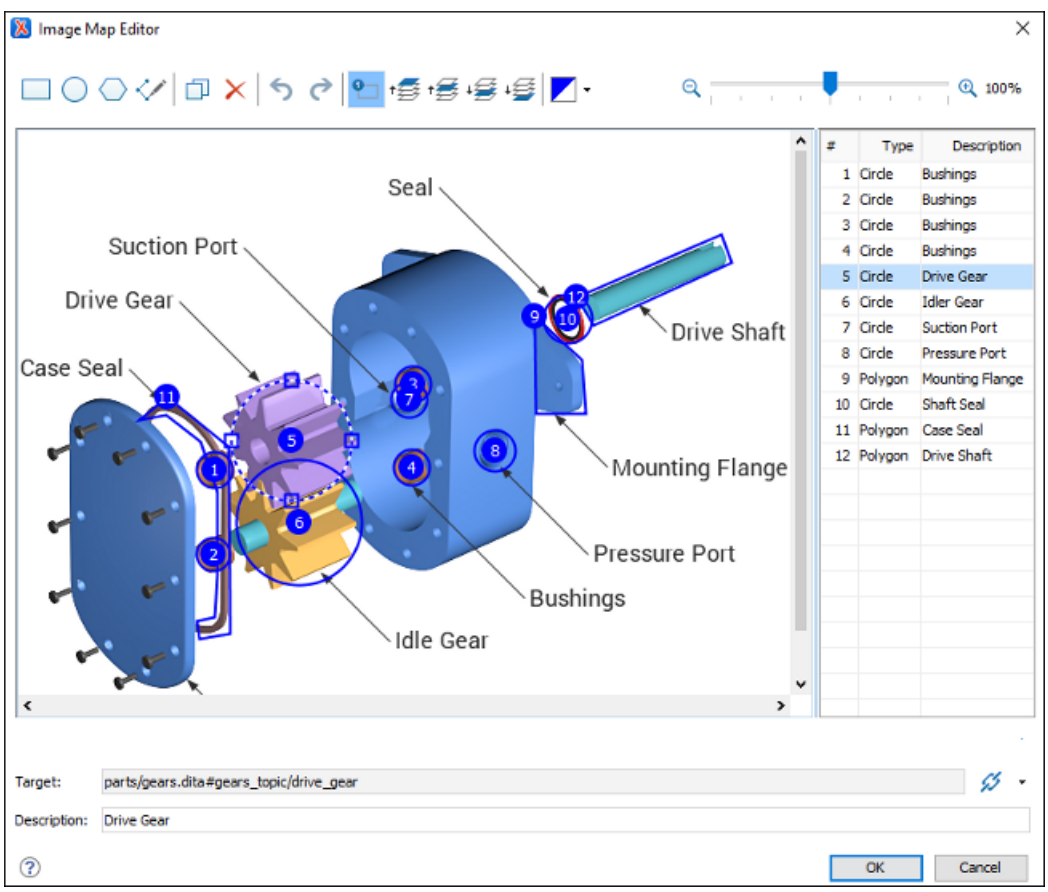


Figure 213: Image Map Editor in DITA

Image Map Editor Interface in DITA

The interface of the **Image Map Editor** consists of the following sections and actions:

Toolbar

New Rectangle

Use this button to draw a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

New Circle

Use this button to draw a circle over an area in the image. You can drag any of the four points to adjust the size of the circle.

New Polygon

Use this button to draw a polygon shape over an area in the image. This actions opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

New Free Form Shape

Use this button to draw a free form shape over an area in the image. After selecting this button, left-click anywhere in the image to place the first point of your shape. Then move the cursor to the location of the next desired point and left-click to place the next point, and so on. To complete the shape (area), click the first point again and a line will automatically be added from the last point that was added, or simply double-click the last point to automatically add the line from the last point back to the first.

Duplicate

Use this button to create a duplicate of the currently selected shape.

Delete

Use this button to delete the currently selected shape.

Undo

Use this button to undo the last action.

Redo

Use this button to redo the last action that was undone.

Show/Hide Numbers

Use this button to toggle between showing or hiding the numbers for the shapes.

Bring Shape to Front

Use this button to bring the currently selected shape forward to the top layer.

Bring Shape Forward

Use this button to bring the currently selected shape forward one layer.

Send Shape Backward

Use this button to send the currently selected shape back one layer.

Send Shape to Back

Use this button to send the currently selected shape back to the bottom layer.

Color Chooser

Use this drop-down menu to select a color scheme for the lines and numbers of the shapes.

Zoom Slider

Use this slider to zoom the image in or out in the main image pane.

Image Pane

This main image pane is where you work with shapes to add hyperlinks to multiple areas within an image. Use the mouse to move shapes around in the image pane. You can hold down the **Ctrl** key to select multiple shapes and then move them simultaneously. You can also drag the points of a selected shape to adjust its size and shape. It is easy to see which shape is selected in this image pane because the border of the selected shape changes from a solid line to a dotted one.

Contextual Menu Actions Available in the Image Pane

You can right-click the shapes, points, or anywhere in the Image Pane to invoke the contextual menu where the following actions are available:

Add Point

Adds a point to *Polygon* or *Free Form* shapes.

Remove Point

Removes the current point from *Polygon* or *Free Form* shapes.

Duplicate

Create a duplicate of the currently selected shape.

Delete

Delete the currently selected shape.

New Rectangle

Creates a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

New Circle

Creates a circle over an area in the image. You can drag any of the four points to adjust the size of the circle.

New Polygon

Creates a polygon shape over an area in the image. This actions opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

Undo

Use this action to undo the last action.

Redo

Use this action to redo the last action that was undone.

Shape Table

The table at the right of the *Image Pane* is a sequential list of all the areas (shapes) that have been added in the image. It shows their number, type, and description (if one has been added). If you select one of the entries in the table, the corresponding shape will be selected in the *Image Pane*.

Properties

Target

Allows you to choose the target resource that you want the selected area (shape) to be linked to. You can enter the path to the target in the text field but the easiest way to select a target is to use the **Link** drop-down menu to the right of the text field. You can choose between the following types of links: **Cross Reference**, **File Reference**, or **Web Link**. All three types will open a dialog box that allows you to define the target resource. This linking process is similar to the normal process of *inserting links in DITA* by using the identical **Link** drop-down menu from the main toolbar.

When you click **OK** to finalize your changes in the **Image Map Editor**, an `xref` element will be inserted with either an `href` attribute or a `keyref` attribute. Additional attributes may also be inserted and their values

depend on the target and the type of link. For details about the three types of links and their dialog boxes, see *Inserting a Link in Oxygen XML Editor* on page 1675.

Description

You can enter an optional description for the selected area (shape) that will be displayed in the ***Image Map Details** section* in **Author** mode and as a tooltip message when the end user hovers over the hyperlink in the output.

How to Create an Image Map in DITA

To create an image map on an existing image in a DITA document, follow these steps:

1. Right-click the image and select **Image Map Editor**.

   Result: This action will apply an *image map* to the current image and open the **Image Map Editor** dialog box.
2. Add hyperlinks to the image by selecting one of the shape buttons (**New Rectangle**, **New Circle**, or **New Polygon**).
3. Move the shape to the desired area in the image and drag any of the points on the shape to adjust its size or form. You can use the *other buttons on the toolbar* to adjust its layer and color, or to perform other editing actions.
4. With the shape selected, use one of the *linking options* in the **Link** drop-down menu to select a target resource (or enter its path in the ***Target*** text field).
5. (Optional) Enter a ***Description*** for the selected area (shape).
6. If you want to add more hyperlinks to the image, select a shape button again and repeat the appropriate steps.
7. When you are finished creating hyperlinks, click **OK** to process your changes.

Result: The *image map* is applied on the image and the appropriate elements and attributes are automatically added. In **Author** mode, the image map is now rendered over the image. If the image includes an `alt` element, its value will be displayed under the image. The following two buttons will also now be available under the image in **Author** mode:

- **Image Map Editor** - Click this button to open the **Image Map Editor**.
- **Image Map Details** - Click this button to expand a section that displays the details of the image map and allows you to change the shape and coordinates of the hyperlinked areas. Keep in mind that if you change the shape in this section, you also need to add or remove coordinates to match the requirements of the new shape.

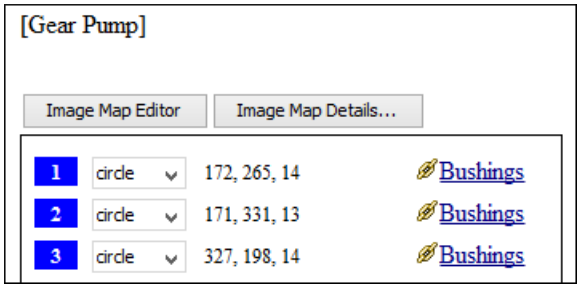


Figure 214: Image Map Details

How to Edit an Existing Image Map in DITA

To edit an existing image map, use any of the following methods:

- Simply double-click the image.
- Right-click the image and select **Image Map Editor**.
- Click the **Image Map Editor** button below the image.

All three methods open the **Image Map Editor** where you can make changes to the image map with a visual editor. You can also make changes to the XML structure of the image map in the **Text** editing mode.

You can also click the **Image Map Details** button below the image to expand a section that displays the details of the image map and allows you to change the shape and coordinates of the hyperlinked areas. Keep in mind that if you change the shape in this section, you also need to add or remove coordinates to match the requirements of the new shape.

Overlapping Areas

If shapes overlap one another in the **Image Map Editor**, the one on the top layer takes precedence. The number shown inside each shape represent its layer (if the numbers are not displayed, click the **Show/Hide Numbers** button on the ***Image Map Editor*** *toolbar*). To change the layer order for a shape, use the layer buttons on the ***Image Map Editor*** *toolbar* (, , ,).

If you insert a shape and all of its coordinates are completely inside another shape, the **Image Map Editor** will display a warning to let you know that the shape is entirely covered by a bigger shape. Keep in mind that if a shape is completely inside another shape, its hyperlink will only be accessible if its layer is on top of the bigger shape.

Warning: PDF output is limited to rectangular shaped image map objects. Therefore, if your image contains circles or polygons, those objects will be redrawn as rectangles in the PDF output. Keep in mind that this might affect overlaps in the output.

Related Information:

DITA 'imagemap' Element Specifications

Adding Images in DITA Topics on page 1625

Image Maps in DocBook

Oxygen XML Editor includes support for **image maps** in DocBook documents through the use of the `areaspec` element. This feature provides an easy way to create hyperlinks in various parts of an image without having to divide the image into separate image files. The visual **Author** editing mode includes an **Image Map Editor** that helps you to easily create and configure image maps.

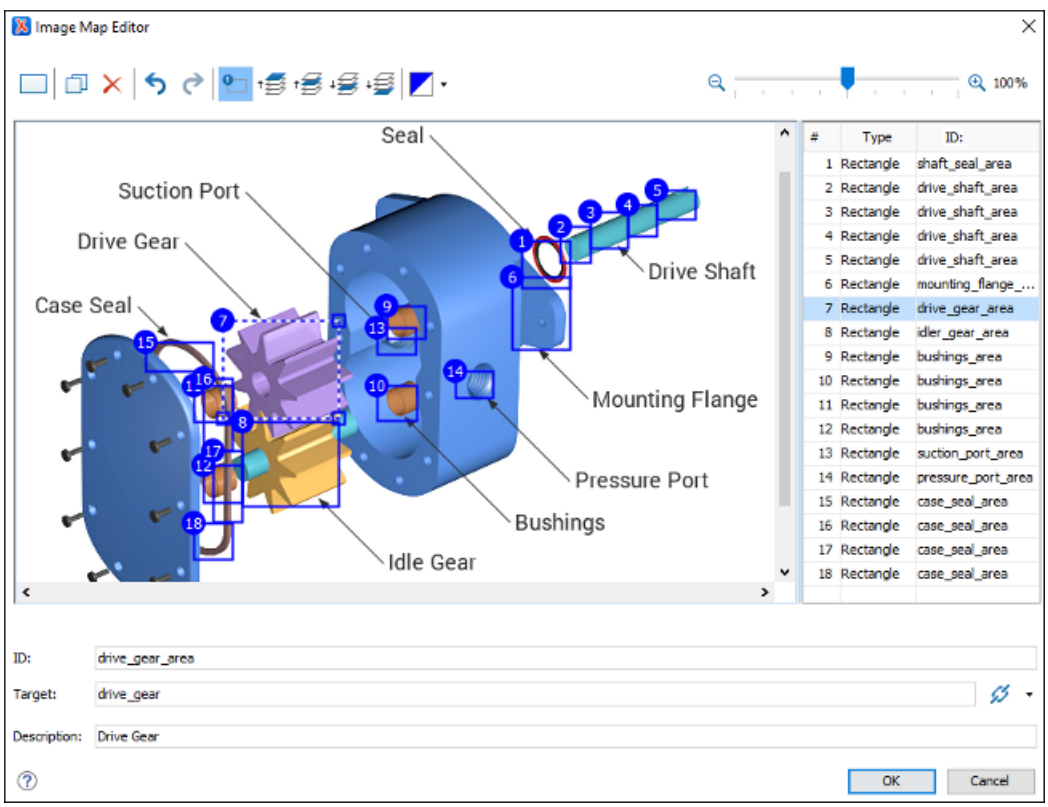


Figure 215: Image Map Editor in DocBook

Image Map Editor Interface in DocBook

The interface of the **Image Map Editor** consists of the following sections and actions:

Toolbar

New Rectangle

Use this button to draw a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

Duplicate

Use this button to create a duplicate of the currently selected shape.

Delete

Use this button to delete the currently selected shape.

Undo

Use this button to undo the last action.

Redo

Use this button to redo the last action that was undone.

Show/Hide Numbers

Use this button to toggle between showing or hiding the numbers for the shapes.

Bring Shape to Front

Use this button to bring the currently selected shape forward to the top layer.

Bring Shape Forward

Use this button to bring the currently selected shape forward one layer.

Send Shape Backward

Use this button to send the currently selected shape back one layer.

Send Shape to Back

Use this button to send the currently selected shape back to the bottom layer.

Color Chooser

Use this drop-down menu to select a color scheme for the lines and numbers of the shapes.

Zoom Slider

Use this slider to zoom the image in or out in the main image pane.

Image Pane

This main image pane is where you work with shapes to add hyperlinks to multiple areas within an image. Use the mouse to move shapes around in the image pane. You can hold down the **Ctrl** key to select multiple shapes and then move them simultaneously. You can also drag the points of a selected shape to adjust its size and shape. It is easy to see which shape is selected in this image pane because the border of the selected shape changes from a solid line to a dotted one.

Contextual Menu Actions Available in the Image Pane

You can right-click the shapes, or anywhere in the Image Pane to invoke the contextual menu where the following actions are available:

Duplicate

Create a duplicate of the currently selected shape.

Delete

Delete the currently selected shape.

New Rectangle

Creates a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

Undo

Use this action to undo the last action.

Redo

Use this action to redo the last action that was undone.

Shape Table

The table at the right of the *Image Pane* is a sequential list of all the areas (shapes) that have been added in the image. It shows their number, type, and ID. If you select one of the entries in the table, the corresponding shape will be selected in the *Image Pane*.

Properties

ID

The identifier for the selected area. This will become the value of the `xml:id` attribute for the particular `area` element.

Target

Allows you to choose the target resource that you want the selected area to be linked to. You can enter the path to the target in the text field but the easiest way to select a target is to use the **Link** drop-down menu to the right of the text field. You can choose between the following types of links: **Cross Reference** or **Web Link**. Both types open a dialog box that allows you to select the target resource and it is inserted as the value of an `xlink:href` attribute.

Description

You can enter an optional description for the selected area that will be displayed in the ***Image Map Details section*** in **Author** mode and as a tooltip message when the end user hovers over the hyperlink in the output.

How to Create an Image Map in DocBook

To create an image map on an existing image in a DocBook document, follow these steps:

1. Right-click the image and select **Image Map Editor**.

   Result: This action will apply an *image map* to the current image and open the **Image Map Editor** dialog box.
2. Add hyperlinks to the image by selecting the **New Rectangle** button.
3. Move the shape to the desired area in the image and drag any of the points on the shape to adjust its size or form. You can use the *other buttons on the toolbar* to adjust its layer and color, or to perform other editing actions.
4. With the shape selected, enter an ***ID*** and use one of the *linking options* in the **Link** drop-down menu to select a target resource (or enter its path in the ***Target*** text field).
5. (Optional) Enter a ***Description*** for the selected area (shape).
6. If you want to add more hyperlinks to the image, select **New Rectangle** button again and repeat the appropriate steps.
7. When you are finished creating hyperlinks, click **OK** to process your changes.

Result: The *image map* is applied on the image and the appropriate elements and attributes are automatically added. In **Author** mode, the image map is now rendered over the image. If the image includes an `alt` element, its value will be displayed above the image. The following two buttons will also now be available at the top of the image in **Author** mode:

- **Image Map Editor** - Click this button to open the **Image Map Editor**.
- **Image Map Details** - Click this button to expand a section that displays the details of the image map.

How to Edit an Existing Image Map in DocBook

To edit an existing image map, use any of the following methods:

- Simply double-click the image.
- Right-click the image and select **Image Map Editor**.
- Click the **Image Map Editor** button below the image.

All three methods open the **Image Map Editor** where you can make changes to the image map with a visual editor. You can also make changes to the XML structure of the image map in the **Text** editing mode.

You can also click the **Image Map Details** button above the image to expand a section that displays the details of the image map and allows you to change the coordinates and IDs of the hyperlinked areas.

Note: If you want to link a set of related `area` elements, you can use `areaset` elements. To add `areaset` elements, and `area` elements to the *areasets*, switch to **Text** mode and insert them manually.

Overlapping Areas

If shapes overlap one another in the **Image Map Editor**, the one on the top layer takes precedence. The number shown inside each shape represent its layer (if the numbers are not displayed, click the **Show/Hide Numbers** button on the ***Image Map Editor*** *toolbar*). To change the layer order for a shape, use the layer buttons on the ***Image Map Editor*** *toolbar* (, , ,).

If you insert a shape and all of its coordinates are completely inside another shape, the **Image Map Editor** will display a warning to let you know that the shape is entirely covered by a bigger shape. Keep in mind that if a shape is completely inside another shape, its hyperlink will only be accessible if its layer is on top of the bigger shape.

Related Information:
DocBook 'areaspec' Element Specifications

Image Maps in TEI

Oxygen XML Editor includes support for **image maps** in TEI documents through the use of the `facsimile` element. In TEI documents, this feature provides an easy way to create areas (using `zone` elements) in an image where the end user can hover or click to retrieve more information about that particular area of the image. The visual **Author** editing mode includes an **Image Map Editor** that helps you to easily create the areas in the image.

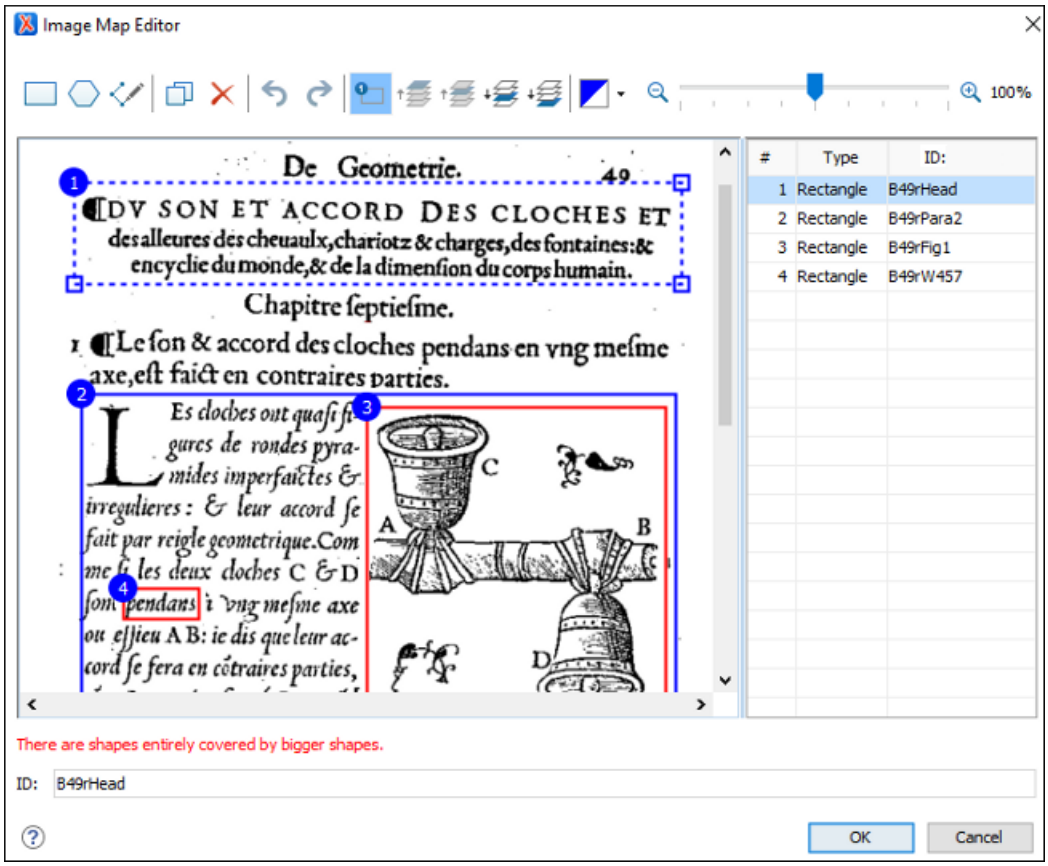


Figure 216: Image Map Editor in TEI

Image Map Editor Interface in TEI

The interface of the **Image Map Editor** consists of the following sections and actions:

Toolbar

New Rectangle

Use this button to draw a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

New Polygon

Use this button to draw a polygon shape over an area in the image. This actions opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

New Free Form Shape

Use this button to draw a free form shape over an area in the image. After selecting this button, left-click anywhere in the image to place the first point of your shape. Then move the cursor to the location of the next desired point and left-click to place the next point, and so on. To complete the shape (area), click the first point again and a line will automatically be added from the last point that was added, or simply double-click the last point to automatically add the line from the last point back to the first.

Duplicate

Use this button to create a duplicate of the currently selected shape.

Delete

Use this button to delete the currently selected shape.

Undo

Use this button to undo the last action.

Redo

Use this button to redo the last action that was undone.

Show/Hide Numbers

Use this button to toggle between showing or hiding the numbers for the shapes.

Bring Shape to Front

Use this button to bring the currently selected shape forward to the top layer.

Bring Shape Forward

Use this button to bring the currently selected shape forward one layer.

Send Shape Backward

Use this button to send the currently selected shape back one layer.

Send Shape to Back

Use this button to send the currently selected shape back to the bottom layer.

Color Chooser

Use this drop-down menu to select a color scheme for the lines and numbers of the shapes.

Zoom Slider

Use this slider to zoom the image in or out in the main image pane.

Image Pane

This main image pane is where you work with shapes to add areas (zones) within an image. Use the mouse to move shapes around in the image to the desired area and drag the points on a selected shape to adjust its size and shape. It is easy to see which shape is selected in this image pane because the border of the selected shape changes from a solid line to a dotted line.

Contextual Menu Actions Available in the Image Pane

You can right-click the shapes, points, or anywhere in the Image Pane to invoke the contextual menu where the following actions are available:

Add Point

Adds a point to *Polygon* or *Free Form* shapes.

Remove Point

Removes the current point from *Polygon* or *Free Form* shapes.

Duplicate

Create a duplicate of the currently selected shape.

Delete

Delete the currently selected shape.

New Rectangle

Creates a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

New Polygon

Creates a polygon shape over an area in the image. This actions opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

Undo

Use this action to undo the last action.

Redo

Use this action to redo the last action that was undone.

Shape Table

The table at the right of the *Image Pane* is a sequential list of all the areas (shapes) that have been added in the image. It shows their number, type, and ID. If you select one of the entries in the table, the corresponding shape will be selected in the *Image Pane*.

Properties

ID

The identifier for the selected area. This will become the value of the `xml:id` attribute for the particular `zone` element. When you insert a new zone, a unique ID is automatically generated and displayed in this field. However, you can change this value if you want to.

How to Create an Image Map in TEI

To create an image map on an existing image in a TEI document, follow these steps:

1. The image (`graphic`) must be inside a `facsimile` element to support the **Image Map Editor** feature.
2. Right-click the image and select **Image Map Editor**.

   Result: This action will apply an *image map* to the current image and open the **Image Map Editor** dialog box.
3. Add areas (zones) in the image by selecting one of the shape buttons (**New Rectangle** or **New Polygon**).
4. Move the shape to the desired area in the image and drag any of the points on the shape to adjust its size or form. You can use the *other buttons on the toolbar* to adjust its layer and color, or to perform other editing actions.
5. With the shape selected, enter an ***ID***.
6. If you want to add more areas (zones) to the image, select a shape button again and repeat the appropriate steps.
7. When you are finished, click **OK** to process your changes.

Result: The *image map* is applied on the image and the appropriate elements and attributes are automatically added. In **Author** mode, the image map is now rendered over the image and the following two buttons will now be available at the bottom of the image:

- **Image Map Editor** - Click this button to open the **Image Map Editor**.
- **Image Map Details** - Click this button to expand a section that displays the details of the image map.

How to Edit an Existing Image Map in TEI

To edit an existing image map, use any of the following methods:

- Simply double-click the image.
- Right-click the image and select **Image Map Editor**.
- Click the **Image Map Editor** button below the image.

All three methods open the **Image Map Editor** where you can make changes to the image map with a visual editor. You can also make changes to the XML structure of the image map in the **Text** editing mode.

You can also click the **Image Map Details** button below the image to expand a section that displays the details of the image map and allows you to change the coordinates and IDs of the hyperlinked areas.

Restriction: Currently, if `zone` elements contain additional content (such as text or comments) and you edit the image map, the **Image Map Editor** does not preserve the additional content. Therefore, if you do need to insert additional content inside the `zone` elements, you should do so after the image map has been created and finalized. Subsequent changes to the image map should then be done in **Text** mode.

Overlapping Areas

If shapes overlap one another in the **Image Map Editor**, the one on the top layer takes precedence. The number shown inside each shape represent its layer (if the numbers are not displayed, click the **Show/Hide Numbers** button on the ***Image Map Editor*** *toolbar*). To change the layer order for a shape, use the layer buttons on the ***Image Map Editor*** *toolbar* (, , ,).

If you insert a shape and all of its coordinates are completely inside another shape, the **Image Map Editor** will display a warning to let you know that the shape is entirely covered by a bigger shape. Keep in mind that if a shape is completely inside another shape, its hyperlink will only be accessible if its layer is on top of the bigger shape.

Warning: PDF output is limited to rectangular shaped image map objects. Therefore, if your image contains circles or polygons, those objects will be redrawn as rectangles in the PDF output. Keep in mind that this might affect overlaps in the output.

Related Information:

TEI 'facsimile' Element Specifications

Image Maps in XHTML

Oxygen XML Editor includes support for **image maps** in XHTML documents. This feature provides an easy way to create hyperlinks in various parts of an image without having to divide the image into separate image files. In HTML, an image (in the form of an `img` element) may be associated with an image map (in the form of a `map` element) by specifying a `usemap` attribute on the `img` element. The visual **Author** editing mode includes an **Image Map Editor** that helps you to easily create and configure image maps.

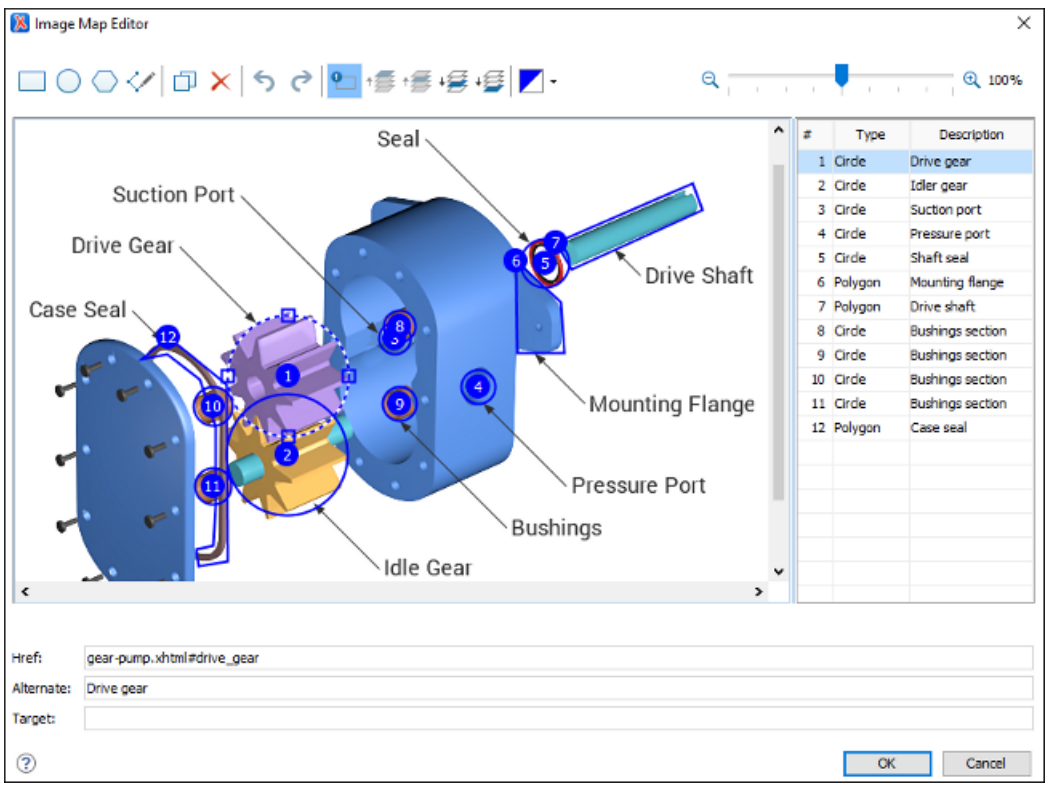


Figure 217: Image Map Editor in XHTML

Image Map Editor Interface in XHTML

The interface of the **Image Map Editor** consists of the following sections and actions:

Toolbar

New Rectangle

Use this button to draw a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

New Circle

Use this button to draw a circle over an area in the image. You can drag any of the four points to adjust the size of the circle.

New Polygon

Use this button to draw a polygon shape over an area in the image. This actions opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

New Free Form Shape

Use this button to draw a free form shape over an area in the image. After selecting this button, left-click anywhere in the image to place the first point of your shape. Then move the cursor to the location of the next desired point and left-click to place the next point, and so on. To complete the shape (area), click the first point again and a line will automatically be added from the last point that was added, or simply double-click the last point to automatically add the line from the last point back to the first.

Duplicate

Use this button to create a duplicate of the currently selected shape.

Delete

Use this button to delete the currently selected shape.

Undo

Use this button to undo the last action.

Redo

Use this button to redo the last action that was undone.

Show/Hide Numbers

Use this button to toggle between showing or hiding the numbers for the shapes.

Bring Shape to Front

Use this button to bring the currently selected shape forward to the top layer.

Bring Shape Forward

Use this button to bring the currently selected shape forward one layer.

Send Shape Backward

Use this button to send the currently selected shape back one layer.

Send Shape to Back

Use this button to send the currently selected shape back to the bottom layer.

Color Chooser

Use this drop-down menu to select a color scheme for the lines and numbers of the shapes.

Zoom Slider

Use this slider to zoom the image in or out in the main image pane.

Image Pane

This main image pane is where you work with shapes to add hyperlinks to multiple areas within an image. Use the mouse to move shapes around in the image pane. You can hold down the **Ctrl** key to select multiple shapes and then move them simultaneously. You can also drag the points of a selected shape to adjust its size and shape. It is easy to see which shape is selected in this image pane because the border of the selected shape changes from a solid line to a dotted one.

Contextual Menu Actions Available in the Image Pane

You can right-click the shapes, points, or anywhere in the Image Pane to invoke the contextual menu where the following actions are available:

Add Point

Adds a point to *Polygon* or *Free Form* shapes.

Remove Point

Removes the current point from *Polygon* or *Free Form* shapes.

Duplicate

Create a duplicate of the currently selected shape.

Delete

Delete the currently selected shape.

New Rectangle

Creates a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

New Circle

Creates a circle over an area in the image. You can drag any of the four points to adjust the size of the circle.

New Polygon

Creates a polygon shape over an area in the image. This actions opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

Undo

Use this action to undo the last action.

Redo

Use this action to redo the last action that was undone.

Shape Table

The table at the right of the *Image Pane* is a sequential list of all the areas (shapes) that have been added in the image. It shows their number, type, and description (value of the **Alternative** property). If you select one of the entries in the table, the corresponding shape will be selected in the *Image Pane*.

Properties

Href

Specifies the hyperlink target for the selected area. This will become the value of the `href` attribute for the particular `area` element. The possible values are:

- **An Absolute URL** - A URL of another web site (for example, `http://www.example.com/index.htm`).
- **A Relative URL** - A link to a file within your web site (for example, `index.htm`).
- **An Element** - A link to the ID of an element within the page (for example, `#top`).
- **Other Protocols** - A specified path using other protocols (such as `https://`, `ftp://`, `mailto:`, `file:`).

- **A Script** - A link to a script (for example, `javascript:alert('Hello');`)

Alternate

The description for the selected area. The value is inserted in an `alt` attribute in the particular `area` element. This is a required attribute to present a text alternative for browsers that do not display images.

Target

Specifies where to open the linked resource. The allowed values are:

- **_blank** - Opens the linked resource in a new window or tab.
- **_self** - Opens the linked resource in the same frame as it was clicked.
- **_parent** - Opens the linked resource in the full body of the window.
- **framename** - Opens the linked resource in the named frame.

How to Create an Image Map in XHTML

To create an image map on an existing image in an XHTML document, follow these steps:

1. Right-click the image and select **Image Map Editor**.

   Result: This action will apply an *image map* to the current image and open the **Image Map Editor** dialog box.
2. Add hyperlinks to the image by selecting one of the shape buttons (**New Rectangle**, **New Circle**, or **New Polygon**).
3. Move the shape to the desired area in the image and drag any of the points on the shape to adjust its size or form. You can use the *other buttons on the toolbar* to adjust its layer and color, or to perform other editing actions.
4. With the shape selected, specify the hyperlink target in the ***Href** field* and enter a description for the selected area in the ***Alternate** field*.
5. (Optional) Specify where the hyperlink resource will be opened in the ***Target** field*.
6. If you want to add more hyperlinks to the image, select a shape button again and repeat the appropriate steps.
7. When you are finished creating hyperlinks, click **OK** to process your changes.

Result: The *image map* is applied on the image and the appropriate elements and attributes are automatically added. In **Author** mode, the image map is now rendered over the image and its properties are displayed in a section below the image.

How to Edit an Existing Image Map in XHTML

To edit an existing image map, use any of the following methods:

- Simply double-click the image.
- Right-click the image and select **Image Map Editor**.
- Click the **Image Map Editor** button below the image.

All three methods open the **Image Map Editor** where you can make changes to the image map with a visual editor. You can also make changes to the XML structure of the image map in the **Text** editing mode.

In **Author** mode, the details of the image map are also displayed below the image and you can edit the description, href, shape, and coordinates of the hyperlinked areas. Keep in mind that if you change the shape in this section, you also need to add or remove coordinates to match the requirements of the new shape.

Overlapping Areas

If shapes overlap one another in the **Image Map Editor**, the one on the top layer takes precedence. The number shown inside each shape represent its layer (if the numbers are not displayed, click the **Show/Hide Numbers** button on the ***Image Map Editor** toolbar*). To change the layer order for a shape, use the layer buttons on the ***Image Map Editor** toolbar* (, , ,).

If you insert a shape and all of its coordinates are completely inside another shape, the **Image Map Editor** will display a warning to let you know that the shape is entirely covered by a bigger shape. Keep in mind that if a

shape is completely inside another shape, its hyperlink will only be accessible if its layer is on top of the bigger shape.

Warning: PDF output is limited to rectangular shaped image map objects. Therefore, if your image contains circles or polygons, those objects will be redrawn as rectangles in the PDF output. Keep in mind that this might affect overlaps in the output.

Related Information:

HTML Image Map Specifications

Adding Video, Audio, and Embedded HTML Resources

You can insert references to media resources (such as videos, audio clips, or embedded HTML frames) in your DITA, DocBook, or XHTML topics. The media resources can be played directly in **Author** mode and in all HTML5-based outputs. There is a toolbar button ( ) that allows you to insert and configure a reference to the media resource. You can also drag media files from your system explorer or the ***Project*** *view* and drop them into your documents (or copy and paste them).

Table 9: Supported Media Types

Media	Description	Type	Supported Size Properties
mp3	Moving Picture Experts Group Layer-3 Audio	audio	Width
wav	Windows Wave	audio	Width
pcm	Pulse Code Modulation	audio	Width
m4a	Moving Picture Experts Group Layer-4 Audio	audio	Width
aif	Audio Interchange Format	audio	Width
mp4	Moving Picture Experts Group Layer-4 Video	video	Width & Height
flv	Flash Video	video	Width & Height
m4v	Itunes Video File	video	Width & Height
avi	Audio Video Interleaved	video	Width & Height
embedded video (such as YouTube or Vimeo)	Embedded Iframe Code	iframe	Width & Height

Adding a Media Resource

To insert a media resource in a document, use the following procedure:

1. Place the cursor at the location where you want the media resource.
2. Select the **Insert Media Resource** action from the toolbar. A **Chose Media** dialog box appears.

   Note: You can also drag media files from your system explorer or the ***Project*** *view* and drop them into your documents (or copy and paste them).
3. Select the URL for the media resource and click **Ok**.

Result in Author Mode: A reference to the specified media object is inserted and rendered in **Author** mode so that it can be played directly from there.

Result in Output: In the publishing stage, the media object is converted to an HTML5 element so that it can be rendered properly and played in all HTML5-based outputs.

Embedding HTML Content in DITA Topics

The DITA Open Toolkit that comes bundled with Oxygen XML Editor includes a pre-installed *plugin* that allows you to embed well-formed HTML content directly in a DITA topic.

For example, suppose you wanted to embed a YouTube video directly in a DITA topic.

The DITA topic would look like this:

```
....
<foreign outputclass="html-embed"><![CDATA[
                <iframe width="420" height="315"
                  src="https://www.youtube.com/embed/qepRkQxhTX"
                  frameborder="0" allowfullscreen="true">
                </iframe>
                ]]></foreign>
....
```

The converted HTML output would look like this:

```
...........
<iframe width="420" height="315" src="https://www.youtube.com/embed/qepRkQxhTX"
frameborder="0" allowfullscreen="true">
</iframe>
............
```

The *plugin* is also available on the *oxygenxml GitHub projects page*.

Related Information:

Adding Video and Audio Objects in DITA WebHelp Output on page 968

Editing MathML Notations

The **Author** editor includes a built-in editor for *MathML* notations. To start the *MathML* editor, double-click a *MathML* notation (for embedded notations, you can also select the **Edit Equation** action from its contextual menu). In the *MathML* editor you can edit the mathematical symbols of a *MathML* notation. You can open a *MathML* file of your current project directly in the *MathML* editor. To do this, select **Open with** > **MathML editor** from the contextual menu in the ***Project*** *view*.

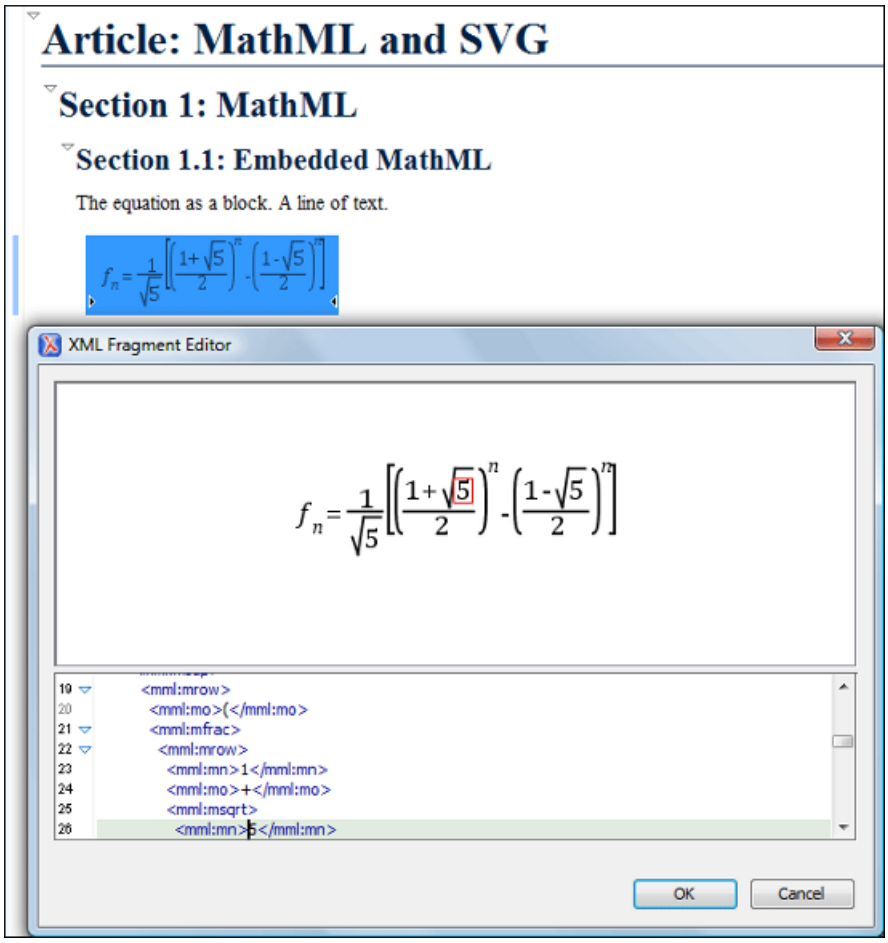


Figure 218: Default *MathML* Editor

The font size and font family that is used for the equations is based upon the context in which the MathML equation appears. To configure the minimum font size of the equation, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Editor** > **Edit modes** > **Author** > **MathML**.

Configure the MathFlow Editor

The *MathFlow* Components product can replace the default *MathML* editor with a specialized *MathML* editor. You have to *purchase a MathFlow Component from Design Science* and configure it in Oxygen XML Editor with the following procedure:

1. Install *MathFlow* by using the *Universal* installer (for versions prior to 2.1, use the *MathFlow SDK*).
2. Set the path to the *MathFlow* install folder *in the **MathML** preferences page*.
3. Set the path to the *MathFlow* license file *in the **MathML** preferences page*.

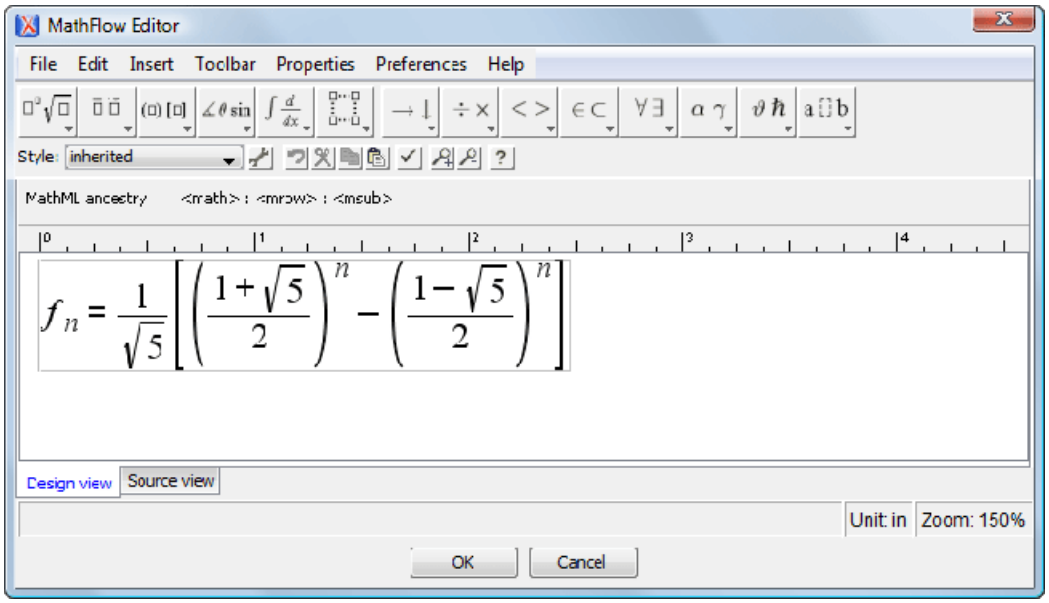


Figure 219: Default MathFlow Editor

MathML Equations in HTML Output

Currently, only **Firefox** can render **MathML** equations embedded in the **HTML** code. *MathJax* is a solution to properly view MathML equations embedded in **HTML** content in a variety of browsers.

If you have DocBook or DITA content that has embedded **MathML** equations and you want to properly view the equations in published HTML output types (WebHelp, CHM, EPUB, etc.), you need to add a reference to the MathJax script in the **head** element of all HTML files that have the equation embedded.

For example:

```
<script type="text/javascript"
  src="https://cdn.mathjax.org/mathjax/latest/MathJax.js?config=TeX-AMS-MML_HTMLorMML">
</script>
```

For DITA documents, you can also use the following procedure:

1. *Edit the DITA Map WebHelp transformation scenario* and open the **Parameters** tab.
2. Set the following parameter to point to an XML resource file that contains your script, depending on your type of WebHelp system.
   - **WebHelp Responsive Systems** - Set the `webhelp.fragment.head` parameter to point to your resource file.
   - **WebHelp Classic Systems** - Set the `webhelp.head.script` parameter to point to your resource file.
3. Run the transformation scenario.

Result: The equation should now be properly rendered in other browsers, such as Edge, IE, or Chrome.

Refreshing the Content

On occasion you may need to reload the content of the document from the disk or reapply the CSS. This can be performed by using the **Reload** action.

To refresh the content of the referenced resources you can use the **Refresh references** action. However, this action will not refresh the expanded external entities, for which you will need to use the **Reload** action.

Generating IDs for Elements in Author Mode

Oxygen XML Editor allows you to manually assign or edit values of `id` attributes in **Author** mode by using the ***Attributes View*** or an *in-place attribute editor*. Oxygen XML Editor also includes mechanisms to generate ID values for elements, either on-request or automatically, in DITA, DocBook, or TEI documents.

Generate IDs On-Request

You can generate ID values for specific elements on-request. To do so, select the element for which you want to generate an ID (or place the cursor inside the element) and select the **Generate IDs** action from the contextual menu or the *framework*-specific menu (**DITA**, **DocBook**, or **TEI**). This action generates a unique ID for the current element. If you invoke the action on a block of selected content, the action will generate IDs for all top-level elements and elements that are listed in the ***ID Options** dialog box* that are found in the current selection.

Note: The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `id` attribute.

Automatically Generate IDs

Oxygen XML Editor includes an option to automatically add unique ID values to certain elements when they are created in **Author** mode. The **Auto generate IDs for elements** option can be found in the ***ID Options** dialog box* that is displayed when you select the **ID Options** action from the *framework*-specific menu (**DITA**, **DocBook**, or **TEI**). If this **Auto generate IDs for elements** option is selected, Oxygen XML Editor automatically generates unique ID values for elements that are listed in this dialog box. You can use this dialog box to customize the format of the ID values and choose which elements will have their ID values automatically generated (for example, you can customize the list of elements to include those that you most often need to identify).

ID Options Dialog Box

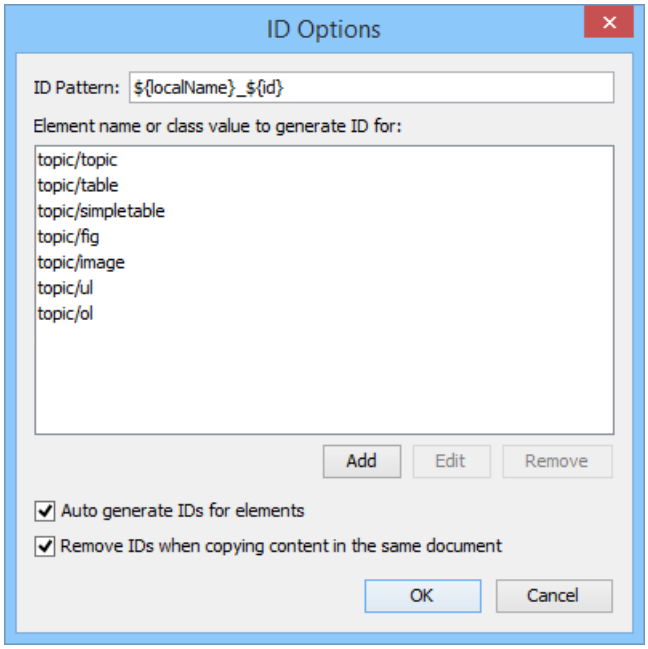


Figure 220: ID Options Dialog Box

The **ID Options** dialog box allows you to configure the following options in regards to generating ID values:

ID Pattern

The pattern for the ID values that will be generated. This text field can be customized using constant strings or any of the Oxygen XML Editor *Editor Variables*.

Element name or class value to generate ID for

The elements for which ID values will be generated, specified using class attribute values. To customize the list, use the **Add**, **Edit**, or **Remove** buttons.

Auto generate IDs for elements

If selected, Oxygen XML Editor will automatically generate unique IDs for the elements listed in this dialog box when they are created in **Author** mode.

Remove IDs when copying content in the same document

When copying and pasting content in the same document, this option allows you to control whether or not pasted elements that are listed in this dialog box should retain their existing IDs. To retain the element IDs, deselect this option.

Note: This option does not have an effect on content that is *cut* and *pasted*.

Duplicating Elements with Existing IDs

If you duplicate elements with existing IDs (for example, through copy/paste or drag/drop actions), all IDs are removed at the resolution of the operation. However, you can use the options in the **ID Options** dialog box to change this behavior. The options in this dialog box affect duplicated elements with existing IDs in the following ways:

Note: Only the elements listed in this dialog box are affected by these options. Therefore, if you want to use these options to preserve IDs or generate new ones, you must first add the elements to be duplicated to the list in this dialog box.

- If the ***Auto generate IDs for elements*** *option* is selected and you duplicate elements with existing IDs, Oxygen XML Editor assigns new, unique ID values to the duplicates.
- If the ***Auto generate IDs for elements*** *option* is not selected and you duplicate elements with existing IDs, the ID values are removed from the duplicates. However, when elements are duplicated in the same document, this option has no effect and IDs are preserved if the ***Remove IDs when copying content in the same document*** *option* is not selected.
- If the ***Remove IDs when copying content in the same document*** *option* is selected, the ID values are removed from elements that are duplicated in the same document. However, selecting this option has no effect if the **Auto generate IDs for elements** option is selected.
- If the ***Remove IDs when copying content in the same document*** *option* is not selected, the ID values are preserved when elements are duplicated in the same document. For all *frameworks* other than DocBook, this option has no affect on elements that are duplicated in other documents. For DocBook, it works the same regardless of whether its the same or another document.

Controlling the Default ID Generation Options

It is possible to configure the default ID generation options for DITA, DocBook, and TEI document types. In the `frameworks` folder for each of those document types, there is an XML configuration file called `idGenerationDefaultOptions.xml` that contains the default settings for generating IDs in each particular type of document. To configure the default settings, you can edit this file and save it back to the same directory.

The configuration file can be found in the `resources` folder within the particular *framework*. For example, the configuration file for the DITA *framework* is located in: *`[OXYGEN_INSTALL_DIR]`*`/frameworks/dita/resources/idGenerationDefaultOptions.xml`.

Sharing Default ID Generation Options

If you want to share your configured default ID generation settings with other member of your team, follow these steps:

1. Configure the `idGenerationDefaultOptions.xml` file for your *framework* according to your needs.
2. Bundle a modified version of the entire *framework* folder (for example, *`[OXYGEN_INSTALL_DIR]`*`/frameworks/dita/`). To do this:
   a. *Open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Document Type Association**.
   b. Select your document type and click the **Extend** button.
   c. In the ***Document type*** *configuration dialog box* that is now displayed, select **External** for the **Storage** option. By default, this will save the extension in a new folder in the `frameworks` folder (for example, *`[OXYGEN_INSTALL_DIR]`*`/frameworks/dita-extension (1)`), but you can also use the **Browse** button to specify a specific name and folder.
   d. In this new extension folder, create a new folder called `resources` and add your modified `idGenerationDefaultOptions.xml` file to this new `resources` folder.

e. Go back to the **Document Type Association** preferences page, select the extended *framework*, and click **Edit**.
f. Go to the ***Classpath** tab*, add a reference to your new `resources` folder, and move this reference up (using the **Move Up** button) so that it is the first one that appears in the list.
g. Click **OK** and exit out of the preferences page.

3. Distribute your newly extended folder to other team members by using one of the methods described in *Sharing a Framework (Document Type)* on page 1239.

Using Form Controls in Author Mode

You can use form controls in **Author** mode in a variety of ways to make it easier to capture, organize, and edit content. Oxygen XML Editor includes *built-in form controls* that can be used by content authors in **Author** mode. The types of built-in form controls that you can use include the following:

- *Audio* - A media object that plays audio clips.
- *Browser* - A media object that renders HTML frames or interact with SVG documents
- *Button* - A graphical user interface object that performs a specific action.
- *Button Group* - A graphical user interface group of buttons (such as radio buttons) that perform specific actions.
- *Checkbox* - A graphical user interface box that you can click to select or deselect a value.
- *Combo Box* - A graphical user interface object that can be a drop-down menu or a combination of a drop-down menu and a single-line text field.
- *Date Picker* - A form control object that allows you to select a date in a specified format.
- *HTML Content* - A graphical user interface box that is used for rendering HTML content.
- *Pop-up* - A contextual menu that provides quick access to various actions.
- *Text Area* - A box that allows you to enter multiple lines of text.
- *Text Field* - A graphical user interface box that allows you to enter a single line of text.
- *URL Chooser* - A dialog box that allows you to select the location of local or remote resources.
- *Video* - A media object that plays videos.

You can also *implement custom form controls* for more specific needs.

The following image is an example of how form controls can be used by content authors in **Author** mode. It includes several button form controls, a combo box, and a text field. The icon is a button form control that is assigned a specific action that changes the layout to an editing mode. The **[+]** and **[-]** icons are also button form controls that are assigned specific actions to add or delete records from the document. The *Direct manager* row includes a combo box form control that is both a drop-down menu and an editable text field, while the *Homepage* row includes a simple editable text field form control.

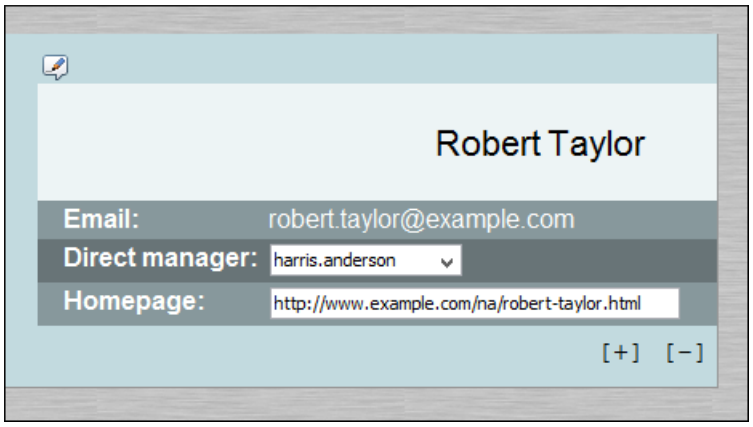


Figure 221: Example of Form Controls in Author Mode

You can use your imagination to envision the multitude of ways that you can use form controls to make the editing experience for content authors easier and more efficient. As a working example, a bundled *samples* project (located in the `samples` folder inside the Oxygen XML Editor installation directory) contains a file called `personal.xml` that contains form controls. You can use this file, along with its corresponding `personal.css`

file (form controls are defined in the CSS) to experiment with an example of how form controls can be implemented in **Author** mode.

Contextual Menu Actions in Author Mode

Oxygen XML Editor includes powerful support for editing XML documents through actions included in the contextual menu. When editing XML documents in **Author** mode, the contextual menu includes *general* actions that are available for all of the recognized document types and *framework*-specific actions that are configured for each document type.

General Contextual Menu Actions in Author Mode

The *general* actions that are available in the contextual menu (some of them are also available in the submenus of the **Document** menu) for all document types include the following:

Quick Fix (Alt + 1 (Command + Alt + 1 on OS X))

Available when the contextual menu is invoked on an error where *Oxygen XML Editor can provide a Quick Fix*.

Open Image

Available when the contextual menu is invoked on an image. This action allows you to open an image in the Oxygen XML Editor *Image Viewer* or in a default system application associated with the current image type.

Track Changes Actions

Available when the *Track Changes feature* is enabled and the contextual menu is invoked on a change. The following options are available:

Accept Change(s)

Accepts the *Tracked Change* located at the cursor position or all of the changes in a selection. If you select a part of a *deletion* or *insertion* change, only the selected content is accepted. If you select multiple changes, all of them are accepted. For an *insertion* change, it keeps the inserted text and for a *deletion* change, it removes the content from the document.

Reject Change(s)

Rejects the *Tracked Change* located at the cursor position or all of the changes in a selection. If you select a part of a *deletion* or *insertion* change, only the selected content is rejected. If you select multiple changes, all of them are rejected. For an *insertion* change, it removes the inserted text and for a *deletion* change, it preserves the original content.

Comment Change

Opens a dialog box that allows you to add a comment to an existing *Tracked Change*. The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

Author Callout Actions

Available when the contextual menu is invoked on a *callout*. If the corresponding options in the ***Show review callouts** section* are selected in the ***Callouts** preferences page*, the callouts are displayed in **Author** mode for comments, tracked insertion changes, or tracked deletion changes.

Insertion or Deletion Callout Actions

The following actions are available in the contextual menu when invoked on an *insertion* or *deletion* callout box:

Reply

Opens a dialog box that allows you to add a reply to a comment or *Tracked Changes*. When replying to a comment, the dialog box shows the entire conversation in the comment thread, starting with the first comment added in the particular thread, followed by all the replies. After replies are added to a comment thread, they are displayed with an indentation in the callouts and ***Review** view*.

Mark as Done

A toggle action that marks or unmarks a comment or comment thread as being done. It is also available for *Tracked Changes* that are displayed in a callout. When a comment or change is marked as done, the callout is grayed out and cannot be edited unless the action is toggled to the unmarked state.

The action applies to the particular comment and all of its descendents. This is useful for marking comments or changes that have been addressed, leaving only those that still need some attention.

Accept Change(s)

Accepts the *Tracked Change* located at the cursor position or all of the changes in a selection. If you select a part of a *deletion* or *insertion* change, only the selected content is accepted. If you select multiple changes, all of them are accepted. For an *insertion* change, it keeps the inserted text and for a *deletion* change, it removes the content from the document.

Reject Change(s)

Rejects the *Tracked Change* located at the cursor position or all of the changes in a selection. If you select a part of a *deletion* or *insertion* change, only the selected content is rejected. If you select multiple changes, all of them are rejected. For an *insertion* change, it removes the inserted text and for a *deletion* change, it preserves the original content.

Comment Change

Opens a dialog box that allows you to add a comment to an existing *Tracked Change*. The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

Edit Reference

If the fragment that contains a callout is a reference, use this option to go to the reference and edit the callout.

Callouts Options

Select this option to open the ***Callouts*** *preference page* where you can configure various callout options.

Comment Callout Actions

The following actions are available in the contextual menu when invoked on a *comment* callout box:

Reply

Opens a dialog box that allows you to add a reply to a comment or *Tracked Changes*. When replying to a comment, the dialog box shows the entire conversation in the comment thread, starting with the first comment added in the particular thread, followed by all the replies. After replies are added to a comment thread, they are displayed with an indentation in the callouts and ***Review*** *view*.

Mark as Done

A toggle action that marks or unmarks a comment or comment thread as being done. It is also available for *Tracked Changes* that are displayed in a callout. When a comment or change is marked as done, the callout is grayed out and cannot be edited unless the action is toggled to the unmarked state. The action applies to the particular comment and all of its descendents. This is useful for marking comments or changes that have been addressed, leaving only those that still need some attention.

Show/Edit Comment

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Callouts Options

Select this option to open the ***Callouts*** *preference page* where you can configure various callout options.

Edit Attributes

Displays an *in-place attributes editor* that allows you to manage the attributes of an element.

Edit Profiling Attributes

Allows you to change the *profiling attributes* defined on all selected elements.

Insert submenu

This submenu includes insert actions that are specific to each *framework*, along with the following general action:

Insert Entity

Allows you to insert a predefined entity or character entity. Surrogate character entities (range #x10000 to #x10FFFF) are also accepted. Character entities can be entered in one of the following forms:

- *#<decimal value>* - e. g. #65
- *&#<decimal value>*; - e. g. &#65
- *#x<hexadecimal value>* - e. g. #x41
- *&#x<hexadecimal value>*; - e. g. &#x41

Cut (Ctrl + X (Command + X on OS X))

Removes the current selected content from the document and places it in the clipboard.

Copy (Ctrl + C (Command + C on OS X))

Places a copy of the current selected content in the clipboard.

Paste (Ctrl + V (Command + V on OS X))

Inserts the current clipboard content into the document at the cursor position.

Paste special submenu

This submenu includes special paste actions that are specific to each *framework*, as well as the following general paste actions:

Paste As XML

Pastes clipboard content that is considered to be XML, preserving its XML structure.

Paste As Text

Pastes clipboard content, ignoring any structure or styling markup.

Select submenu

This submenu allows you to select the following:

Element

Selects the entire element at the current cursor position.

Content

Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

Parent

Selects the entire parent element at the current cursor position.

Text submenu

This submenu contains the following actions:

To Lower Case

Converts the selected content to lower case characters.

To Upper Case

Converts the selected content to upper case characters.

Capitalize Sentences

Converts to upper case the first character of every selected sentence.

Capitalize Words

Converts to upper case the first character of every selected word. 0034

Count Words

Counts the number of words and characters (no spaces) in the entire document or in the selection for regular content and read-only content.

Note: The content marked as deleted with *change tracking* is ignored when counting words.

Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on OS X))

Converts a sequence of hexadecimal characters to the corresponding Unicode character. The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed of 2 to 4 hexadecimal characters and may or may not be preceded by the `0x` or `0X` prefix. Examples of valid sequences: `0x0045`, `0X0125`, `1253`, `265`, `43`.

Refactoring submenu

Contains a series of actions designed to alter the XML structure of the document:

Toggle Comment

Encloses the currently selected text in an XML comment, or removes the comment if it is commented.

Move Up (Alt + UpArrow)

Moves the current node or selected nodes in front of the previous node.

Move Down (Alt + DownArrow)

Moves the current node or selected nodes after the subsequent node.

Split Element (Alt + Shift + D (Ctrl + Alt + D on OS X))

Splits the content of the closest element that contains the position of the cursor. Thus, if the cursor is positioned at the beginning or at the end of the element, the newly created sibling will be empty.

Join Elements

Joins two adjacent *block elements* that have the same name. The action is available only when the cursor position is between the two adjacent *block elements*. Also, joining two *block elements* can be done by pressing the **Delete** or **Backspace** keys and the cursor is positioned between the boundaries of these two elements.

Surround with Tags (Ctrl + E (Command + E on OS X))

Allows you to choose a tag to enclose a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.

- If the ***Position cursor between tags*** *option* is selected in the **Content Completion** preferences page, the cursor is placed between the start and end tag.
- If the ***Position cursor between tags*** *option* is not selected in the **Content Completion** preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

Surround with '[tag]' (Ctrl + ForwardSlash (Command + ForwardSlash on OS X))

Surround the selected content with the last tag used.

Rename Element

The element from the cursor position, and any elements with the same name, can be renamed according with the options from the **Rename** dialog box.

Delete Element Tags

Deletes the tags of the closest element that contains the position of the cursor. This operation is also executed if the start or end tags of an element are deleted by pressing the **Delete** or **Backspace** keys.

Remove All Markup

Removes all the XML markup inside the selected block of content and keeps only the text content.

Remove Text

Removes the text content of the selected block of content and keeps the markup in tact with empty elements.

Attributes submenu

Contains predefined XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

Add/Change attribute

Allows you to change the value of an attribute or insert a new one.

Delete attribute

Allows you to remove one or more attributes.

Rename attribute

Allows you to rename an attribute.

Replace in attribute value

Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

Comments submenu

Contains predefined XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

Delete comments

Allows you to delete comments found inside one or more elements.

DITA submenu

Contains predefined XML refactoring operations that pertain to DITA documents with some of the information preconfigured based upon the current context.

Change topic ID to file name

Use this operation to change the ID of a topic to be the same as its file name.

Convert conrefs to conkeyrefs

Use this operation to convert `conref` attributes to `conkeyref` attributes.

Convert simple tables to CALS tables

Use this operation to convert DITA simple tables to CALS tables.

Convert to Concept

Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept).

Convert to Reference

Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference).

Convert to Task

Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task).

Convert to Topic

Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic).

Convert to Troubleshooting

Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting).

Elements submenu

Contains predefined XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

Delete element

Allows you to delete elements.

Delete element content

Allows you to delete the content of elements.

Insert element

Allows you to insert new elements.

Rename element

Allows you to rename elements.

Unwrap element

Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

Wrap element

Allows you to surround elements with element tags.

Wrap element content

Allows you to surround the content of elements with element tags.

Fragments submenu

Contains predefined XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

Insert XML fragment

Allows you to insert an XML fragment.

Replace element content with XML fragment

Allows you to replace the content of elements with an XML fragment.

Replace element with XML fragment

Allows you to replace elements with an XML fragment.

JATSKit submenu

Contains predefined XML refactoring operations that pertain to JATS documents with some of the information preconfigured based upon the current context.

Add BITS DOCTYPE - NLM/NCBI Book Interchange 2.0

Adds an NLM 'BITS' 2.0 DOCTYPE declaration.

Add Blue DOCTYPE - NISO JATS Publishing 1.1

Adds a JATS 'Blue' 1.1 DOCTYPE declaration.

Normalize IDs

Assigned IDs are normalized and IDs are assigned to some elements that are missing them.

Review submenu

This submenu includes the following actions:

Track Changes

Enables or disables the *Track Changes* support for the current document.

Accept Change(s) and Move to Next

Accepts the *Tracked Change* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a *deletion* or *insertion* change, only the selected content is accepted.

Accept All Changes

Accepts all *Tracked Changes* in the current document.

Reject Change(s) and Move to Next

Rejects the *Tracked Change* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a *deletion* or *insertion* change, only the selected content is rejected.

Reject All Changes

Rejects all *Tracked Changes* in the current document.

Comment Change

Opens a dialog box that allows you to add a comment to an existing *Tracked Change*. The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

Highlight

Enables the highlighting tool that allows you to mark text in your document.

Colors

Allows you to select the color for highlighting text.

Stop highlighting

Use this action to deactivate the highlighting tool.

Remove highlight(s)

Use this action to remove highlighting from the document.

Add Comment

Inserts a comment at the cursor position. The comment appears in a callout box and a tooltip (when hovering over the change).

Show/Edit Comment

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Manage Reviews

Opens the ***Review*** *view*.

Manage IDs submenu

This submenu is available for XML documents that have an associated DTD, XML Schema, or Relax NG schema. It includes the following actions:

Rename in

Renames the ID and all its occurrences. Selecting this action opens the **Rename XML ID** dialog box. This dialog box lets you insert the new ID value and choose the scope of the rename operation.

Search References

Searches for the references of the ID. By default, the scope of this action is the current project. If you configure a scope using the ***Select the scope for the Search and Refactor operations*** dialog box, this scope will be used instead.

Search References in

Searches for the references of the ID. Selecting this action opens the ***Select the scope for the Search and Refactor operations***.

Search Occurrences in file

Searches for the occurrences of the ID in the current document.

Folding submenu

This submenu includes the following actions:

Toggle Fold

Toggles the state of the current fold.

Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on OS X))

Folds all the elements except the current element.

Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on OS X))

Folds the elements indented with one level inside the current element.

Expand Child Folds

Unfolds all child elements of the currently selected element.

Expand All (Ctrl + NumPad* (Command + NumPad* on OS X))

Unfolds all elements in the current document.

Inspect Styles

Opens the ***CSS Inspector*** *view* that allows you to examine the CSS rules that match the currently selected element.

Options

Opens the *Author mode options page*.

Document Type-Specific Contextual Menu Actions in Author Mode

Other *document type-specific* actions are available in the contextual menu of **Author** mode for the following document types (click the links to see the default actions that are available for each specific document types):

- ***DocBook4*** *Author Actions*
- ***DocBook5*** *Author Actions*
- ***DITA*** *Author Actions*
- ***DITA Map*** *Author Actions*
- ***XHTML*** *Author Actions*
- ***TEI ODD*** *Author Actions*
- ***TEI P5*** *Author Actions*
- ***JATS*** *Author Actions*

Validating XML Documents

The W3C XML specification states that a program should not continue to process an XML document if it finds a validation error. The reason is that XML software should be easy to write and all XML documents should be compatible. With HTML, for example, it is possible to create documents with lots of errors (for instance, when you forget an end tag). One of the main reasons that various HTML browsers have performance and compatibility problems is that they have different methods of figuring out how to render a document when an HTML error is encountered. Using XML helps to eliminate such problems.

Even when creating XML documents, errors are easily introduced. When working with large projects or a large number of files, the probability that errors will occur is even greater. Preventing and solving errors in your projects can be time consuming and frustrating. Fortunately, Oxygen XML Editor provides validation functions that allow you to easily identify errors and their location.

Related Information:

Working with Modular XML Files in the Master Files Context

Checking XML Well-Formedness

A *Well-formed XML* document is a document that conforms to the XML syntax rules. A *Namespace Well-Formed XML* document is a document that is *Well-formed XML* and is also *Namespace-wellformed* and *Namespace-valid*.

Well-Formedness Rules

The XML Syntax rules for *Well-formed XML* are as follows:

- All XML elements must have a closing tag.
- XML tags are case-sensitive.
- All XML elements must be properly nested.
- All XML documents must have a root element.
- Attribute values must always be quoted.

- With XML, whitespace is preserved.

The *Namespace-wellformed* rules are as follows:

- All element and attribute names contain either zero or one colon.
- No entity names, processing instruction targets, or notation names contain any colons.

The *Namespace-valid* rules are as follows:

- The *xml* prefix is by definition bound to the namespace name: *http://www.w3.org/XML/1998/namespace*. It MAY be declared, but MUST NOT be undeclared or bound to any other namespace name. Other prefixes MUST NOT be bound to this namespace name.
- The *xmlns* prefix is used only to declare namespace bindings and is by definition bound to the namespace name: *http://www.w3.org/2000/xmlns/*. It MUST NOT be declared or undeclared. Other prefixes MUST NOT be bound to this namespace name.
- All other prefixes beginning with the three-letter sequence *x*, *m*, *l*, in any case combination, are reserved. This means that users SHOULD NOT use them except as defined by later specifications and processors MUST NOT treat them as fatal errors.
- The namespace prefix (unless it is *xml* or *xmlns*) MUST have been declared in a namespace declaration attribute in either the start tag of the element where the prefix is used or in an ancestor element (for example, an element in whose content the prefixed markup occurs). Furthermore, the attribute value in the innermost such declaration MUST NOT be an empty string.

Check for Well-Formedness

To check if a document is *Namespace Well-Formed XML* and *Namespace-valid*, select the **Check Well-Formedness (Ctrl + Shift + W (Command + Shift + W on OS X))** action from the **Document** > **Validate** menu or from the **Validation** drop-down menu on the toolbar.

The selected files in the current project can also be checked for well-formedness with a single action by selecting the **Check Well-Formedness** action from the **Validate** submenu when invoking the contextual menu in the ***Project*** *view*.

If any errors are found, the result is displayed in the message panel at the bottom of the editor. Each error is displayed as one record in the result list and is accompanied by an error message. Clicking the record will open the document containing the error and highlight its approximate location.

Example: A non *Well-formed XML* Document

```
<root><tag></root>
```

When **Check Well-Formedness** is performed the following error is raised:

```
The element type "tag" must be terminated by the matching end-tag "</tag>"
```

To resolve the error, click the record in the result list and it will locate and highlight the approximate position of the error. In this case, identify the tag that is missing an end tag and insert </tag>.

Example: A non *Namespace-wellformed* Document

```
<x::y></x::y>
```

When **Check document form** is performed the following error is raised:

```
Element does not match QName production: QName::=(NCName':')?NCName.
```

Example: A non *Namespace-valid* Document

```
<x:y></x:y>
```

When **Check document form** is performed the following error is raised:

```
The prefix "x" for element "x:y" is not bound.
```

Validating XML Documents Against a Schema

A *Valid* XML document is a *Well-Formed* XML document that also conforms to the rules of a schema that defines the legal elements of an XML document. The schema type can be: XML Schema, Relax NG (full or compact syntax), Schematron, Document Type Definition (DTD), or Namespace-based Validation Dispatching Language (NVDL).

The purpose of the schema is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements.

The **Validate** function ensures that your document is compliant with the rules defined by an associated DTD, XML Schema, Relax NG, or Schematron schema. XML Schema or Relax NG schema also allows you to embed Schematron rules. For Schematron validations you can also select the **validation phase**.

Related Information:

Presenting Schematron Validation Issues

Associating a Schema to XML Documents on page 464

Automatic Validation

Oxygen XML Editor can be configured to automatically mark validation errors in the document as you are editing. The ***Enable automatic validation*** *option* in the ***Document Checking*** *preferences page* controls whether or not all validation errors and warnings will automatically be highlighted in the editor panel.

The automatic validation starts parsing the document and marking the errors after a *configurable delay* from the last key typed. Errors are highlighted with underline markers in the main editor panel and small rectangles on the right side ruler of the editor panel. Hovering over a validation error presents a tooltip message with more details about the error.

If the error message is long and it is not displayed completely in the error line at the bottom of the editing area, double-clicking the error icon at the left of the error line or on the error line displays an information dialog box with the full error message. You can use the arrow buttons in this dialog box to navigate through the errors issued by the Automatic Validation feature.

Related Information:

Manual Validation Actions on page 446

Presenting Validation Errors in Text Mode on page 206

Presenting Validation Errors in Author Mode on page 236

Manual Validation Actions

You can choose to validate documents at any time by using the manual validation actions that are available in Oxygen XML Editor.

Manually Validate Current Document

To manually validate the currently edited document, use one of the following actions:

Validate (Ctrl + Shift + V (Command + Shift + V on OS X))

Available from the **Validation** drop-down menu on the toolbar, the **Document** > **Validate** menu, or from the **Validate** submenu when invoking the contextual menu in the ***Project*** *view*.

An error list is presented in the message panel at the bottom of the editor. Markup of the current document is checked to conform with the specified DTD, XML Schema, or Relax NG schema rules. This action also re-parses the *XML Catalogs* and resets the schema used for content completion.

Validate (cached)

Available from the **Validation** drop-down menu on the toolbar or the **Document** > **Validate** menu.

This action caches the schema, allowing it to be reused for the next validation. Markup of the current document is checked to conform with the specified DTD, XML Schema, or Relax NG schema rules.

Note: Automatic validation also caches the associated schema.

Validate with

Available from the **Validation** drop-down menu on the toolbar, (or **Document** > **Validate** menu).

This action opens a dialog box that allows you to *specify a schema for validating the current document*.

You can use this action to validate the current document using a schema of your choice (XML Schema, DTD, Relax NG, NVDL, Schematron schema), other than the associated one. An error list is presented in the message panel at the bottom of the editor. Markup of current document is checked to conform with the specified schema rules.

Note: The **Validate with** action does not work for files loaded through an *Oxygen XML Editor custom protocol plugin* developed independently and added to Oxygen XML Editor after installation.

Validate with Schema

Available from the **Validate** submenu when invoking contextual menu in the ***Project*** *view*.

This action opens a dialog box that allows you to *specify a schema for validating all selected files*.

Other Validation Options

To quickly open the schema used for validating the current document, select the **Open Associated Schema** action from the toolbar (or **Document** > **Schema** menu).

The **Validation options** button, available in the **Document** > **Validate** menu, allows you to quickly access to the *validation options* for the built-in validator in the Oxygen XML Editor preferences page.

Tip: If a large number of validation errors are detected and the validation process takes too long, you can *limit the maximum number of reported errors in the* ***Document Checking*** *preferences page*.

Related Information:

Automatic Validation on page 446
Presenting Validation Errors in Text Mode on page 206
Presenting Validation Errors in Author Mode on page 236

Presenting Validation Errors in Text Mode

Oxygen XML Editor can be configured to *automatically validate documents* while editing in the **Text** mode, and actions are also available to *manually validate documents* on-request.

In **Text** mode, validation issues are marked in the following locations:

- In the main editing pane, with the issue underlined in a color according to the type of issue.
- In the right-side vertical stripe, with a marker that is colored according to the type of issue.
- For attributes with detected issues, in the ***Attributes*** *view*, with the attribute and its value colored according to the type of issue.

The colors for each type of issue are as follows:

- **Validation Errors** [Red] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the **Attributes** view are colored in red.
- **Validation Warnings** [Yellow] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the **Attributes** view are colored in yellow.
- **Validation Info** [Blue] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the **Attributes** view are colored in blue.

You can configure the color for each type in the ***Document Checking*** *preferences page*.

Hovering over a validation issue presents a tooltip message with more details about the problem and *possible quick fixes* (if available for that issue).

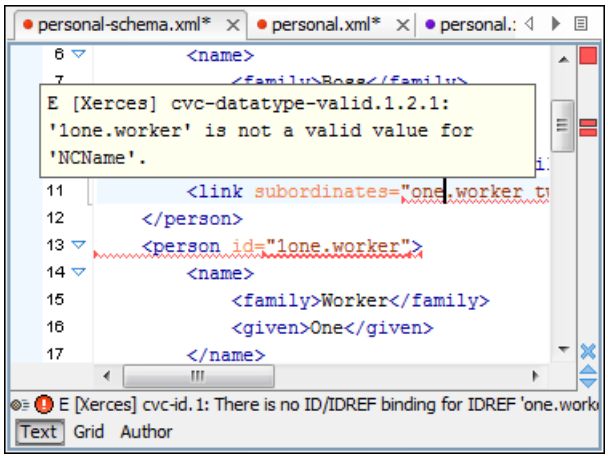


Figure 222: Presenting Validation Errors in Text Mode

Also, the stripe on the right side of the editor panel is designed to display the issues found during the validation process and to help you locate them in the document. The stripe contains the following:

Upper Part of the Stripe

A success indicator square will turn green if the validation is successful or only info messages are found, red if validation errors are found, or yellow if only validation warnings are found. More details about the issues are displayed in a tool tip when you hover over indicator square. If there are numerous problems, only the first three are presented in the tool tip.

Middle Part of the Stripe

Errors are depicted with red markers, warnings with yellow markers, and info message with blue markers. If you want to limit the number of markers that are displayed, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Editor** > **Document checking**, and specify the desired limit in the ***Maximum number of validation highlights** option*.

Clicking a marker will highlight the corresponding text area in the editor. The validation message is also displayed both in a tool tip (when hovering over the marker) and in the message area on the bottom of the editor panel (clicking the **Document checking options** button opens the ***Document Checking** preferences page*.

Bottom Part of the Stripe

Two navigation arrows ( ) allow you to jump to the next or previous issue. The same actions can be triggered from **Document** > **Automatic validation** > **Next error** (**Ctrl + Period (Command + Period on OS X)**) and **Document** > **Automatic validation** > **Previous error** (**Ctrl + Comma (Command + Comma on OS X)**). Also, the button can be used to clear all the validation markers.

Status messages from every validation action are also logged in the ***Information** view*.

If you want to see all the validation messages grouped in the ***Results** panel*, you should use the **Validate** action from the toolbar or **Document** > **Validate** menu..

Related Information:

Validating XML Documents Against a Schema on page 446

Presenting Schematron Validation Issues

Presenting Validation Errors in Author Mode

Oxygen XML Editor can be configured to *automatically validate documents* while editing in the **Author** mode, and actions are also available to *manually validate documents* on-request.

In **Author** mode, validation issues are marked in the following locations:

- In the main editing pane, with the issue underlined in a color according to the type of issue.
- In the right-side vertical stripe, with a marker that is colored according to the type of issue.

- For attributes with detected issues, in the ***Attributes** view*, with the attribute and its value colored according to the type of issue.

The colors for each type of issue are as follows:

- **Validation Errors** [Red] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the **Attributes** view are colored in red.
- **Validation Warnings** [Yellow] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the **Attributes** view are colored in yellow.
- **Validation Info** [Blue] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the **Attributes** view are colored in blue.

You can configure the color for each type in the ***Document Checking** preferences page*.

Hovering over a validation issue presents a tooltip message with more details about the problem and *possible quick fixes* (if available for that issue).

Information about the issue is also displayed in the message area on the bottom of the editor panel (clicking the **Document checking options** button opens the ***Document Checking** preferences page* where you can configure some validation options (such as the colors used to present the validation issues). Some validation messages include an icon ( ) that provides a link to a style guide or specification.

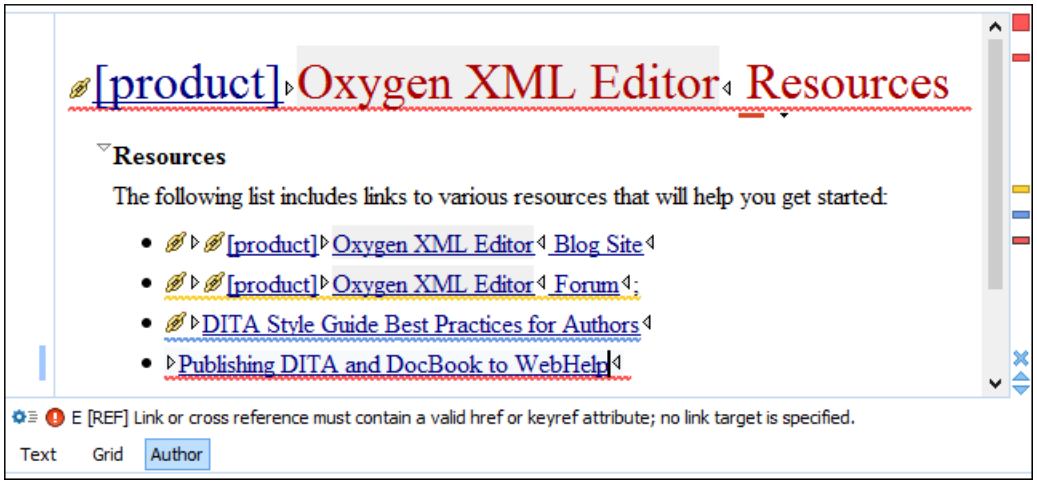


Figure 223: Presenting Validation Errors in Author Mode

Also, the stripe on the right side of the editor panel is designed to display the issues found during the validation process and to help you locate them in the document. The stripe contains the following:

Upper Part of the Stripe

A success indicator square will turn green if the validation is successful or only info messages are found, red if validation errors are found, or yellow if only validation warnings are found. More details about the issues are displayed in a tool tip when you hover over indicator square. If there are numerous problems, only the first three are presented in the tool tip.

Middle Part of the Stripe

Errors are depicted with red markers, warnings with yellow markers, and info message with blue markers. If you want to limit the number of markers that are displayed, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Editor** > **Document checking**, and specify the desired limit in the ***Maximum number of validation highlights** option*.

Clicking a marker will highlight the corresponding text area in the editor. The validation message is also displayed both in a tool tip (when hovering over the marker) and in the message area on the bottom of the editor panel (clicking the **Document checking options** button opens the ***Document Checking** preferences page*.

Bottom Part of the Stripe

Two navigation arrows ( ) allow you to jump to the next or previous issue. The same actions can be triggered from **Document** > **Automatic validation** > **Next error** (**Ctrl + Period (Command + Period on OS X)**) and **Document** > **Automatic validation** > **Previous error** (**Ctrl + Comma (Command + Comma on OS X)**). Also, the ✖ button can be used to clear all the validation markers.

Status messages from every validation action are also logged in the ***Information*** *view*.

If you want to see all the validation messages grouped in the ***Results*** *panel*, you should use the **Validate** action from the toolbar or **Document** > **Validate** menu..

Related Information:

Validating XML Documents Against a Schema on page 446
Presenting Schematron Validation Issues

Customizing Assert Error Messages

To customize the error messages that the Xerces or Saxon validation engines display for the `assert` and `assertion` elements, set the `message` attribute on these elements.

- For Xerces, the `message` attribute has to belong to the `http://xerces.apache.org` namespace.
- For Saxon, the `message` attribute has to belong to the `http://saxon.sourceforge.net/` namespace.

The value of the `message` attribute is the error message displayed if the assertion fails.

Custom Validators

If you need to validate the edited document with a validation engine that is different from the built-in engine, you can configure external validators in the *Custom Validation Engines preferences page*. After a custom validation engine is *properly configured*, it can be applied on the current document by selecting it from the list of custom validation engines in the **Validation** toolbar drop-down menu. The document is validated against the schema declared in the document.

Some validators are configured by default but there are third-party processors that do not support the *output message format* of Oxygen XML Editor for linked messages:

- **LIBXML** - Included in Oxygen XML Editor (Windows edition only). It is associated to XML Editor. It is able to validate the edited document against XML Schema, Relax NG schema full syntax, internal DTD (included in the XML document) or a custom schema type. Support for *XML Catalogs* (the --catalogs parameter) and XInclude processing (--xinclude) are enabled by default in the preconfigured LIBXML validator. The --postvalid parameter is also set by default and it allows LIBXML to validate correctly the main document even if the XInclude fragments contain IDREFS to ID's located in other fragments.

  For validation against an external DTD specified by URI in the XML document, add the --dtdvalid ${ds} parameter manually to the DTD validation command line. `${ds}` represents the detected DTD declaration in the XML document.

  CAUTION: File paths containing spaces are not handled correctly in the LIBXML processor. For example, the built-in *XML Catalog* files of the predefined document types (DocBook, TEI, DITA, etc.) are not handled by LIBXML if Oxygen XML Editor is installed in the default location on Windows (`C:\Program Files`) because the built-in *XML catalog* files are stored in the `frameworks` subfolder of the installation folder and in this case, the file path contains at least one space character.

  Attention: On OS X, if the full path to the LIBXML executable file is not specified in the **Executable path** text field, some errors may occur during validation against a W3C XML Schema, such as:

  ```
  Unimplemented block at ... xmlschema.c
  ```

  To avoid these errors, specify the full path to the LIBXML executable file.
- **Saxon-EE** - Included in Oxygen XML Editor. It is associated to XML Editor and XSD Editor. It is able to validate XML Schema schemas and XML documents against XML Schema schemas. The validation is done according to the W3C XML Schema 1.0 or 1.1. This can be *configured in Preferences*.

- **MSXML 4.0** - Included in Oxygen XML Editor (Windows edition only). It is associated to XML Editor, XSD Editor and XSL Editor. It is able to validate the edited document against XML Schema, internal DTD (included in the XML document), external DTD or a custom schema type.
- **MSXML.NET** - Included in Oxygen XML Editor (Windows edition only). It is associated to XML Editor, XSD Editor and XSL Editor. It is able to validate the edited document against XML Schema, internal DTD (included in the XML document), external DTD or a custom schema type.
- **XSV** - Not included in Oxygen XML Editor. Windows and Linux distributions of XSV can be downloaded from *http://www.cogsci.ed.ac.uk/~ht/xsv-status.html*. The executable path is *already configured in Oxygen XML Editor* for the `[OXYGEN_INSTALL_DIR]/xsv` installation folder. If it is installed in a different folder, the predefined executable path must be *corrected in Preferences.* It is associated to XML Editor and XSD Editor. It is able to validate the edited document against XML Schema or a custom schema type.
- **SQC (Schema Quality Checker from IBM)** - Not included in Oxygen XML Editor. It can be downloaded *from here* (it comes as a .zip file, at the time of this writing SQC2.2.1.zip is about 3 megabytes).
  The executable path and working directory are already configured for the SQC installation directory `[OXYGEN_INSTALL_DIR]/sqc`. If it is installed in a different folder, the predefined executable path and working directory must be *corrected in the Preferences page.* It is associated to XSD Editor.

A custom validator cannot be applied on files loaded through an *Oxygen XML Editor custom protocol plugin* developed independently and added to Oxygen XML Editor after installation.

Linked Output Messages of an External Engine

Validation engines display messages in an output view at the bottom of the Oxygen XML Editor window. If such an output message (**warning**, **error**, **fatal error**, etc) spans between three to six lines of text and has the format specified below, then the message is linked to a location in the validated document. Clicking the message in the output view highlights the location of the message in an editor panel containing the file referenced in the message. This behavior is similar to the linked messages generated by the default built-in validator.

Linked messages have the following format:

- *Type*:[F|E|W] (the string *Type:* followed by a letter for the type of the message: fatal error, error, warning) - this property is optional in a linked message.
- *SystemID*: a system ID of a file (the string `SystemID:` followed by the system ID of the file that will be opened for highlighting when the message is clicked in the output message - usually the validated file, the schema file or an included file).
- *Line*: a line number (the string *Line:* followed by the number of the line that will be highlighted).
- *Column*: a column number (the string *Column:* followed by the number of the column where the highlight will start on the highlighted line) - this property is optional in a linked message.
- *EndLine*: a line number (the string *EndLine*: followed by the number of the line where the highlight ends) - this property is optional in a linked message.
- *EndColumn*: a column number (the string *EndColumn*: followed by the number of the column where the highlight ends on the end line) - this property is optional in a linked message.

  Note: The *Line*/*Column* pair works in conjunction with the *EndLine*/*EndColumn* pair. Thus, if both pairs are specified, then the highlight starts at *Line*/*Column* and ends at *EndLine*/*EndColumn*. If the *EndLine*/*EndColumn* pair is missing, the highlight starts from the beginning of the line identified by the *Line* parameter and ends at the column identified by the *Column* parameter.
- *AdditionalInfoURL*: the URL string pointing to a remote location where additional information about the error can be found - this line is optional in a linked message.
- *Description*: message content (the string *Description:* followed by the content of the message that will be displayed in the output view).

Example:

Example of how a custom validation engine can report an error using the format specified above:

```
Type: E
SystemID: file:///c:/path/to/validatedFile.xml
Line: 10
Column: 20
EndLine: 10
EndColumn: 35
AdditionalInfoURL: http://www.host.com/path/to/errors.html#errorID
Description: custom validator message
```

Validation Scenarios

A complex XML document is split in smaller interrelated modules. These modules do not make much sense individually and cannot be validated in isolation due to interdependencies with other modules. Oxygen XML Editor validates the main module of the document when an imported module is checked for errors.

A typical example is the chunking of a DocBook XSL stylesheet that has `chunk.xsl` as the main module and `param.xsl`, `chunk-common.xsl`, and `chunk-code.xsl` as imported modules. `param.xsl` only defines XSLT parameters. The module `chunk-common.xsl` defines an XSLT template with the name `chunk`. `Chunk-code.xsl` calls this template. The parameters defined in `param.xsl` are used in the other modules without being redefined.

Validating `chunk-code.xsl` as an individual XSLT stylesheet generates misleading errors in regards to parameters and templates that are used but undefined. These errors are only caused by ignoring the context in which this module is used in real XSLT transformations and in which it is validated. To validate such a module, define a validation scenario to set the main module of the stylesheet and the validation engine used to find the errors. Usually this engine applies the transformation during the validation process to detect the errors that the transformation generates.

You can validate a stylesheet with several engines to make sure that you can use it in various environments and have the same results. For example, an XSLT stylesheet may be applied with Saxon 6.5, Xalan, and MSXML 4.0 engines in different production systems.

Other examples of documents that can benefit from a validation scenario include:

- A complex XQuery file with a main module that imports modules developed independently but validated in the context of the main module of the query. In an XQuery validation scenario, the default validator of Oxygen XML Editor (Saxon 9) or any connection to a database that supports validation (Berkeley DB XML Database, eXist XML Database, Documentum xDB (X-Hive/DB) 10 XML Database, MarkLogic version 5 or newer) can be set as a validation engine.
- An XML document in which the *master file* includes smaller fragment files using XML entity references.

  Note: If a *master file* is associated with the current file, the validation scenarios defined in the *master file*, along with any Schematron schema defined in the default scenarios for that particular *framework*, are used for the validation. These take precedence over other types of validation units defined in the default scenarios for the particular *framework*. For more information on *master files*, see *Master Files Support* on page 286 or *Working with Modular XML Files in the Master Files Context* on page 481.

To watch our video demonstration about how to use a validation scenario in Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/Validation_Scenario.html*.

Related Information:

Validating XML Documents Against a Schema on page 446
Presenting Validation Errors in Author Mode on page 236
Presenting Validation Errors in Text Mode on page 206

Creating a New Validation Scenario

To create a validation scenario, follow these steps:

1. Select the **Configure Validation Scenario(s)** from the **Validation** toolbar drop-down menu, or from the **Document** > **Validate** menu (or the **Validate** submenu when invoking the contextual menu on a file in the ***Project** view*).
   The **Configure Validation Scenario(s)** dialog box is displayed. It contains predefined and user-defined scenarios. The predefined scenarios are organized in categories depending on the type of file they apply to and you can identify them by a yellow key icon that marks them as *read-only*. The user-defined scenarios are organized under a single category. The default scenarios for the particular *framework* are rendered in bold.

   Note: If a *master file* is associated with the current file, the validation scenarios defined in the *master file*, along with any Schematron schema defined in the default scenarios for that particular *framework*, are used for the validation. These take precedence over other types of validation units defined in the default scenarios for the particular *framework*. For more information on *master files*, see *Master Files Support* on page 286 or *Working with Modular XML Files in the Master Files Context* on page 481.

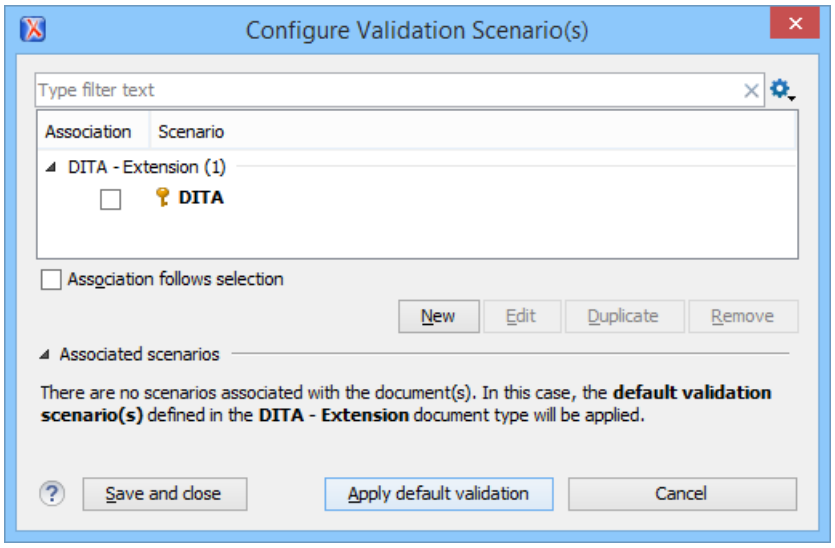


Figure 224: Configure Validation Scenario Dialog Box

The top section of the dialog box contains a filter that allows you to search through the scenarios list and the **Settings** button allows you to configure the following options:

Show all scenarios

Select this option to display all the available scenarios, regardless of the document they are associated with.

Show only the scenarios available for the editor

Select this option to only display the scenarios that Oxygen XML Editor can apply for the current document type.

Show associated scenarios

Select this option to only display the scenarios associated with the document you are editing.

Import scenarios

This option opens the **Import scenarios** dialog box that allows you to select the `scenarios` file that contains the scenarios you want to import. If one of the scenarios you import is identical to an existing scenario, Oxygen XML Editor ignores it. If a conflict appears (an imported scenario has the same name as an existing one), you can choose between two options:

- Keep or replace the existing scenario.
- Keep both scenarios.

  Note: When you keep both scenarios, Oxygen XML Editor adds `imported` to the name of the imported scenario.

Export selected scenarios

Use this option to export selected scenarios individually. Oxygen XML Editor creates a `scenarios` file that contains the scenarios that you export. This is useful if you want to share scenarios with others or export them to another computer.

2. To add a scenario, click the **New** button.
   The **New scenarios** dialog box is displayed and it lists all the validation units for the scenario.

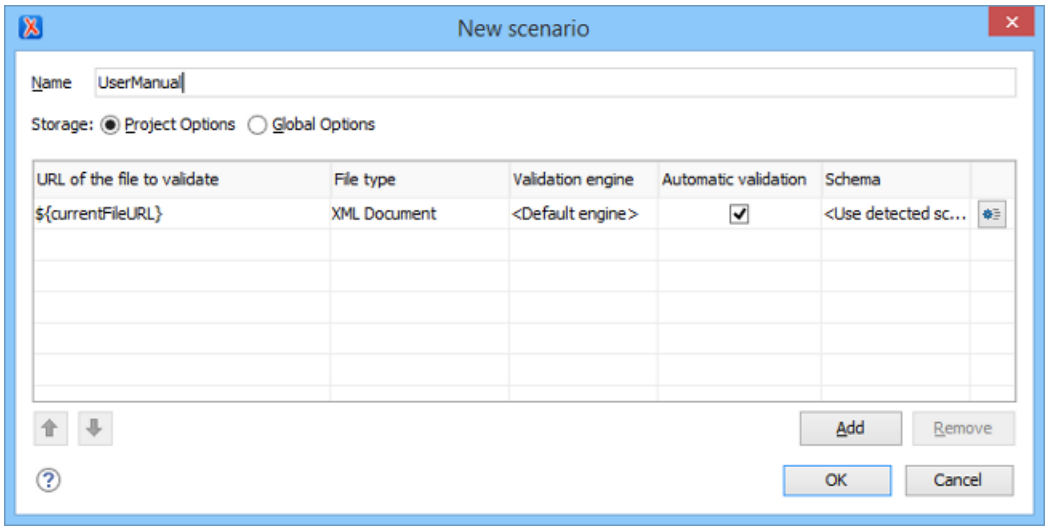


Figure 225: Create New Validation Scenario

This scenario configuration dialog box allows you to configure the following information and options:

Name

The name of the validation scenario.

Storage

You can choose between storing the scenario in the ***Project Options*** or ***Global Options***.

URL of the file to validate

The URL of the main module that includes the current module. It is also the entry module of the validation process when the current one is validated. To edit the URL, double-click its cell and specify the URL of the main module by doing one of the following:

- Enter the URL in the text field or select it from the drop-down list.
- Use the **Browse** drop-down button to browse for a local, remote, or archived file.
- Use the **Insert Editor Variable** button to insert an *editor variable* or a *custom editor variable*.

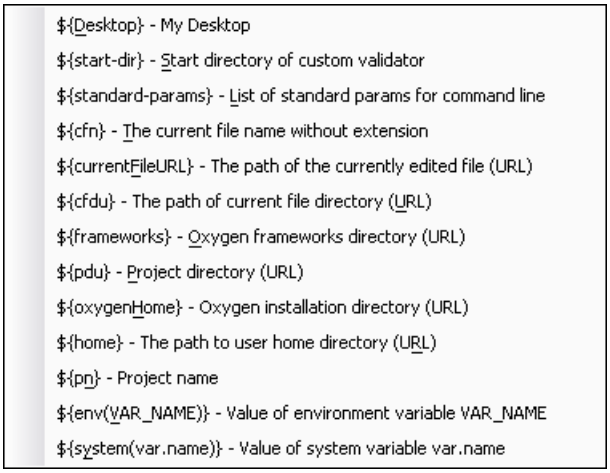


Figure 226: Insert an Editor Variable

File type

The type of the document that is validated in the current validation unit. Oxygen XML Editor automatically selects the file type depending on the value of the **URL of the file to validate** field.

Validation engine

You can select one of the engines available in Oxygen XML Editor for validation of the particular document type.

Default engine means that the default engine is used to run the validation for the current document type, as specified in the preferences page for that type of document (for example, *XSLT preferences page*, *XQuery preferences page*, *XML Schema preferences page*).

The **DITA Validation** engine performs DITA-specific checks in the context of the specifications (it is similar to the checks done with the **DITA Maps Manager** ***Validate and Check for Completeness*** *action*, but for a local file rather than an entire *DITA map*).

The **Table Layout Validation** engine looks for table layout problems (for more information, see *Report table layout problems* on page).

Automatic validation

If this option is selected, the validation operation defined by this row is also applied by *the automatic validation feature*. If the **Automatic validation** feature is disabled in the *Document Checking preferences page*, then this option is ignored, as the preference setting has a higher priority.

Schema

This option becomes active when you set the **File type** to **XML Document** and allows you to specify the schema used for the validation unit.

Specify Schema

Opens the **Specify Schema** dialog box that allows you to set a schema to be used for validating XML documents.

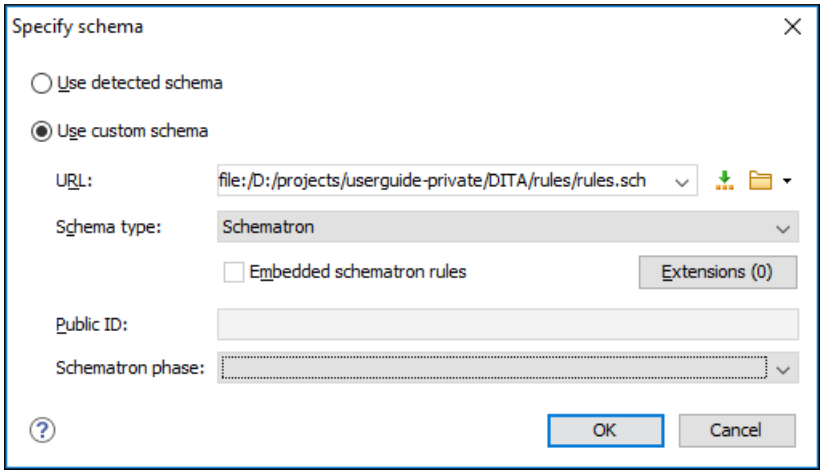


Figure 227: Specify Schema Dialog Box

The **Specify Schema** dialog box contains the following options:

Use detected schema

Uses the *schema detected for the particular document*.

Use custom schema

Allows you to specify the schema using the following options:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a *custom protocol*. You can specify the URL by using the text field, the history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.
- **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the **URL** field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.

- **Embedded schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.
- **Extensions**- Opens a dialog box that allows you to specify *Java extension JARs* to be used during the validation.
- **Public ID** - Allows you to specify a public ID if you have selected a DTD.
- **Schematron phase** - If you select a Schematron schema, this drop-down list allows you to select a Schematron phase that you want to use for validation. The listed phases are defined in the Schematron document.

Move Up

Moves the selected scenario up one spot in the list.

Move Down

Moves the selected scenario down one spot in the list.

Add

Adds a new validation unit to the list.

Remove

Removes an existing validation unit from the list.

3. Configure any of the existing validation units according to the information above, and you can use the buttons at the bottom of the table to add, remove, or move validation units. Note that if you add a Schematron validation unit, you can also select the **validation phase**.
4. Press **OK**.
   The newly created validation scenario will now be included in the list of scenarios in the **Configure Validation Scenario(s)** dialog box. You can select the scenario in this dialog box to associate it with the current document and press the **Apply associated** button to run the validation scenario.

Editing a Validation Scenario

To edit an existing validation scenario, follow these steps:

1. Select the **Configure Validation Scenario(s)** from the **Validation** toolbar drop-down menu, or from the **Document** > **Validate** menu (or the **Validate** submenu when invoking the contextual menu on a file in the ***Project*** *view*).
   The **Configure Validation Scenario(s)** dialog box is displayed. It contains predefined and user-defined scenarios. The predefined scenarios are organized in categories depending on the type of file they apply to and you can identify them by a yellow key icon that marks them as *read-only*. The user-defined scenarios are organized under a single category. The default scenarios for the particular *framework* are rendered in bold.

   Note: If a *master file* is associated with the current file, the validation scenarios defined in the *master file*, along with any Schematron schema defined in the default scenarios for that particular *framework*, are used for the validation. These take precedence over other types of validation units defined in the default scenarios for the particular *framework*. For more information on *master files*, see *Master Files Support* on page 286 or *Working with Modular XML Files in the Master Files Context* on page 481.

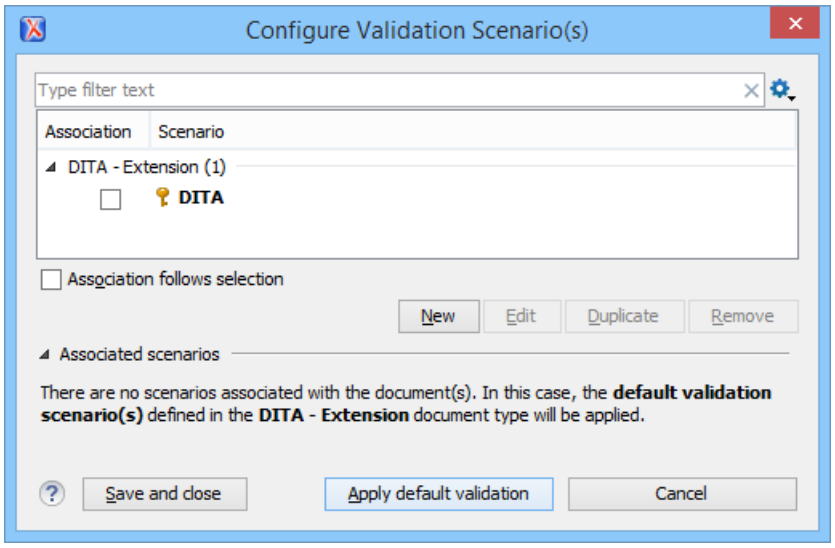


Figure 228: Configure Validation Scenario Dialog Box

The top section of the dialog box contains a filter that allows you to search through the scenarios list and the **Settings** button allows you to configure the following options:

Show all scenarios

Select this option to display all the available scenarios, regardless of the document they are associated with.

Show only the scenarios available for the editor

Select this option to only display the scenarios that Oxygen XML Editor can apply for the current document type.

Show associated scenarios

Select this option to only display the scenarios associated with the document you are editing.

Import scenarios

This option opens the **Import scenarios** dialog box that allows you to select the `scenarios` file that contains the scenarios you want to import. If one of the scenarios you import is identical to an existing scenario, Oxygen XML Editor ignores it. If a conflict appears (an imported scenario has the same name as an existing one), you can choose between two options:

- Keep or replace the existing scenario.
- Keep both scenarios.

  Note: When you keep both scenarios, Oxygen XML Editor adds `imported` to the name of the imported scenario.

Export selected scenarios

Use this option to export selected scenarios individually. Oxygen XML Editor creates a `scenarios` file that contains the scenarios that you export. This is useful if you want to share scenarios with others or export them to another computer.

2. Select the scenario and press the **Edit** button. If you try to edit one of the *read-only* predefined scenarios, you will receive a warning message that Oxygen XML Editor needs to creates customizable duplicate (you can also use the **Duplicate** button).
   The **Edit scenario** dialog box is displayed and it lists all the validation units for the scenario.

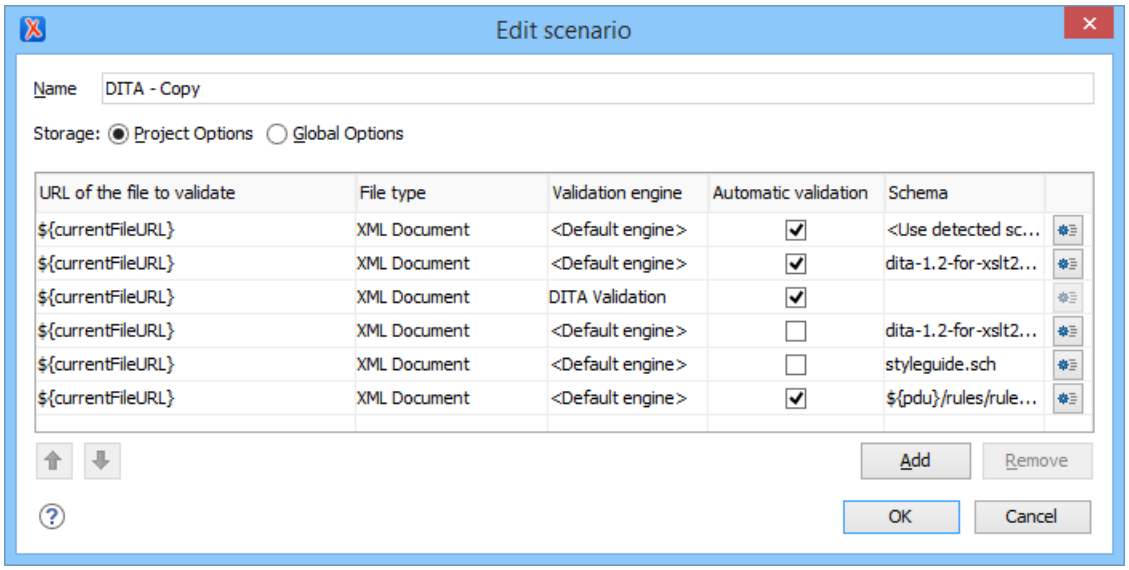


Figure 229: Edit Validation Scenario

This scenario configuration dialog box allows you to configure the following information and options:

Name

The name of the validation scenario.

Storage

You can choose between storing the scenario in the ***Project Options*** or ***Global Options***.

URL of the file to validate

The URL of the main module that includes the current module. It is also the entry module of the validation process when the current one is validated. To edit the URL, double-click its cell and specify the URL of the main module by doing one of the following:

- Enter the URL in the text field or select it from the drop-down list.
- Use the **Browse** drop-down button to browse for a local, remote, or archived file.
- Use the **Insert Editor Variable** button to insert an *editor variable* or a *custom editor variable*.

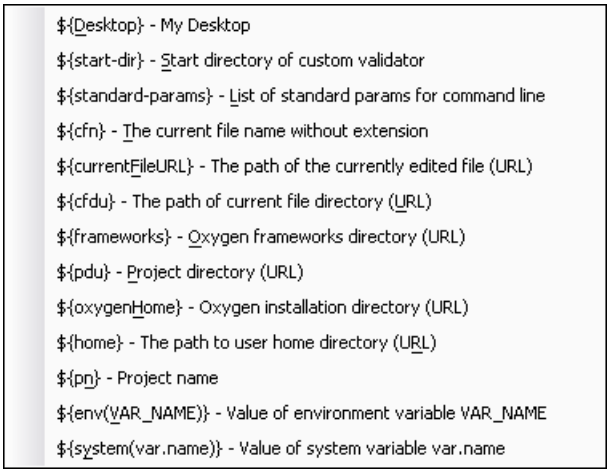


Figure 230: Insert an Editor Variable

File type

The type of the document that is validated in the current validation unit. Oxygen XML Editor automatically selects the file type depending on the value of the **URL of the file to validate** field.

Validation engine

You can select one of the engines available in Oxygen XML Editor for validation of the particular document type.

Default engine means that the default engine is used to run the validation for the current document type, as specified in the preferences page for that type of document (for example, *XSLT preferences page*, *XQuery preferences page*, *XML Schema preferences page*).

The **DITA Validation** engine performs DITA-specific checks in the context of the specifications (it is similar to the checks done with the **DITA Maps Manager** ***Validate and Check for Completeness*** *action*, but for a local file rather than an entire *DITA map*).

The **Table Layout Validation** engine looks for table layout problems (for more information, see *Report table layout problems* on page).

Automatic validation

If this option is selected, the validation operation defined by this row is also applied by *the automatic validation feature*. If the **Automatic validation** feature is disabled in the *Document Checking preferences page*, then this option is ignored, as the preference setting has a higher priority.

Schema

This option becomes active when you set the **File type** to **XML Document** and allows you to specify the schema used for the validation unit.

Specify Schema

Opens the **Specify Schema** dialog box that allows you to set a schema to be used for validating XML documents.

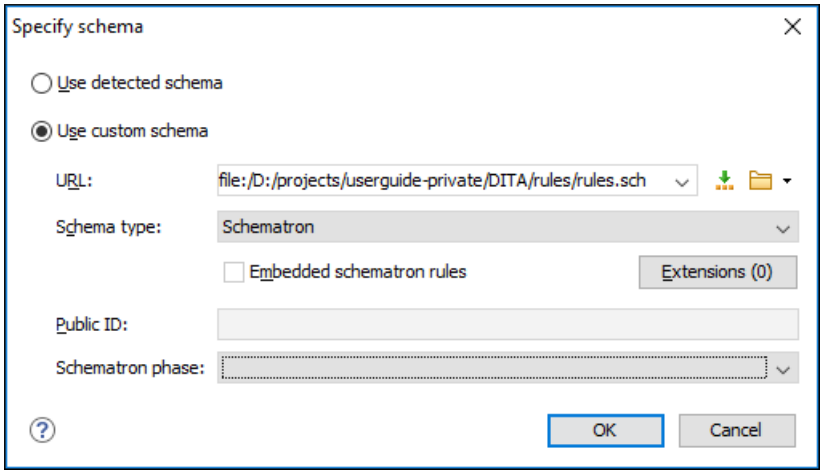


Figure 231: Specify Schema Dialog Box

The **Specify Schema** dialog box contains the following options:

Use detected schema

Uses the *schema detected for the particular document*.

Use custom schema

Allows you to specify the schema using the following options:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a *custom protocol*. You can specify the URL by using the text field, the history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.
- **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the **URL** field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.

- **Embedded schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.
- **Extensions**- Opens a dialog box that allows you to specify *Java extension JARs* to be used during the validation.
- **Public ID** - Allows you to specify a public ID if you have selected a DTD.
- **Schematron phase** - If you select a Schematron schema, this drop-down list allows you to select a Schematron phase that you want to use for validation. The listed phases are defined in the Schematron document.

Move Up

Moves the selected scenario up one spot in the list.

Move Down

Moves the selected scenario down one spot in the list.

Add

Adds a new validation unit to the list.

Remove

Removes an existing validation unit from the list.

3. Configure any of the existing validation units according to the information above, and you can use the buttons at the bottom of the table to add, remove, or move validation units. Note that if you add a Schematron validation unit, you can also select the **validation phase**.
4. When you are done configuring the scenario, press **OK**.
   The modified validation scenario will now be included in the list of scenarios in the **Configure Validation Scenario(s)** dialog box. If you chose to duplicate an existing one, the modified scenario will be listed with the word *copy* at the end of its name.

Sharing Validation Scenarios

The validation scenarios and their settings can be shared with other users by saving them at *project level* or by *exporting them to a specialized scenarios file* that can then be imported. When you create a new validation scenario or edit an existing one, there is a **Storage** option to control whether the scenarios are stored in ***Project Options*** or ***Global Options***.

Storage: ◉ Project Options ○ Global Options

Selecting ***Project Options*** stores the scenario in the project file and can be shared with other users that have access to the project. If your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) then your team will have access to the scenarios that you define. When you create a scenario at the project level, the URLs from the scenario become relative to the project URL.

Selecting ***Global Options*** stores the scenario in the global options that are stored in the user home directory.

You can also change the storage options on existing validation scenarios by using the **Change storage** action from the contextual menu of the list of scenarios.

Related Information:

Sharing Application Settings on page 163

References to XML Schema Specification

If validation is done against XML Schema, Oxygen XML Editor indicates a specification reference relevant for each validation error. The error messages contain an **Info** field that, when clicked, will open the browser on the *XML Schema Part 1:Structures* specification at exactly the point where the error is described. This allows you to understand the reason for that error.

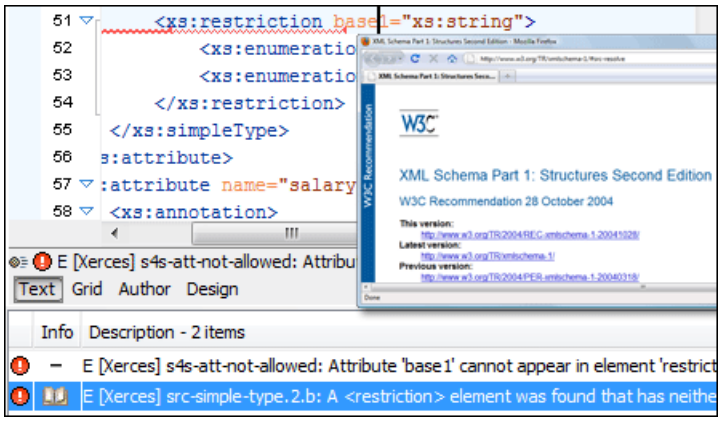


Figure 232: Link to Specification for XML Schema Errors

Resolving References to Remote Schemas with an XML Catalog

When a reference to a remote schema must be used in the validated XML document for interoperability purposes, but a local copy of the schema should actually be used for performance reasons, the reference can be resolved to the local copy of the schema with an *XML Catalog*.

For example, if the XML document contains a reference to a remote schema `docbook.rng` like this:

```
<?xml-model href="http://www.oasis-open.org/docbook/xml/5.0/rng/docbook.rng"
    type="application/xml" schematypens="http://relaxng.org/ns/structure/1.0"?>
```

it can be resolved to a local copy with a catalog entry like this:

```
<uri name="http://www.oasis-open.org/docbook/xml/5.0/rng/docbook.rng"
    uri="rng/docbook.rng"/>
```

An *XML Catalog* can also be used to map a W3C XML Schema specified with a URN in the `xsi:schemaLocation` attribute of an XML document to a local copy of the schema. For example, if the XML document specifies the schema with:

```
<topic xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="urn:oasis:names:tc:dita:xsd:topic.xsd:1.1">
```

the URN can be resolved to a local schema file with a catalog entry like this:

```
<uri name="urn:oasis:names:tc:dita:xsd:topic.xsd:1.1"
    uri="topic.xsd"/>
```

Related Information:

Working with XML Catalogs on page 484

Validation Example - A DocBook Validation Error

In the following DocBook 4 document, the content of the `listitem` element does not match the rules of the DocBook 4 schema (`docbookx.dtd`).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.4//EN"
     "http://www.docbook.org/xml/4.4/docbookx.dtd">
<article>
    <title>Article Title</title>
    <sect1>
      <title>Section1 Title</title>
      <itemizedlist>
        <listitem>
          <link>a link here</link>
        </listitem>
      </itemizedlist>
    </sect1>
</article>
```

The **Validate Document** action will return the following error:

```
Unexpected element "link". The content of the parent element type must match
"(calloutlist|glosslist|bibliolist|itemizedlist|orderedlist|segmentedlist|simplelist
|variablelist|caution|important|note|tip|warning|literallayout|programlisting
|programlistingco|screen|screenco|screenshot|synopsis|cmdsynopsis|funcsynopsis
|classsynopsis|fieldsynopsis|constructorsynopsis|destructorsynopsis|methodsynopsis
|formalpara|para|simpara|address|blockquote|graphic|graphicco|mediaobject|mediaobjectco
|informalequation|informalexample|informalfigure|informaltable|equation|example|figure
|table|msgset|procedure|sidebar|qandaset|task|anchor|bridgehead|remark|highlights
|abstract|authorblurb|epigraph|indexterm|beginpage)+".
```

This error message is a little more difficult to understand, so understanding of the syntax or processing rules for the DocBook XML DTD `listitem` element is recommended. However, the error message does offer a clue as to the source of the problem, indicating that “The content of element type must match”.

Fortunately, most standards-based DTDs, XML Schemas, and Relax NG schemas are supplied with reference documentation. This enables you to lookup the element and read about it. In this case, you should learn about the child elements of `listitem` and their nesting rules. Once you have correctly inserted the required child element and nested it in accordance with the XML rules, the document will become valid.

XML Quick Fixes

The Oxygen XML Editor *Quick Fix support* helps you resolve errors that appear in an XML document by offering *Quick Fixes* to problems such as missing required attributes or invalid elements. *Quick Fixes* are available in **Text** mode and **Author** mode

To activate this feature, hover over or place the cursor in the highlighted area of text where a validation error or warning occurs. If a *Quick Fix* is available for that particular error or warning, you can access the *Quick Fix* proposals with any of the following methods:

- When hovering over the error or warning, the proposals may be presented in a tooltip pop-up window and the available quick *Quick Fixes* include a link that can be used to perform the fix.

Figure 233: Quick Fix Presented in a Tooltip in Text Mode

Figure 234: Quick Fix Presented in a Tooltip in Author Mode

- When hovering over the error or warning in **Author** mode, a small *Quick Fix* drop-down menu is presented. You can use the drop-down menu to display a list of available *Quick Fixes* to select from for the particular error or warning.

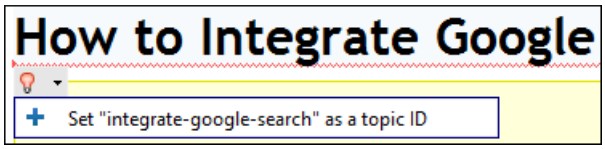


Figure 235: Quick Fix Drop-Down Menu in Author Mode

- If you place the cursor in the highlighted area where a validation error or warning occurs, a *Quick Fix* icon ( ) is displayed in the stripe on the left side of the editor. If you click this icon, Oxygen XML Editor displays the list of available fixes.

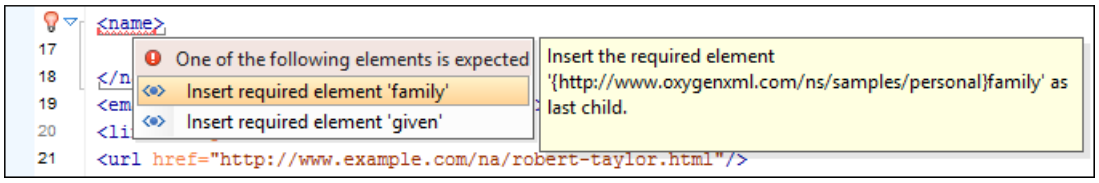


Figure 236: Quick Fix Menu Invoked by Clicking on the Icon

- With the cursor placed in the highlighted area of the error or warning, you can also invoke the *Quick Fix* menu by pressing **Alt + 1 (Command + Alt + 1 on OS X)** on your keyboard.

Whenever you make a modification in the XML document or you apply a fix, the list of *Quick Fixes* is recomputed to ensure that you always have valid proposals.

Note: A *Quick Fix* that adds an element inserts it along with required and optional elements, and required and fixed attributes, depending on how the ***Content Completion*** *preferences* are configured.

Quick Fixes for DTD, XSD, and Relax NG Errors

Oxygen XML Editor offers *Quick Fixes* for common errors that appear in XML documents that are validated against DTD, XSD, or Relax NG schemas.

Note: For XML documents validated against XSD schemas, the *Quick Fixes* are only available if you use the default Xerces validation engine.

Quick Fixes are available in **Text** mode and **Author** mode.

Oxygen XML Editor provides *Quick Fixes* for numerous types of problems, including the following:

Problem Type	Available Quick Fixes
A specific element is required in the current context	Insert the required element
An element is invalid in the current context	Remove the invalid element
The content of the element should be empty	Remove the element content
An element is not allowed to have child elements	Remove all child elements
Text is not allowed in the current element	Remove the text content
A required attribute is missing	Insert the required attribute
An attribute is not allowed to be set for the current element	Remove the attribute
The attribute value is invalid	Propose the correct attribute values
ID value is already defined	Generate a unique ID value
References to an invalid ID	Change the reference to an already defined ID

Related Information:

Schematron Quick Fixes (SQF) on page 463

Schematron Quick Fixes (SQF)

Oxygen XML Editor provides support for Schematron *Quick Fixes* (SQF). They help you resolve issues that appear in XML documents that are validated against Schematron schemas by offering you solution proposals. The Schematron *Quick Fixes* are an extension of the Schematron language and they allow you to define fixes for Schematron validation messages. Specifically, they are associated with *assert* or *report* messages.

A typical use case is using Schematron *Quick Fixes* to assist content authors with common editing tasks. For example, you can use Schematron rules to automatically report certain validation warnings (or errors) when performing regular editing tasks, such as inserting specific elements or changing IDs to match

specific naming conventions. For more details and examples, please see the following blog post: *http://blog.oxygenxml.com/2015/05/schematron-checks-to-help-technical.html*.

Displaying the Schematron Quick Fix Proposals

The defined Schematron *Quick Fixes* are displayed on validation errors in **Text** mode and **Author** mode.

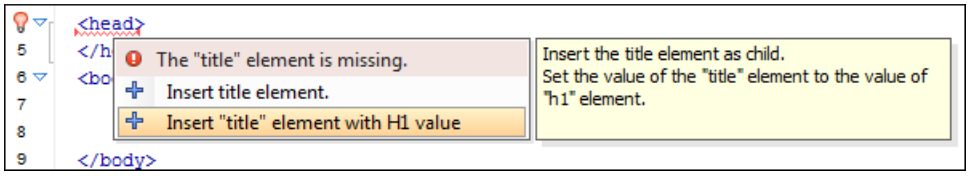


Figure 237: Example of a Schematron Quick Fix

Related Information:

Editing Schematron Quick Fixes on page 695
Schematron Quick Fix Specifications
Presenting Schematron Validation Issues

Associating a Schema to XML Documents

Oxygen XML Editor relies on schemas to validate XML documents and to compute valid proposals for the *Content Completion Assistant*.

Supported Types of Schema

The following schema types are supported:

- W3C XML Schema 1.0 and 1.1 (with and without embedded Schematron rules)
- DTD
- Relax NG - XML syntax (with and without embedded Schematron rules)
- Relax NG - compact syntax
- NVDL
- Schematron (both ISO Schematron and Schematron 1.5)

Detecting a Schema

For the purposes of using validation and content completion mechanisms, Oxygen XML Editor tries to detect a schema by searching multiple locations, in the following order:

- The *schema defined in a validation scenario*.
- The *schema defined in the validation scenarios associated with the particular document type* (if defined).
- The *schema that is associated directly in the XML document*.

  Note: If a DTD schema is specified in the document, the content completion for **Author** mode is based on this schema (even if there is already one detected from the validation scenario).
- The *schema defined in the framework (document type) configuration*.

In addition, the locations of the schema can be mapped through the use of an *XML Catalog*. For more information, see *Resolving Schema Locations Through XML Catalogs* on page 471.

Tip: To quickly open the schema used for validating the current document, select the **Open Associated Schema** action from the toolbar (or **Document** > **Schema** menu).

Related Information:

Working with Modular XML Files in the Master Files Context

Associating a Schema Through a Validation Scenario

Oxygen XML Editor uses the rules defined in the detected schema to report errors and warnings during automatic and manual validations that help maintain the structural integrity of your XML documents. You can specify the schema to be used for validation directly in *validation scenarios* and there are several methods that can be used to do so.

Configure a Validation Scenario and Specify the Schema

To associate a schema to a validation scenario to be used whenever the scenario is invoked, follow these steps:

1. Select the **Configure Validation Scenario(s)** from the **Validation** toolbar drop-down menu, or from the **Document** > **Validate** menu (or the **Validate** submenu when invoking the contextual menu on a file in the ***Project*** *view*).
2. Press the **New** button to create a new validation scenario or the **Edit** button to modify an existing one.
3. *Add or configure validation units* according to your needs and click the **Specify Schema** button.

   Step Result: The **Specify Schema** dialog box is displayed:

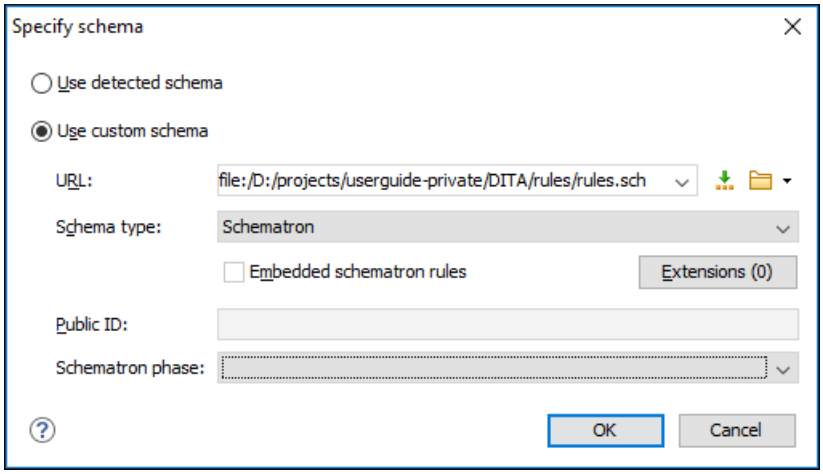


Figure 238: Specify Schema Dialog Box

The **Specify Schema** dialog box contains the following options:

Use detected schema

Uses the *schema detected for the particular document*.

Use custom schema

Allows you to specify the schema using the following options:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a *custom protocol*. You can specify the URL by using the text field, the history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.
- **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the **URL** field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.
- **Embedded schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.
- **Public ID** - Allows you to specify a public ID if you have selected a DTD.
- **Extensions**- Opens a dialog box that allows you to specify *Java extension JARs* to be used during the validation.

- **Schematron phase** - If you select a Schematron schema, this drop-down list allows you to select a Schematron phase that you want to use for validation. The listed phases are defined in the Schematron document.

4. Select the schema to be associated with the validation unit and configure the rest of the options according to your preferences.
5. Click **OK** on both dialog boxes.

Result: The schema is now associated with that validation scenario whenever it is invoked.

Use the Validate with Action to Specify a Schema for Validating the Current Document

To validate the current document using a specified schema, follow these steps:

1. Select the **Validation with** action from the **Validation** drop-down menu on the toolbar (or **Document** > **Validate** menu).

   Step Result: The **Validate with** dialog box is displayed:

   Figure 239: Validate with Dialog Box

   This dialog box contains the following options:

   - **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a *custom protocol*. You can specify the URL by using the text field, the history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.
   - **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the **URL** field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.
   - **Embedded schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.
   - **Public ID** - Allows you to specify a public ID if you have selected a DTD.
   - **Extensions**- Opens a dialog box that allows you to specify *Java extension JARs* to be used during the validation.
   - **Schematron phase** - If you select a Schematron schema, this drop-down list allows you to select a Schematron phase that you want to use for validation. The listed phases are defined in the Schematron document.

2. Select the schema to be associated with the manual validation and configure the rest of the options according to your preferences.
3. Click **OK**.

Result: The current document is validated using the schema you specified.

Tip: To quickly open the schema used for validating the current document, select the **Open Associated Schema** action from the toolbar (or **Document** > **Schema** menu).

Use the Validate with Schema Action to Specify a Schema for Validating all Selected Documents

To validate multiple documents using a specified schema, follow these steps:

1. Select all the documents you want to validate in the **Project** view .
2. Invoke the contextual menu (right-click) and select the **Validate with Schema** action from the **Validate** submenu.

   Step Result: The **Validate with** dialog box is displayed:

   Figure 240: Validate with Dialog Box

   This dialog box contains the following options:

   - **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a *custom protocol*. You can specify the URL by using the text field, the history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.
   - **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the **URL** field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.
   - **Embedded schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.
   - **Public ID** - Allows you to specify a public ID if you have selected a DTD.
   - **Extensions**- Opens a dialog box that allows you to specify *Java extension JARs* to be used during the validation.
   - **Schematron phase** - If you select a Schematron schema, this drop-down list allows you to select a Schematron phase that you want to use for validation. The listed phases are defined in the Schematron document.
3. Select the schema that you want to use to validate all selected documents and configure the rest of the options according to your preferences.
4. Click **OK**.

Result: The selected documents are validated using the schema you specified.

Associating a Schema in Validation Scenarios Defined in the Document Type

To report errors and warnings during automatic and manual validations that help maintain the structural integrity of particular XML document types, Oxygen XML Editor uses rules defined in the schema that is detected in the validation scenarios that are associated to each particular document type.

To associate a schema in validation scenarios defined in the *framework* (document type) configuration, follow these steps:

1. *Open the **Preferences** dialog box **(Options > Preferences)*** and go to **Document Type Association**.
2. Select your particular document type and click the **Edit** or **Duplicate** button to modify an existing *framework* (or use the **New** button to create a new one).

   Step Result: This opens a ***Document type** configuration dialog box*.
3. Go to the ***Validation** tab*.
4. Create or edit a validation scenario:

   a. To *create a new validation scenario*, click the **New** button.

b. To *edit the properties of an existing validation scenario*, select it and click the **Edit** button (you can also use the **Duplicate** button to copy an existing scenario and edit its properties).

5. *Add or configure validation units* according to your needs and click the **Specify Schema** button.

   Step Result: The **Specify Schema** dialog box is displayed:

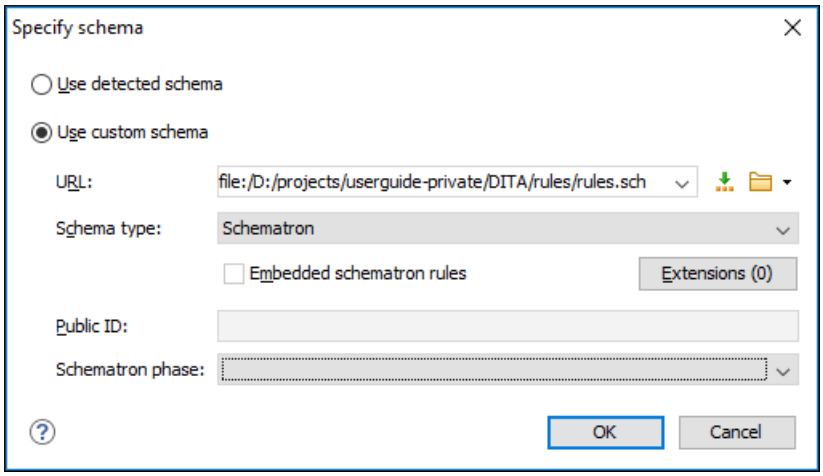


Figure 241: Specify Schema Dialog Box

The **Specify Schema** dialog box contains the following options:

Use detected schema

Uses the *schema detected for the particular document*.

Use custom schema

Allows you to specify the schema using the following options:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a *custom protocol*. You can specify the URL by using the text field, the history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.
- **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the **URL** field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.
- **Embedded schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.
- **Public ID** - Allows you to specify a public ID if you have selected a DTD.
- **Extensions**- Opens a dialog box that allows you to specify *Java extension JARs* to be used during the validation.
- **Schematron phase** - If you select a Schematron schema, this drop-down list allows you to select a Schematron phase that you want to use for validation. The listed phases are defined in the Schematron document.

6. Select the schema to be associated with the validation unit and configure the rest of the options according to your preferences.
7. Click **OK** on both dialog boxes.

Result: The schema is now associated with the validation scenario you just configured for that particular document type.

Associate Schema Action

The schema used by the *Content Completion Assistant* and document validation engine can be associated with the current document by using the **Associate Schema** action. For most of the schema types, it uses *the `xml-model` processing instruction*, with the exceptions of:

- **W3C XML Schema** - The `xsi:schemaLocation` attribute is used.
- **DTD** - The `DOCTYPE` declaration is used.

The association can specify a relative file path or a URL of the schema. The advantage of relative file path is that you can configure the schema at file level instead of *framework* level.

To associate a schema to the current document, follow these steps:

1. Select the **Associate Schema** action from the toolbar (or **Document** > **Schema** menu).

   Step Result: The **Associate Schema** dialog box is displayed:

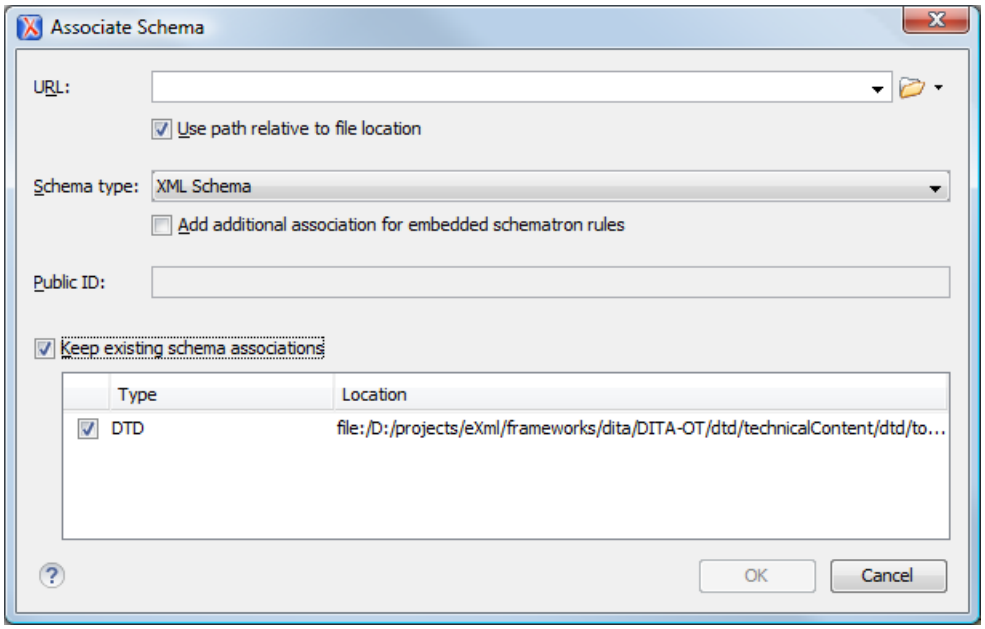


Figure 242: Associate Schema Dialog Box

This dialog box contains the following options:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a *custom protocol*.
- **Use path relative to file location** - Select this option if the XML instance document and the associated schema contain relative paths. The location of the schema file is inserted in the XML instance document as a relative file path. This practice allows you, for example, to share these documents with other users without running into problems caused by multiple project locations on physical disk.
- **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the **URL** field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.
- **Add additional association for embedded schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.
- **Public ID** - Allows you to specify a public ID if you have selected a DTD.
- **Keep existing schema associations** - Select this option to use the existing schema associations of the currently edited document.

2. Select the schema that will be associated with the XML document and configure the rest of the options according to your preferences.
3. Click **OK**.

Result: The schema association is created based upon the specified type.

- **XML Schema** - The association with an XML Schema is added as an attribute of the root element with one of the following:
  - `xsi:schemaLocation` attribute, if the root element of the document sets a default namespace with an `xmlns` attribute.
  - `xsi:noNamespaceSchemaLocation` attribute, if the root element does not set a default namespace.
- **DTD** - The association with a DTD is added as a `DOCTYPE` declaration.
- **Other** - The association with a Relax NG, Schematron, or NVDL schema is added as *`xml-model` processing instruction*.

Tip: To quickly open the schema used for validating the current document, select the **Open Associated Schema** action from the toolbar (or **Document** > **Schema** menu).

Associate Schema with the `xml-model` Processing Instruction

The `xml-model` processing instruction associates a schema with the XML document that contains the processing instruction. It must be added at the beginning of the document, just after the XML prolog. The following code snippet contains an `xml-model` processing instruction declaration:

```
<?xml-model href="../schema.sch" type="application/xml"
 schematypens="http://purl.oclc.org/dsdl/schematron" phase="ALL"
 title="Main schema"?>
```

It is available in the *Content Completion Assistant*, before XML document root element, and includes the following attributes:

- `href` (required) - The schema file location.
- `type` - The content type of the schema. This is an optional attribute with the following possible values for each specified type:
  - DTD - The recommended value is `application/xml-dtd`.
  - W3C XML Schema - The recommended value is `application/xml`, or can be left unspecified.
  - RELAX NG XML Syntax - The recommended value is `application/xml`, or can be left unspecified.
  - RELAX NG Compact Syntax - The recommended value is `application/relax-ng-compact-syntax`.
  - Schematron - The recommended value is `application/xml`, or can be left unspecified.
  - NVDL - The recommended value is `application/xml`, or can be left unspecified.
- `schematypens` - The namespace for the schema language of the referenced schema with the following possible values:
  - DTD - Not specified.
  - W3C XML Schema - The recommended value is `http://www.w3.org/2001/XMLSchema`.
  - RELAX NG XML Syntax - The recommended value is `http://relaxng.org/ns/structure/1.0`.
  - RELAX NG Compact Syntax - Not specified.
  - Schematron - The recommended value is `http://purl.oclc.org/dsdl/schematron`.
  - NVDL - The recommended value is `http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0`.
- `phase` - The phase name for the validation function in Schematron schema. This is an optional attribute. To run all phases from the Schematron, use the special `#ALL` value. If the phase is not specified, the default phase that is configured in the Schematron will be applied.
- `title` - The title for the associated schema. This is an optional attribute.

Older versions of Oxygen XML Editor used the `oxygen` processing instruction with the following attributes:

- `RNGSchema` - Specifies the path to the Relax NG schema that is associated with the current document.
- `type` - Specifies the type of Relax NG schema. It is used along with the `RNGSchema` attribute and can have the value `xml` or `compact`.

- `NVDLSchema` - Specifies the path to the NVDL schema that is associated with the current document.
- `SCHSchema` - Specifies the path to the SCH schema that is associated with the current document.

Note: Documents that use the `oxygen` processing instruction are compatible with newer versions of Oxygen XML Editor.

Related Information:

Validating XML Documents on page 444
Content Completion Assistant in Text Mode on page 300
Content Completion Assistant in Author Mode on page 345

Associating a Schema in a Framework (Document Type) Configuration

The schema used to compute valid proposals in the *Content Completion Assistant* and by the document validation engine to report errors and warnings can be defined in each particular *framework* (document type). This schema will be used only if one is not *detected in the current XML file*.

To associate a schema in a particular *framework* (document type), follow these steps:

1. *Open the **Preferences** dialog box **(Options > Preferences)*** and go to **Document Type Association**.
2. Select your particular document type and click the ***Edit***, ***Extend***, or ***Duplicate*** button to modify an existing *framework* (or use the **New** button to create a new one).

   Step Result: This opens a ***Document type** configuration dialog box*.
3. Go to the ***Schema** tab*.
4. Select the schema type and its URI.
5. Click **OK**.

Result: The schema is now associated with the particular document type and will be used by the *Content Completion Assistant* and validation engine if a schema is not detected in the current XML document.

Resolving Schema Locations Through XML Catalogs

Schema locations can be mapped using an *XML Catalog*. Oxygen XML Editor resolves the location of a schema in the following order:

- First, it attempts to resolve the schema location as a URI (`uri`, `uriSuffix`, `rewriteUri`, `delegateUri` mappings from the *XML Catalog*). If this succeeds, the process end here.
- If the ***Resolve schema locations also through system mappings** option* is selected, it attempts to resolve the schema location as a system ID (`system`, `systemSuffix`, `rewriteSuffix`, `rerwriteSystem` from the *XML Catalog*). If this succeeds, the process ends here.
- If the ***Process "schemaLocation" namespaces through URI mappings for XML Schema** option* is selected, the target namespace of the imported XML Schema is resolved through URI mappings. If the schema specified in the *schemaLocation* attribute is not resolved successfully, the namespace of the root element is taken into account. If this succeeds, the process ends here.
- If none of these succeeds, the actual *schema location* is used.

Related Information:

Working with XML Catalogs on page 484

Learn Document Structure when Schema is not Detected

When working with documents that do not specify a schema, or for which the schema is not known or does not exist, Oxygen XML Editor is able to learn and translate the document structure to a DTD. You can choose to save the learned structure to a file to provide a DTD as an initialization source for *content completion* and *document validation*. This feature is also useful for producing DTDs for documents that contain personal or custom element types.

When you open a document that is not associated with a schema, Oxygen XML Editor automatically learns the document structure and uses it for *content completion*. To disable this feature, deselect the ***Learn on open document** option in the user preferences*.

Related Information:

Detecting a Schema on page 464

Create a DTD from Learn Document Structure Option

When there is no schema associated with an XML document, Oxygen XML Editor can learn the document structure by parsing the document internally. This feature is enabled by the ***Learn on open document*** *option* that is available in the user preferences.

To create a DTD from the learned structure, follow these steps:

1. Open the XML document for which a DTD will be created.
2. Go to **Document** > **XML Document** > **Learn Structure** (**Ctrl + Shift + L (Command + Shift + L on OS X)**). The **Learn Structure** action reads the mark-up structure of the current document. The **Learn completed** message is displayed in the application status bar when the action is finished.
3. Go to **Document** > **XML Document** > **Save Structure** (**Ctrl + Shift + S (Command + Shift + S on OS X)**) and enter the DTD file path.
4. Press the **Save** button.

Finding and Replacing Text in the Current File

This section walks you through the find and replace features available in Oxygen XML Editor.

You can use a number of advanced views depending on what you need to find in the document you are editing or in your entire project. The ***Find/Replace*** *dialog box* allows you to search through the current project or selected resources and offers a set of options to improve your search. The ***Find All Elements/Attributes*** *dialog box* allows you to search through the structure of the current document for elements and attributes.

As an alternative to the dedicated search operations, you can also use the ***Quick Find*** *toolbar*.

Find/Replace Dialog Box

To open the **Find/Replace** dialog box, use the **Find/Replace** action that is available in the **Find** menu, on the toolbar, or by pressing **Ctrl + F (Command + F on OS X)**. It is also invoked by the **Find/Replace** contextual menu action found in certain views.

You can use the **Find/Replace** dialog box to perform the following operations:

- Replace occurrences of target defined in the **Find** area with a new fragment of text defined in **Replace with** area.
- Find all the occurrences of a word or string of characters (that can span over multiple lines) in the document you are editing. This operation also takes into account all the white spaces contained in the fragment you are searching for.

Note: The **Find/Replace** dialog box counts the number of occurrences of the text you are searching for and displays it at the bottom of the dialog box, above the **Close** button. This number is also displayed in *the* ***Results*** *view*.

The *find* operation works on multiple lines, meaning that a find match can cover characters on multiple lines of text. To input multiple-line text in the **Find** and **Replace with** areas, do one of the following:

- Press **Ctrl + Enter (Command + Enter on OS X)** on your keyboard.
- Use the **Insert newline** contextual menu action.

You can use *Perl-like regular expressions syntax* to define patterns. A content completion assistance window is available in the **Find** and **Replace with** areas to help you edit regular expressions. It is activated every time you type ****(backslash key) or on-demand if you press **Ctrl + Space (Command + Space on OS X)** on your keyboard.

The *replace* operation can bind regular expression capturing groups ($1 , $2, etc.) from the find pattern.

Tip: To replace the `tag-name` start tag and its attributes with the `new-tag-name` tag use as **Find** the expression *<tag-name(\s+)(.*)>* and as **Replace with** the expression *<new-tag-name$1$2>*.

The **Find/Replace** dialog box contains the following options:

- **Find** - The target character string to search for. You can search for Unicode characters specified in the `\uNNNN` format. Also, hexadecimal notation (`\xNNNN`) and octal notation (`\0NNNN`) can be used. In this case you have to select the ***Regular expression*** *option*. For example, to search for a space character you can use the `\u0020` code.

- **Replace with** - The character string with which to replace the target. The string for replace can be on a line or on multiple lines. It can contain Perl notation capturing groups, only if the search expression is a regular expression and the ***Regular expression*** *option* is selected.

  Note: Some regular expressions can indefinitely block the application. If the execution of the regular expression does not end in about 5 seconds, the application displays a dialog box that allows you to interrupt the operation.

  Tip: Special characters such as *newline* and *tab* can be inserted in the **Find** and **Replace with** text boxes using dedicated actions in the contextual menu (**Insert newline** and **Insert tab**).

  Unicode characters in the `\uNNNN` format can also be used in the **Replace with** area.
- The **History** button - Contain lists of the last find and replace expressions. Use the **Clear history** action from the bottom of the lists to remove these expressions.
- **XPath** - The XPath 2.0 expression you input in this combo is used for restricting the search scope.

  Note: The *Content Completion Assistant* helps you input XPath expressions, valid in the current context.
- **Direction** - Specifies if the search direction is from current position to end of file (**Forward**) or to start of file (**Backward**).
- **Scope** - Specifies whether the **Find/Replace** operation is executed over the entire content of the edited document (**All** option), or over the selected lines of text (**Only selected lines** option). If the selection spans across multiple lines, when you open the **Find/Replace** dialog box, the scope is set to **Only selected lines**.
- **Case sensitive** - When selected, the search operation follows the exact letter case of the text entered in the **Find** field.
- **Whole words only** - Only entire occurrences of a word are included in the search operation.
- **Incremental** - The search operation is started every time you type or delete a letter in the **Find** text box.
- **Regular expression** - When this option is selected, you can use regular expressions in *Perl-like regular expressions syntax* to look for specific pieces of text.
  - **Dot matches all** - A dot used in a regular expression also matches end of line characters.
  - **Canonical equivalence** - If selected, two characters will be considered a match if, and only if, their full *canonical* decompositions match. For example, the **ã** symbol can be inserted as a single character or as two characters (the **a** character, followed by the tilde character). This option is not selected by default.
- **Wrap around** - When the end of the document is reached, the search operation is continued from the start of the document, until its entire content is covered.
- **Enable XML search options** - This option is only available when editing in **Text** mode. It provides access to a set of options that allow you to search specific XML component types:
  - **Element names** - Only the element names are included in the search operation that ignores XML-tag notations ('<', '/', '>'), attributes or white-spaces.
  - **Element contents** - Search in the text content of XML elements.
  - **Attribute names** - Only the attribute names are included in the search operation, without the leading or trailing white-spaces.
  - **Attribute values** - Only the attribute values are included in the search operation, without single quotes(') or double quotes(").
  - **Comments** - Only the content of comments is included in the search operation, excluding the XML comment delimiters ('<!--', '-->').
  - **PIs** (Processing Instructions) - Only the content are searched, skipping '<?', '?>'. e. g.: <?processing instruction?>
  - **CDATA** - Searches inside content of CDATA sections.
  - **DOCTYPE** - Searches inside content of DOCTYPE sections.
  - **Entities** - Only the entity names are searched.

  The two buttons **Select All** and **Deselect All** allow a simple activation and deactivation of all types of XML components.

  Note: Even if you select all options of the **Enable XML search options** section, the search is still XML-aware. If you want to perform the search over the entire file content, deselect **Enable XML search options**.
- **Find All Elements** - Available when editing in **Author** mode, you can use this link to extend the search scope to XML-specific markup (names and values of both attributes and elements).

- **Find** - Executes a find operation for the next occurrence of the target. It stops after highlighting the find match in the editor panel.
- **Replace** - Executes a replace operation for the target followed by a find operation for the next occurrence.
- **Find All** - Executes a find operation and displays all results in the ***Results*** *view*. The results are *displayed in the* ***Results*** *view*.
- **Replace All** - Executes a replace operation in the entire scope of the document.
- **Replace to End** - Executes a replace operation starting from current target until the end of the document, in the direction specified by the current selection of the **Direction** switch (**Forward** or **Backward**).

Find All Elements Dialog Box

To open the **Find All Elements** dialog box, go to **Find** > **Find All Elements**(**Ctrl + Shift + E (Command + Shift + E on OS X)**) or from the shortcut **Find All Elements** that is available in *the* ***Find / Replace*** *dialog box*. It assists you in defining XML element / attribute search operations in the current document.

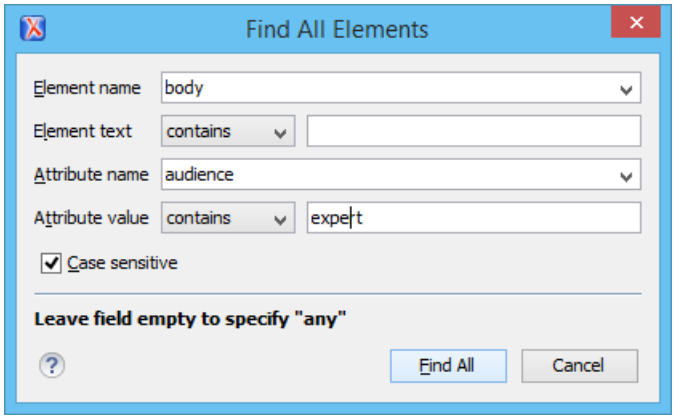


Figure 243: Find All Elements Dialog Box

The dialog box can perform the following actions:

- Find all the elements with a specified name
- Find all the elements that contain, or does not contain, a specified string in their text content
- Find all the elements that have a specified attribute
- Find all the elements that have an attribute with, or without, a specified value

You can combine all of these search criteria to filter your results.

The following fields are available in the dialog box:

- **Element name** - The qualified name of the target element to search for. You can use the drop-down menu to find an element or enter it manually. It is populated with valid element names collected from the associated schema. To specify *any* element name, leave the field empty.

  Note: Use the qualified name of the element (`<namespace prefix>:<element name>`) when the document uses this element notation.
- **Element text** - The target element text to search for. The drop-down menu beside this field allows you to specify whether you are looking for an exact or partial match of the element text. For *any* element text, select **contains** from the drop-down menu and leave the field empty. If you leave the field empty but select **equals** from the drop-down menu, only elements with no text will be found. Select **not contains** to find all elements that do not include the specified text.
- **Attribute name** - The name of the attribute that must be present in the element. You can use the drop-down menu to select an attribute or enter it manually. It is populated with valid attribute names collected from the associated schema. For *any* or no attribute name, leave the field empty.

  Note: Use the qualified name of the attribute (`<namespace prefix>:<attribute name>`) when the document uses this attribute notation.
- **Attribute value** - The drop-down menu beside this field allows you to specify that you are looking for an exact or partial match of the attribute value. For *any* or no attribute value, select **contains** from the drop-down menu and leave the field empty. If you leave the field empty but select **equals** from the drop-down menu,

only elements that have at least an attribute with an empty value will be found. Select **not contains** to find all elements that have attributes without a specified value.
- **Case sensitive** - When this option is selected, operations are case-sensitive

When you press **Find All**, Oxygen XML Editor tries to find the items that match all the search parameters. The results of the operation are presented as a list in the message panel.

Quick Find Toolbar

A reduced version of the ***Find / Replace*** *dialog box* is available as *a dockable toolbar*. To display it press the **Alt + Shift + F (Command + Alt + F on OS X)** key combination or invoke the **Find** > **Quick Find** action. By default, the toolbar is displayed at the bottom of the Oxygen XML Editor window, above the status bar, but can be changed at any time by dragging (and docking) it to a different location. To hide the toolbar, use the **Close** button.

All matches are highlighted in the current editor.

Figure 244: Quick Find Toolbar

The toolbar offers the following controls:

- Search input box - This is where you can insert the text you want to search for. The input box keeps a history of the last used search text. The background color of the input box turns red when no match is found.
- **Next** - Advances to the next match.
- **Previous** - Jumps to the previous match.
- **All** - Highlights all matches of the search string in the current document.
- **Incremental** - If selected, the search operation is started every time you type or delete a character in the search input box.
- **Case sensitive** - If selected, the search operation follows the exact letter case of the search text.
- **Find/Replace** - Opens the ***Find/Replace*** *dialog box*.
- **Find/Replace in Files** - Opens the ***Find/Replace in Files*** *dialog box*.
- **Close** - Closes the **Quick Find** toolbar.

Keyboard Shortcuts for Finding the Next and Previous Match

Navigating from one match to the next or previous one is very easy to perform using the **F3** and **Shift + F3 (Command + Shift + G on OS X)** keyboard shortcuts. They are useful for quickly repeating the last find action performed in the ***Find / Replace*** *dialog box*, taking into account the same find options.

Restriction: These shortcuts only take XPath expressions into account if the **Find / Replace** dialog box remains opened. Once you close it, the XPath expressions are no longer considered.

Regular Expressions Syntax

Oxygen XML Editor uses the *Java regular expression syntax*. It is **similar** to that used in Perl 5, with several exceptions. Thus, Oxygen XML Editor does not support the following constructs:

- The conditional constructs `(?{X})` and `(?(condition)X|Y)`.
- The embedded code constructs `(?{code})` and `(??{code})`.
- The embedded comment syntax `(?#comment)`.
- The preprocessing operations `\l`, `\u`, `\L`, and `\U`.

When using regular expressions, note that some sets of characters from *XPath/XML Schema/Schematron* are slightly different than the ones used by Oxygen XML Editor/Java in the text searches from the ***Find/Replace*** *dialog box* and ***Find/Replace in Files*** *dialog box*. The most common example is with the `\w` and `\W` set of characters. To ensure consistent results between the two, it is recommended that you use the following constructs in the ***Find/Replace*** *dialog box* and ***Find/Replace in Files*** *dialog box*:

- **/w** - `[#x0000-#x10FFFF]-[\p{P}\p{Z}\p{C}]` instead of `\w`
- **/W** - `[\p{P}\p{Z}\p{C}]` instead of `\W`

There are some other notable differences that may cause unexpected results, including the following:

- In Perl, `\1` through `\9` are always interpreted as back references. A backslash-escaped number greater than 9 is treated as a back reference if at least that many sub-expressions exist. Otherwise, it is interpreted, if possible, as an octal escape. In this class octal escapes must always begin with a zero. In Java, `\1` through `\9` are always interpreted as back references, and a larger number is accepted as a back reference if at least that many sub-expressions exist at that point in the regular expression. Otherwise, the parser will drop digits until the number is smaller or equal to the existing number of groups or it is one digit.
- Perl uses the g flag to request a match that resumes where the last match left off.
- In Perl, embedded flags at the top level of an expression affect the whole expression. In Java, embedded flags always take effect at the point where they appear, whether they are at the top level or within a group. In the latter case, flags are restored at the end of the group just as in Perl.
- Perl is forgiving about malformed matching constructs, as in the expression `*a`, as well as dangling brackets, as in the expression `abc]`, and treats them as literals. This class also accepts dangling brackets but is strict about dangling meta-characters such as `+`, `?` and `*`.

Related Information:

Comparison between the Java and Perl 5 regular expression syntax

Finding and Replacing Text in Multiple Files

To open the **Find/Replace in Files** dialog box, use the **Find/Replace in Files** action that is available in the following locations::

- The **Find** menu.
- The main toolbar.
- The contextual menu of the ***DITA Maps Manager*** *view*.
- The contextual menu of the ***Project*** *view*.
- The contextual menu of the ***Data Source Explorer*** *view* for most types of database connections.

The operation works on both local and remote files from an (S)FTP, WebDAV or *CMS* server.

It enables you to define *Search for* or *Search for and Replace* operations across a number of files. You can use *Perl-like regular expression syntax* to match patterns in text content. The *replace* operation can bind regular expression capturing groups ($1, $2, etc.) from the find pattern.

Tip: To replace the `tag-name` start tag and its attributes with the `new-tag-name` tag use as **Text to find** the expression *<tag-name(\s+)(.*)>* and as **Replace with** the expression *<new-tag-name$1$2>*.

The encoding used to read and write the files is detected from the XML header or from the BOM. If a file does not have an XML header or BOM Oxygen XML Editor uses by default the UTF-8 encoding for files of type XML, that is for files with one of the extensions: `.xml`, `.xsl`, `.fo`, `.xsd`, `.rng`, `.nvdl`, `.sch`, `.wsdl` or *an extension associated with the XML editor type*. For the other files it uses *the encoding configured for non-XML files.*

You can cancel a long operation at any time by pressing the **Cancel** button of the progress dialog box displayed when the operation is executed.

Since the content of read-only files cannot be modified, the **Replace** operation is not processing those files. For every such file, a warning message is displayed in the message panel.

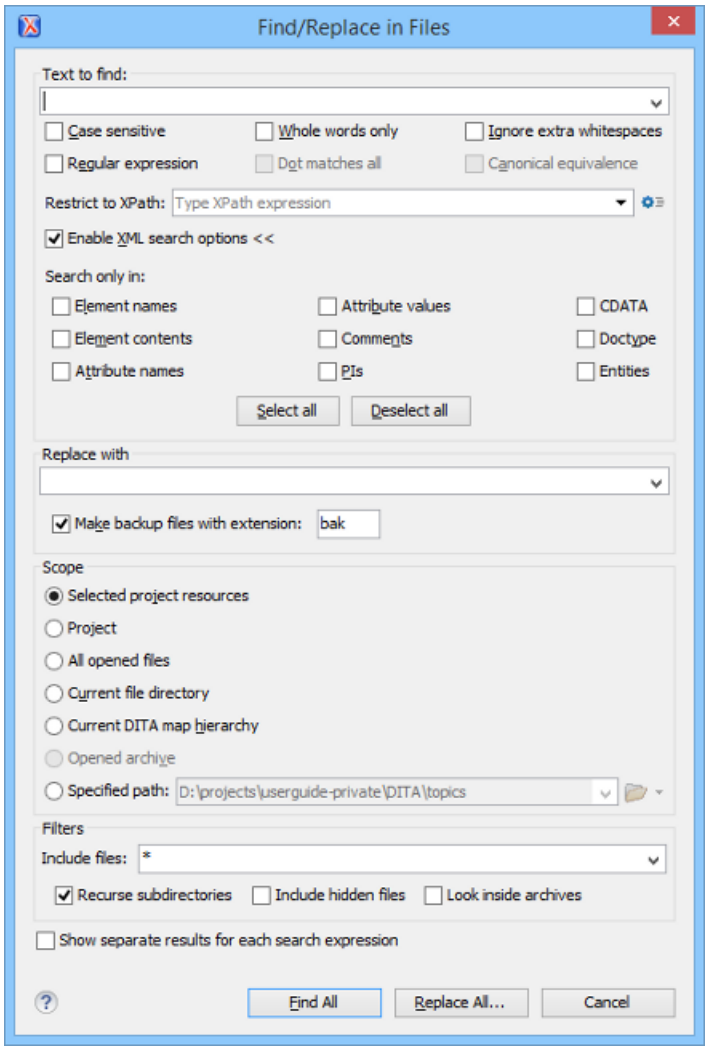


Figure 245: Find / Replace in Files Dialog Box

The dialog box contains the following options:

- **Text to find** - The target character string to search for. You can search for Unicode characters specified in the `\uNNNN` format. Also, hexadecimal notation (`\xNNNN`) and octal notation (`\0NNNN`) can be used. In this case you have to select the **Regular expression** option. For example, to search for a space character you can use the `\u0020` code.
- **Case sensitive** - When selected, the search operation follows the exact letter case of the value entered in the **Text to find** field.
- **Whole words only** - Only entire occurrences of a word are included in the search operation.
- **Ignore extra whitespaces** - If selected, the application normalizes the content (collapses any sequence of whitespace characters into a single space) and trims its leading and trailing whitespaces when performing the search operation.
- **Regular expression** - When this option is selected, you can use regular expressions in *Perl-like regular expressions syntax* to look for specific pieces of text.
  - **Dot matches all** - A dot used in a regular expression also matches end of line characters.
  - **Canonical equivalence** - If selected, two characters will be considered a match if, and only if, their full *canonical* decompositions match. For example, the **ã** symbol can be inserted as a single character or as two characters (the **a** character, followed by the tilde character). This option is not selected by default.
- **XPath** - The XPath 2.0 expression you input in this combo is used for restricting the search scope.

  Note: The *Content Completion Assistant* helps you input XPath expressions, valid in the current context.
- **Enable XML search options** - This option is only available when editing in **Text** mode. It provides access to a set of options that allow you to search specific XML component types:

- **Element names** - Only the element names are included in the search operation that ignores XML-tag notations ('<', '/', '>'), attributes or white-spaces.
- **Element contents** - Search in the text content of XML elements.
- **Attribute names** - Only the attribute names are included in the search operation, without the leading or trailing white-spaces.
- **Attribute values** - Only the attribute values are included in the search operation, without single quotes(') or double quotes(").
- **Comments** - Only the content of comments is included in the search operation, excluding the XML comment delimiters ('<!--', '-->').
- **PIs** (Processing Instructions) - Only the content are searched, skipping '<?', '?>'. e. g.: <?processing instruction?>
- **CDATA** - Searches inside content of CDATA sections.
- **DOCTYPE** - Searches inside content of DOCTYPE sections.
- **Entities** - Only the entity names are searched.

The two buttons **Select All** and **Deselect All** allow a simple activation and deactivation of all types of XML components.

Note: Even if you select all options of the **Enable XML search options** section, the search is still XML-aware. If you want to perform the search over the entire file content, deselect **Enable XML search options**.

- **Replace with** - The character string with which to replace the target. It may contain regular expression group markers if the search expression is a regular expression and the **Regular expression** checkbox is selected.
- **Make backup files with extension** - In the replace process Oxygen XML Editor makes backup files of the modified files. The default extension is `.bak`, but you can change the extension as you prefer.
- **Selected project resources** - Searches only in the selected files of the currently opened project. This option is not displayed when this dialog box is opened from the contextual menu of the ***DITA Maps Manager*** *view* and ***Archive Browser*** *view*.
- **Project files** - Searches in all files from the current project. This option is not displayed when this dialog box is opened from the contextual menu of the ***DITA Maps Manager*** *view* and ***Archive Browser*** *view*.
- **All opened files** - Searches in all files opened in Oxygen XML Editor (regular files or *DITA maps*). You are prompted to save all modified files before any operation is performed. This option is not displayed when this dialog box is started from the contextual menu of the ***DITA Maps Manager*** *view* and ***Archive Browser*** *view*.
- **Current file directory** - The search is done in the directory of the file opened in the current editor panel. If there is no opened file, this option is not available. Also, this option is not displayed when this dialog box is opened from the contextual menu of the ***DITA Maps Manager*** *view* and ***Archive Browser****view*.
- **Current DITA Map hierarchy** - The search is done in all maps and topics referenced by the currently selected *DITA map*, opened in the ***DITA Maps Manager*** *view*.
- **Opened archive** - The search is done in an archive opened in the *Archive Browser* view. Displayed only when this dialog box is opened from the ***Archive Browser*** *view*.
- **Specified path** - Chooses the search path.
- **Include files** - Narrows the scope of the operation only to the files that match the given filters.
- **Recurse subdirectories** - When selected, the search is performed recursively for the specified scope. The one exception is that this option is ignored if the scope is set to **All opened files**.
- **Include hidden files** - When selected, the search is also performed in the hidden files.
- **Include archives** - When selected, the search is also done in all individual file entries from all supported ZIP-type archives.
- **Show separate results for each search expression** - When selected, the application opens a new tab to display the result of each new search expression. When the option is unchecked, the search results are displayed in the *Find in Files* tab, replacing any previous search results.
- **Find All** - Executes a find operation and returns the result list to the message panel. The results are *displayed in a view* that allows grouping the results as a tree with two levels.
- **Replace All** - Replaces all occurrences of the target contained in the specified files.

When you replace a fragment of text, Oxygen XML Editor provides a preview of the changes you make. The **Preview** dialog box is divided in two sections. The first section presents a list of all the documents containing the fragment of text you want to modify. The second section offers a view of the original file and a view of the final

result. It also allows you to highlight all changes using the vertical bar from the right side of the view. The **Next change** and **Previous change** buttons allow you to navigate through the changes displayed in the **Preview** dialog box.

CAUTION: Use this option with caution. Global search and replace across all project files does not open the files containing the targets, nor does it prompt on a per occurrence basis, to confirm that a replace operation must be performed. As the operation simply matches the string defined in the find field, this may result in replacement of matching strings that were not originally intended to be replaced.

Regular Expressions Syntax

Oxygen XML Editor uses the *Java regular expression syntax*. It is **similar** to that used in Perl 5, with several exceptions. Thus, Oxygen XML Editor does not support the following constructs:

- The conditional constructs `(?{X})` and `(?(condition)X|Y)`.
- The embedded code constructs `(?{code})` and `(??{code})`.
- The embedded comment syntax `(?#comment)`.
- The preprocessing operations `\l`, `\u`, `\L`, and `\U`.

When using regular expressions, note that some sets of characters from *XPath/XML Schema/Schematron* are slightly different than the ones used by Oxygen XML Editor/Java in the text searches from the ***Find/Replace** dialog box* and ***Find/Replace in Files** dialog box*. The most common example is with the `\w` and `\W` set of characters. To ensure consistent results between the two, it is recommended that you use the following constructs in the ***Find/Replace** dialog box* and ***Find/Replace in Files** dialog box*:

- **/w** - `[#x0000-#x10FFFF]-[\p{P}\p{Z}\p{C}]` instead of `\w`
- **/W** - `[\p{P}\p{Z}\p{C}]` instead of `\W`

There are some other notable differences that may cause unexpected results, including the following:

- In Perl, `\1` through `\9` are always interpreted as back references. A backslash-escaped number greater than 9 is treated as a back reference if at least that many sub-expressions exist. Otherwise, it is interpreted, if possible, as an octal escape. In this class octal escapes must always begin with a zero. In Java, `\1` through `\9` are always interpreted as back references, and a larger number is accepted as a back reference if at least that many sub-expressions exist at that point in the regular expression. Otherwise, the parser will drop digits until the number is smaller or equal to the existing number of groups or it is one digit.
- Perl uses the g flag to request a match that resumes where the last match left off.
- In Perl, embedded flags at the top level of an expression affect the whole expression. In Java, embedded flags always take effect at the point where they appear, whether they are at the top level or within a group. In the latter case, flags are restored at the end of the group just as in Perl.
- Perl is forgiving about malformed matching constructs, as in the expression `*a`, as well as dangling brackets, as in the expression `abc]`, and treats them as literals. This class also accepts dangling brackets but is strict about dangling meta-characters such as `+`, `?` and `*`.

Related Information:

Comparison between the Java and Perl 5 regular expression syntax

Search and Refactoring Actions for IDs and IDREFS

Oxygen XML Editor allows you to search for ID declarations and references (IDREFS) and to *define the scope of the search and refactor operations*. These operations are available for XML documents that have an associated DTD, XML Schema, or Relax NG schema. These operations are available through the search and refactor actions in the contextual menu. In **Text** mode, these actions are also available in the ***Quick Assist*** menu.

The search and refactor actions from the contextual menu are grouped in the **Manage IDs** section:

Rename in

Renames the ID and all its occurrences. Selecting this action opens the **Rename XML ID** dialog box. This dialog box lets you insert the new ID value and *choose the scope of the rename operation*. For a preview of the changes you are about to make, click **Preview**. This opens the **Preview** dialog box, which presents a list with the files that contain changes and a preview zone of these changes.

Rename in File

Renames the ID you are editing and all its occurrences from the current file.

Note: Available in the **Text** mode only.

Search References

Searches for the references of the ID. By default, the scope of this action is the current project. If you configure a scope using the ***Select the scope for the Search and Refactor operations*** dialog box, this scope will be used instead.

Search References in

Searches for the references of the ID. Selecting this action opens the ***Select the scope for the Search and Refactor operations***.

Search Declarations

Searches for the declaration of the ID reference. By default, the scope of this action is the current project. If you configure a scope using the ***Select the scope for the Search and Refactor operations*** dialog box, this scope will be used instead.

Note: Available in the **Text** mode only.

Search Declarations in

Searches for the declaration of the ID reference. Selecting this action opens the ***Select the scope for the Search and Refactor operations***.

Note: Available in the **Text** mode only.

Search Occurrences in file

Searches for the declaration and references of the ID in the current document.

Tip: A quick way to go to the declaration of an ID in **Text** mode is to move the cursor over an ID reference and use the **Ctrl + Single-Click (Command + Single-Click on OS X)** navigation.

Selecting an ID that you use for search or refactor operations differs between the **Text** and **Author** modes. In the **Text** mode, you position the cursor inside the declaration or reference of an ID. In the **Author** mode, Oxygen XML Editor collects all the IDs by analyzing each element from the path to the root. If more IDs are available, you are prompted to choose one of them.

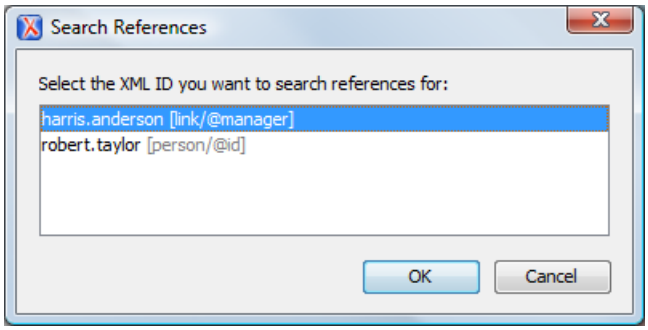


Figure 246: Selecting an ID in the Author Mode

Related Information:

Working with Modular XML Files in the Master Files Context

Search and Refactor Operations Scope

The *scope* is a collection of documents that define the context of a search and refactor operation. To control it you can use the **Change scope** operation, available in the *Quick Assist* action set or on the **Resource Hierarchy/ Dependency View** toolbar. You can restrict the scope to the current project or to one or multiple *working sets*. The **Use only Master Files, if enabled** checkbox allows you to restrict the scope of the search and refactor operations to the resources from the **Master Files** directory. Click **read more** for details about the *Master Files support*.

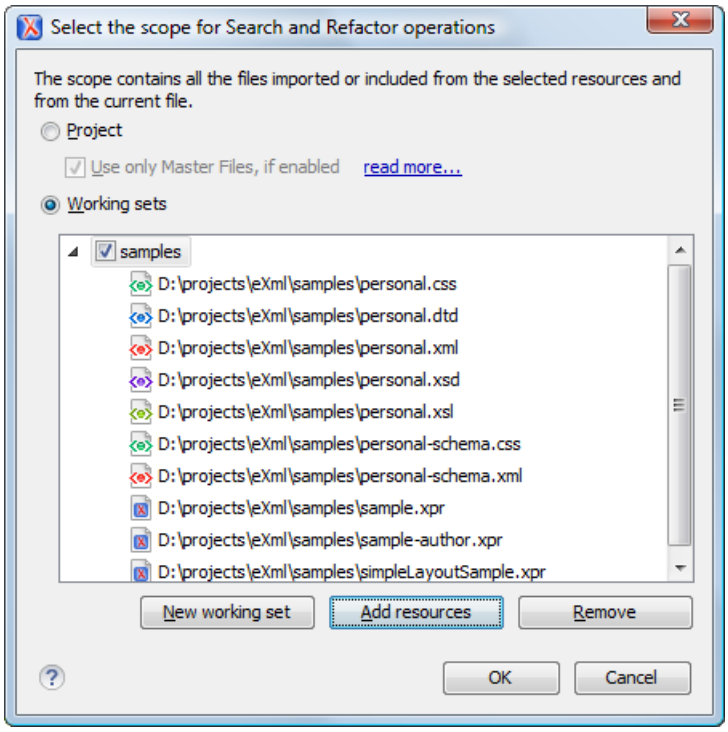


Figure 247: Change Scope Dialog Box

The scope you define is applied to all future search and refactor operations until you modify it. Contextual menu actions allow you to add or delete files, folders, and other resources to the *working set* structure.

Working with Modular XML Files in the Master Files Context

Smaller interrelated modules that define a complex XML modular structure cannot be correctly edited or validated individually, due to their interdependency with other modules. Oxygen XML Editor provides the support for defining the main module (or modules), allowing you to edit any file from the hierarchy in the context of the *master files*.

You cat set a main XML document either using the *master files support from the **Project** view*, or using a validation scenario.

To set a *master file* using a validation scenario, add validation units that point to the main modules. Oxygen XML Editor warns you if the current module is not part of the dependencies graph computed for the main XML document. In this case, it considers the current module as the main XML document.

The advantages of working with modular XML files in the context of a *master file* include:

- Correct validation of a module in the context of a larger XML structure.
- *Content Completion Assistant* displays all collected entities and IDs starting from the *master files*.
- Oxygen XML Editor uses the schema defined in the *master file* when you edit a module that is included in the hierarchy through the *External Entity* mechanism.
- The *master files* defined for the current module determines the *scope of the search and refactoring actions* for ID/IDREFS values and for updating references when renaming/moving a resource. Oxygen XML Editor performs the search and refactoring actions in the context that the *master files* determine, improving the speed of execution.

To watch our video demonstration about editing modular XML files in the *master files* context, go to *https://www.oxygenxml.com/demo/Working_With_XML_Modules.html*.

Related Information:

Master Files Support on page 286

XML Resource Hierarchy/Dependencies View on page 482

XML Resource Hierarchy/Dependencies View

The **Resource Hierarchy/Dependencies** view allows you to easily see the hierarchy / dependencies for an XML document. The tree structure presented in this view is built based on the *XIinclude* and *External Entity* mechanisms. To define the scope for calculating the dependencies of a resource, click ***Configure dependencies search scope*** on the **Resource Hierarchy/Dependencies** toolbar.

To open this view, go to **Window** > **Show View** > **Resource Hierarchy/Dependencies** . As an alternative, right-click the current document and either select **Resource Hierarchy** or **Resource Dependencies**.

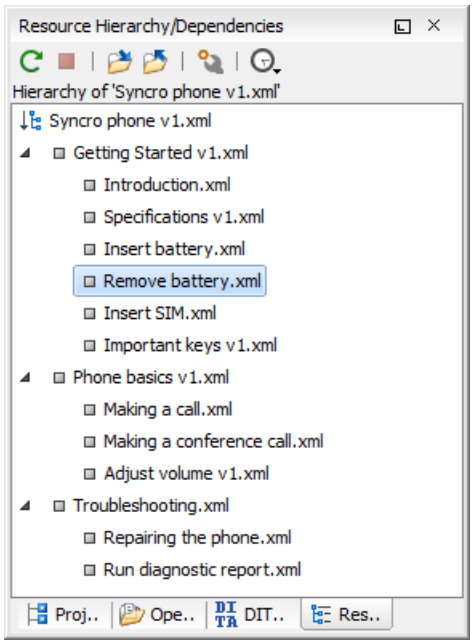


Figure 248: Resource Hierarchy/Dependencies View - Hierarchy for `Syncro phone v1.xml`

The build process for the dependencies view is started with the **Resource Dependencies** action available on the contextual menu.

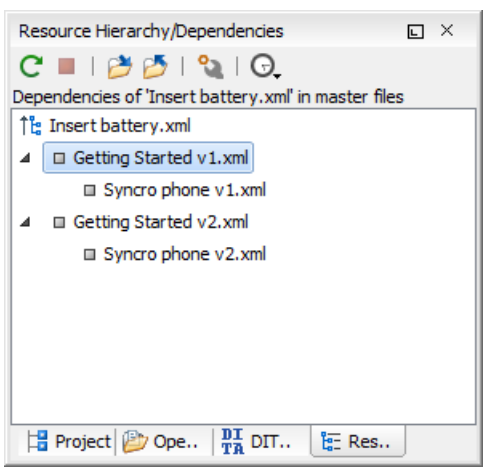


Figure 249: Resource Hierarchy/Dependencies View - Dependencies for `Insert battery.xml`

The following actions are available in the **Resource Hierarchy/Dependencies** view:

Refresh

Refreshes the Hierarchy/Dependencies structure.

Stop

Stops the hierarchy/dependencies computing.

Show Hierarchy

Allows you to choose a resource to compute the hierarchy structure.

Show Dependencies

Allows you to choose a resource to compute the dependencies structure.

Configure

Allows you to configure a scope to compute the dependencies structure. There is also an option for automatically using the defined scope for future operations.

History

Provides access to the list of previously computed dependencies. Use the **Clear history** button to remove all items from this list.

The contextual menu contains the following actions:

Open

Opens the resource. You can also double-click a resource in the Hierarchy/Dependencies structure to open it.

Copy location

Copies the location of the resource.

Move resource

Moves the selected resource.

Rename resource

Renames the selected resource.

Show Resource Hierarchy

Shows the hierarchy for the selected resource.

Show Resource Dependencies

Shows the dependencies for the selected resource.

Add to Master Files

Adds the currently selected resource in *the **Master Files** directory*.

Expand All

Expands all the children of the selected resource from the Hierarchy/Dependencies structure.

Collapse All

Collapses all children of the selected resource from the Hierarchy/Dependencies structure.

Tip: When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon .

Note: The **Move resource** or **Rename resource** actions give you the option to *update the references to the resource*. Only the references made through the *XInclude* and *External Entity* mechanisms are handled.

Related Information:

Working with Modular XML Files in the Master Files Context on page 481
Search and Refactor Operations Scope on page 480

Moving/Renaming XML Resources

When you select the **Rename** action in the contextual menu of the **Resource/Hierarchy Dependencies** view, the **Rename resource** dialog box is displayed. The following fields are available:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references** - Select this option to update the references to the resource you are renaming.

When you select the **Move** action from the contextual menu of the **Resource/Hierarchy Dependencies** view, the **Move resource** dialog box is displayed. The following fields are available:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.

- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, in accordance with the new location and name.

If the **Update references of the moved resource(s)** option is selected, a **Preview** option (which opens the **Preview** dialog box) is available for both actions. The **Preview** dialog box presents a list with the resources that are updated.

Working with XML Catalogs

Oxygen XML Editor uses *XML Catalogs* to resolve references for validations and transformations and they are especially helpful for resolving external resources when internet access is not available or your connection is slow.

Oxygen XML Editor supports any *XML Catalog* file that conforms to one of the following:

1. *OASIS XML Catalogs Committee Specification v1.1*.
2. *OASIS Technical Resolution 9401:1997*, including the plain-text flavor described in that resolution.

The version 1.1 of the OASIS *XML Catalog* specification introduces the possibility to map a system ID, public ID, or a URI to a local copy using only a suffix of the ID or URI used in the actual document. This is done using the catalog elements `systemSuffix` and `uriSuffix`.

Depending on the resource type, Oxygen XML Editor uses different catalog mappings.

Table 10: Catalog Mappings

Document	Referenced Resource	Mappings
XML	DTD	*system* or *public* The ***Prefer** option* controls which one of the mappings should be used.
	XML Schema	The following strategy is used (if one step fails to provide a resource, the next is applied): **1.** resolve the schema using *URI* catalog mappings. **2.** resolve the schema using *system* catalog mappings. This happens only if the ***Resolve schema locations also through system mappings** option* is selected (it is by default). **3.** resolve the root *namespace* using *URI* catalog mappings.
	Relax NG	
	Schematron	
	NVDL	
XSL	XSL/ANY	*URI*
CSS	CSS	*URI*
XML Schema	XML Schema	The following strategy is used (if one step fails to provide a resource, the next is applied): **1.** resolve schema reference using *URI* catalog mappings. **2.** resolve schema reference using *system* catalog mappings. This happens only if the ***Resolve schema locations also through system mappings** option* is selected (it is by default). **3.** resolve schema *namespace* using *URI* catalog mappings. This happens only if the ***Process namespaces through URI mappings for XML Schema** option* is selected (it is not by default).
Relax NG	Relax NG	

Creating an XML Catalog with a Template

An *XML Catalog* file can be created quickly in Oxygen XML Editor starting from the two document templates called *OASIS XML Catalog 1.0* and *OASIS XML Catalog 1.1*. They are available when *creating new document templates*.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE catalog
      PUBLIC "-//OASIS//DTD XML Catalogs V1.1//EN"
      "http://www.oasis-open.org/committees/entity/release/1.1/catalog.dtd">

<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">

    <!-- Use "system" and "public" mappings when resolving DTDs -->
    <system
          systemId="http://www.docbook.org/xml/4.4/docbookx.dtd"
          uri="frameworks/docbook/4.4/dtd/docbookx.dtd"/>

    <!-- "systemSuffix" matches any system ID ending in a specified string -->
    <systemSuffix
          systemIdSuffix="docbookx.dtd"
          uri="frameworks/docbook/dtd/docbookx.dtd"/>

    <!-- Use "uri" for resolving XML Schema and XSLT stylesheets -->
    <uri
          name="http://www.oasis-open.org/docbook/xml/5.0/rng/docbook.rng"
          uri="frameworks/docbook/5.0/rng/docbookxi.rng"/>

    <!-- The "uriSuffix" matches any URI ending in a specified string -->
    <uriSuffix
          uriSuffix="docbook.xsl"
          uri="frameworks/docbook/xsl/fo/docbook.xsl"/>

</catalog>
```

How Oxygen XML Editor Determines which Catalog to Use

Oxygen XML Editor uses *XML Catalogs* to resolve references for validations and transformations and it maps such references to the built-in local copies of the schemas associated with the various *frameworks* (DocBook, DITA, TEI, XHTML, SVG, etc.)

Oxygen XML Editor includes default global catalogs as well as default catalogs for each of the built-in *frameworks*, and you can also create your own.

Oxygen XML Editor looks for catalogs in the following order:

- Global user-defined catalogs that are set in the ***XML Catalog*** *preferences page*.
- User-defined catalogs that are set for each *frameworks* in the ***Catalog*** *tab* of the ***Document Type*** *configuration dialog box*.
- Default built-in catalogs.

Example:

An *XML Catalog* can be used to map a W3C XML Schema specified with an URN in the `xsi:noNamespaceSchemaLocation` attribute of an XML document to a local copy of the schema.

Considering the following XML document code snippet:

```
<topic xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="urn:oasis:names:tc:dita:xsd:topic.xsd:1.1">
```

The URN can be resolved to a local schema file with a catalog entry like this:

```
<uri name="urn:oasis:names:tc:dita:xsd:topic.xsd:1.1"
    uri="topic.xsd"/>
```

Related Information:

XML Catalog Preferences on page 113

Resolving Schema Locations Through XML Catalogs

Schema locations can be mapped using an *XML Catalog*. Oxygen XML Editor resolves the location of a schema in the following order:

- First, it attempts to resolve the schema location as a URI (`uri`, `uriSuffix`, `rewriteUri`, `delegateUri` mappings from the *XML Catalog*). If this succeeds, the process end here.
- If the ***Resolve schema locations also through system mappings*** *option* is selected, it attempts to resolve the schema location as a system ID (`system`, `systemSuffix`, `rewriteSuffix`, `rerwriteSystem` from the *XML Catalog*). If this succeeds, the process ends here.
- If the ***Process "schemaLocation" namespaces through URI mappings for XML Schema*** *option* is selected, the target namespace of the imported XML Schema is resolved through URI mappings. If the schema specified in the *schemaLocation* attribute is not resolved successfully, the namespace of the root element is taken into account. If this succeeds, the process ends here.
- If none of these succeeds, the actual *schema location* is used.

Related Information:

Working with XML Catalogs on page 484

Editing Large XML Documents with DTD Entities or XInclude

Consider the case of documenting a large project. It is likely that there will be several people involved. The resulting document can be few megabytes in size. The question becomes how to deal with this amount of data in such a way that work parallelism will not be affected.

Fortunately, XML provides two solutions for this: **DTD Entities** and **XInclude**. A master document can be created, with references to the other document parts, containing the document sections. The users can edit the documents individually, then apply an XSLT stylesheet over the master and obtain the output files in various formats (for example, PDF or HTML).

Including Document Parts with DTD Entities

There are two conditions for including a part using DTD entities:

- The master document should declare the DTD to be used, while the external entities should declare the XML sections to be referenced.
- The document containing the section must not define again the DTD.

A master document looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "../xml/docbookx.dtd" [
<!ENTITY testing SYSTEM "testing.xml" > ]
>
<book>
<chapter> ...
```

The referenced document looks like this:

```
<section> ... here comes the section content ... </section>
```

Note:

The indicated DTD and the element names (*section*, *chapter*) are used here only for illustrating the inclusion mechanism. You can use any DTD and element names you need.

At a certain point in the master document there can be inserted the section *testing.xml* entity:

```
... &testing; ...
```

When splitting a large document and including the separate parts in the *master file* using external entities, only the *master file* will contain the Document Type Definition (DTD) or other type of schema. The included sections can not define the schema again because the main document will not be valid. If you want to validate the parts separately you have to *use XInclude* for assembling the parts together with the *master file*.

Including Document Parts with XInclude

XInclude is a standard for assembling XML instances into another XML document through inclusion. It enables larger documents to be dynamically created from smaller XML documents without having to physically duplicate the content of the smaller files in the *master file*. XInclude is targeted as the replacement for External Entities. The advantage of using XInclude is that, unlike the entities method, each of the assembled documents is

permitted to contain a Document Type Declaration (DOCTYPE). This means that each file is a valid XML instance and can be independently validated. It also means that the main document, which includes smaller instances, can be validated without having to remove or comment out the DOCTYPE. as is the case with External Entities. This makes XInclude a more convenient and effective method for managing XML instances that need to be stand-alone documents and part of a much larger project.

The main application for XInclude is in the document-oriented content such as manuals and Web pages. Employing XInclude enables you to manage content in a modular fashion that is akin to Object Oriented methods used in languages such as Java, C++ or C#.

The advantages of modular documentation include: reusable content units, smaller file units that are easier to be edited, better version control and distributed authoring.

The XInclude support in Oxygen XML Editor is enabled by default. It is controlled by the ***Enable XInclude processing*** *option* in the ***XML > XML Parser*** *preferences page*. When enabled, Oxygen XML Editor will be able to validate and transform documents comprised of parts added using XInclude.

Example: Using XInclude to Include a Chapter in an Article

Chapter file (`introduction.xml`) looks like this:

```
<?xml version="1.0"?>
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook XML V4.3//EN"
"http://www.oasis-open.org/docbook/xml/4.3/docbookx.dtd">
<chapter>
    <title>Getting started</title>
    <section>
        <title>Section title</title>
        <para>Para text</para>
    </section>
</chapter>
```

Main article file looks like this:

```
<?xml version="1.0"?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.3//EN"
"http://www.docbook.org/xml/4.3/docbookx.dtd"
[ <!ENTITY % xinclude SYSTEM "../frameworks/docbook/dtd/xinclude.mod">
%xinclude;
]>
<article>
    <title>Install guide</title>
    <para>This is the install guide.</para>
    <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
                    href="introduction.xml">
      <xi:fallback>
        <para>
          <emphasis>FIXME: MISSING XINCLUDE CONTENT</emphasis>
        </para>
      </xi:fallback>
    </xi:include>
</article>
```

In this example, note the following:

- The DOCTYPE declaration defines an entity that references a file containing the information to add the *xi* namespace to certain elements defined by the DocBook DTD.
- The href attribute of the xi:include element specifies that the `introduction.xml` file will replace the *xi:include* element when the document is parsed.
- If the `introduction.xml` file cannot be found, the parser will use the value of the *xi:fallback* element - a FIXME message.

Example: Include only a Fragment

If you want to include only a fragment of a file in the *master file*, the fragment must be contained in a tag having an `xml:id` attribute and you must use an XPointer expression pointing to the `xml:id` value.

Notice: Oxygen XML Editor supports the *XPointer Framework* and the *XPointer element() Scheme*, but it does NOT support the *XPointer xpointer() Scheme*.

For example, if the *master file* is:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="test.rng" type="application/xml"
        schematypens="http://relaxng.org/ns/structure/1.0"?>
<test>
```

```
    <sup:include href="a.xml" xpointer="a1"
        xmlns:sup="http://www.w3.org/2001/XInclude"/>
</test>
```

and the `a.xml` file is:

```
<?xml version="1.0" encoding="UTF-8"?>
<test>
    <a xml:id="a1">test</a>
</test>
```

after resolving the XPointer reference the document is:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="test.rng" type="application/xml"
          schematypens="http://relaxng.org/ns/structure/1.0"?>
<test>
    <a xml:id="a1" xml:base="a.xml">test</a>
</test>
```

Viewing Status Information

Status information generated by operations such as *schema detection*, *manual validation*, *automatic validation*, and *transformations* are fed into the **Information** view, allowing you to monitor how the operation is being executed.

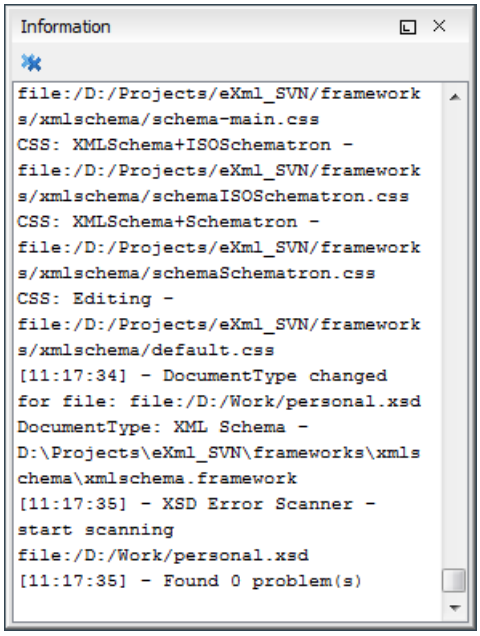


Figure 250: Information view messages

Messages contain a timestamp, the name of the thread that generated it and the actual status information. The number of displayed messages can be controlled with the ***Maximum number of lines*** *option* in the **Views** preference page.

To make the view visible, select **Window** > **Show View** > **Information**.

Making a Persistent Copy of Results

The ***Results*** *view* displays the results from the following operations:

- *document validation*
- *checking the form of documents*
- *XSLT or FO transformation*
- *find all occurrences of a string in a file*
- *find all occurrences of a string in multiple files*
- *applying an XPath expression to the current document*

To make a persistent copy of the ***Results** view*, use one of these actions:

File > Save Results

Displays the **Save Results** dialog box, used to save the result list of the current message tab. The action is also available on the right-click menu of the **Results** panel.

File > Print Results

Displays the **Page Setup** dialog box used to define the page size and orientation properties for printing the result list of the current **Results** panel. The action is also available on the right-click menu of the **Results** panel.

Save Results as XML from the contextual menu

Saves the content of the **Results** panel in an XML file with the format:

```
<Report>
    <Incident>
        <engine>The engine who provide the error.<engine>
        <severity>The severity level<severity>
        <Description>Description of output message.</Description>
        <SystemID>The location of the file linked to the message.</SystemID>
        <Location>
           <start>
                <line>Start line number in file.<line>
                <column>Start column number in file<column>
           </start>
           <end>
                <line>End line number in file.<line>
                <column>End column number in file<column>
           </start>
        </Location>
    </Incident>
</Report>
```

Related Information:

Results View on page 203

Editor Highlights

An *editor highlight* is a text fragment emphasized by a colored background.

Highlights are generated in both **Text** and **Author** mode when the following actions generate results:

- ***Find/Replace in Files***
- ***Find/Replace***
- ***Open/Find Resource***
- **Find All**
- ***Find All Elements***
- ***XPath in Files***
- ***Search References***
- ***Search Declarations***

By default, Oxygen XML Editor uses a different color for each type of highlight (*XPath in Files*, *Find/Replace*, *Search References*, *Search Declarations*, etc.) You can customize these colors and the maximum number of highlights displayed in a document on the ***Editor** preferences page*. The default maximum number of highlights is 10000.

You can navigate the highlights in the current document by using the following methods:

- Clicking the markers from the range ruler, located at the right side of the document.
- Clicking the **Next** and **Previous** buttons ( ) from the bottom of the range ruler.

  Note: When there are multiple types of highlights in the document, the **Next** and **Previous** buttons ( ) navigate through highlights of the same type.
- Clicking the messages displayed in the ***Results** view* at the bottom of the editor.

To remove the highlights, you can do the following:

- Click the **Remove all** button from bottom of the range ruler.

- Close the results tab at the bottom of the editor that contains the output of the action that generated the highlights.
- Click the **Remove all** button on the right side of the ***Results** panel* at the bottom of the editor.

  Note: Use the **Highlight all results in editor** button (on the right side of the **Results** panel) to either display all the highlights or hide them.

Printing a Document

Printing is supported in **Text**, **Author,** and **Grid** modes. The **Print**(**Ctrl + P (Command + P on OS X)**) action that is available from **File** menu displays the **Page Setup** dialog box, used for defining the page size and orientation properties for printing.

A **Print Preview** action is also available in the **File** menu. It allows you to manage the format of the printed document.

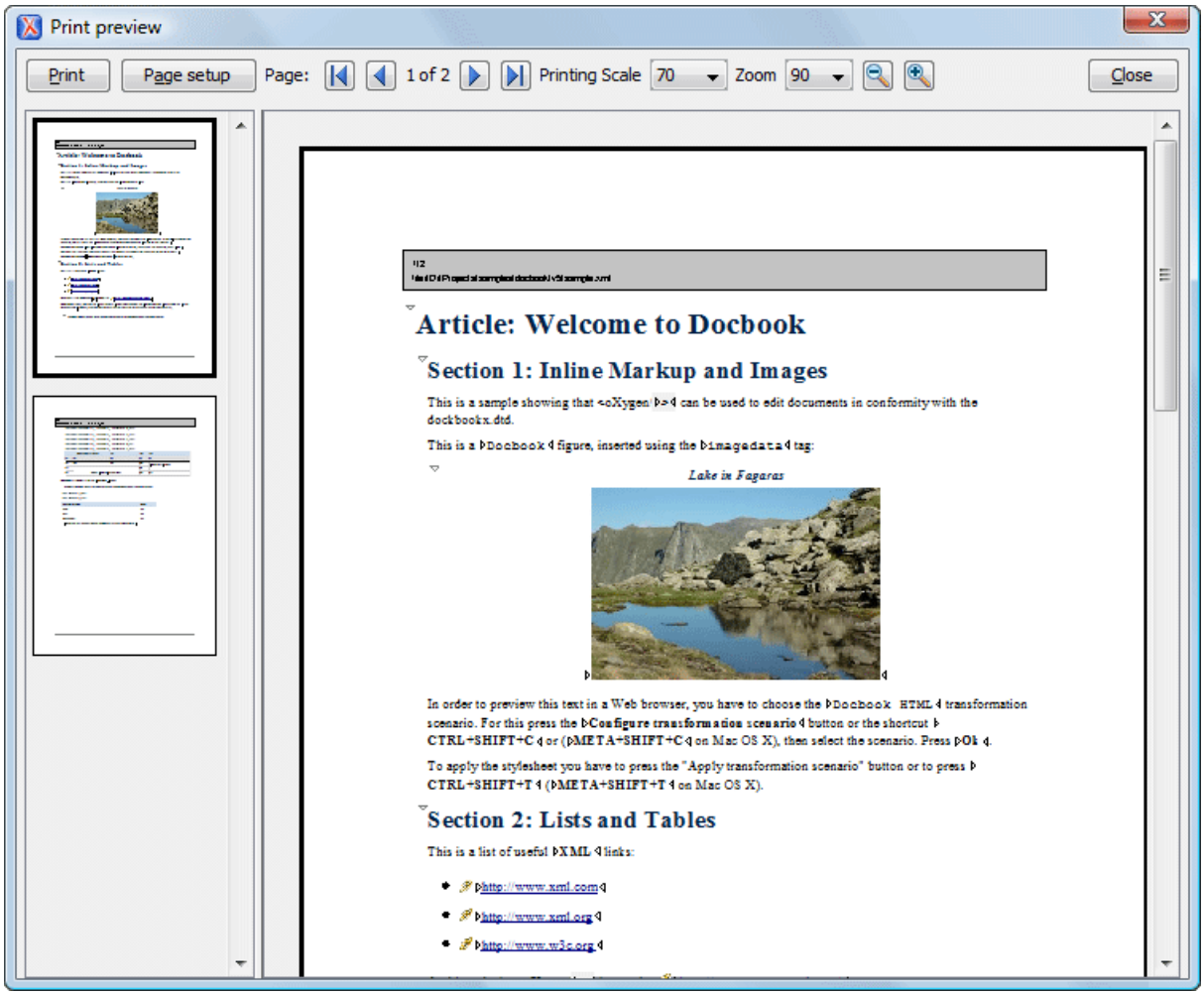


Figure 251: Print Preview Dialog Box

The main window is split in three sections:

- **Preview area** - Displays the formatted document page as it will appear on printed paper.
- **Left stripe** - The left-side stripe that displays a list of thumbnail pages. Clicking any of them displays the page content in the main preview area.
- **Toolbar** - The toolbar area at the top that contains controls for printing, page settings, page navigation, print scaling, and zoom.

Other Printing Features

- If you are printing a document that is opened in the **Author** visual editing mode, you can use the *CSS print media type* to change the styling in the printed output.
- If you are printing a document that is opened in **Author** mode and it contains *Tracked Changes*, you can print (or print preview) a copy of the document as if all changes have been accepted by switching the ***Track Changes Visualization Mode*** to **View Final**.
- If you are printing a document that is opened in **Text** mode and line numbers are displayed (the ***Show line numbers*** *option* is selected), the printed output will include the line numbers.
- If you are printing an XML document that is opened in **Text** mode and the *folding support* is activated (the ***Enable folding*** *option* is selected), the printed output will include the current *folded* state. Note that this applies to printing an entire document and not selections within the document.
- If you are printing an XML document that is opened in **Text** mode and a block of content is selected, you have the ability to print only the selection of text rather than the entire document. When you invoke the print action with a block of content selected in **Text** mode, a dialog box will be presented that offers you the choice to print the selection or the entire document.

Refactoring XML Documents

In the life cycle of XML documents there are instances when the XML structure needs to be changed to accommodate various needs. For example, when an associated schema is updated, an attribute may have been removed, or a new element added to the structure.

These types of situations cannot be resolved with a traditional *Find/Replace* tool, even if the tool accepts regular expressions. The problem becomes even more complicated if an XML document is computed or referenced from multiple modules, since multiple resources need to be changed.

To assist you with these types of refactoring tasks, Oxygen XML Editor includes a specialized **XML Refactoring** tool that helps you manage the structure of your XML documents.

XML Refactoring Tool

The **XML Refactoring** tool is presented in the form of an easy to use wizard that is designed to reduce the time and effort required to perform various structure management tasks. For example, you can insert, delete, or rename an attribute in all instances of a particular element that is found in all documents within your project.

To access the tool, select the **XML Refactoring** action from one of the following locations:

- The **Tools** menu.
- The **Refactoring** submenu from the contextual menu in the ***Project*** *view*.
- The **Refactoring** submenu from the contextual menu in the ***DITA Maps Manager*** *view*.

Note: The predefined refactoring operations are also available from the **Refactoring** submenu in the contextual menu of **Author** or **Text** mode. This is useful because by selecting the operations from the contextual menu, Oxygen XML Editor considers the editing context to skip directly to the wizard page of the appropriate operation and to help you by preconfiguring some of the parameter values. For your convenience, the last 5 operations that were *finished* or *previewed* also appear in the **Refactoring** submenu of the contextual menu in the **Project** view and the **DITA Maps Manager**.

XML Refactoring Wizard

The XML Refactoring tool includes the following wizard pages:

Refactoring operations

The first wizard page presents the available operations, grouped by category. To search for an operation, you can use the filter text box at the top of the page.

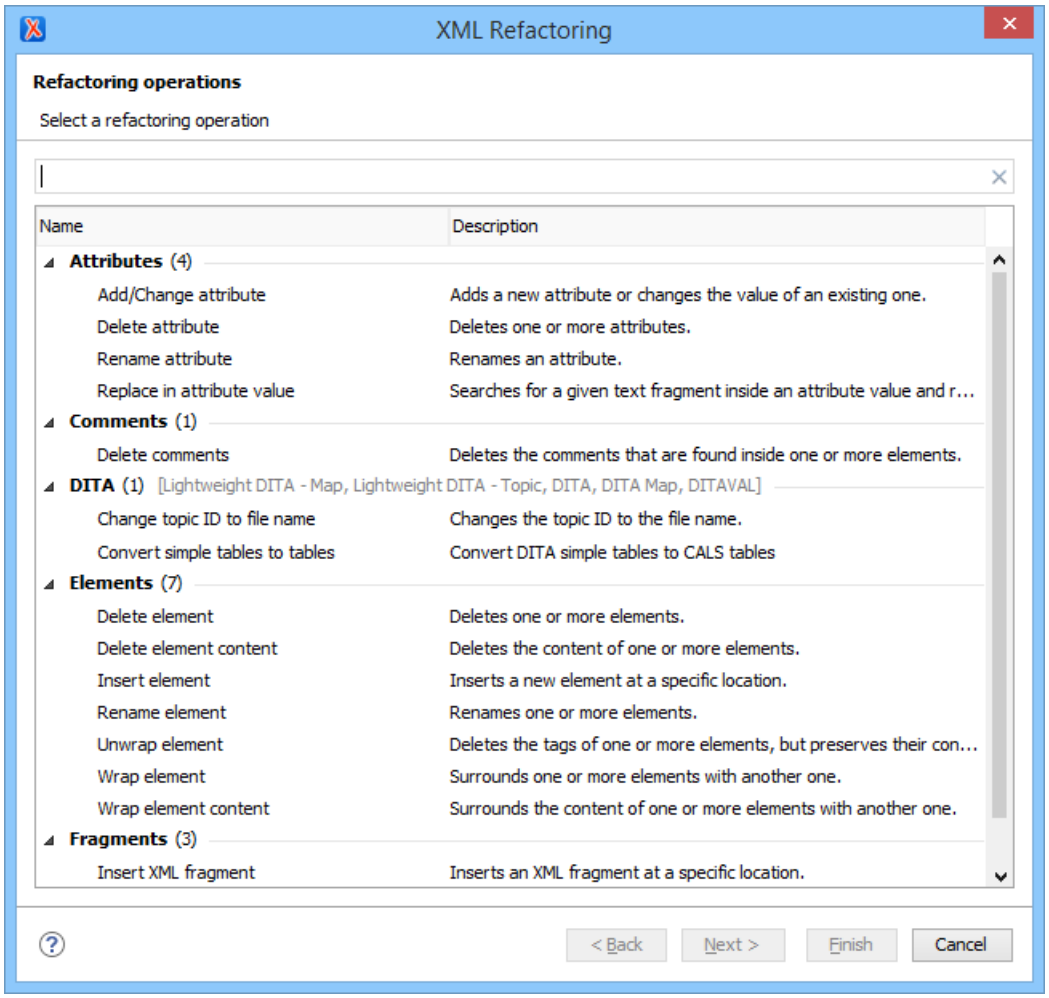


Figure 252: XML Refactoring Wizard

Configure Operation Parameters

The next wizard page allows you to specify the parameters for the refactoring operation. The parameters are specific to the type of refactoring operation that is being performed. For example, to delete an attribute you need to specify the parent element and the qualified name of the attribute to be removed.

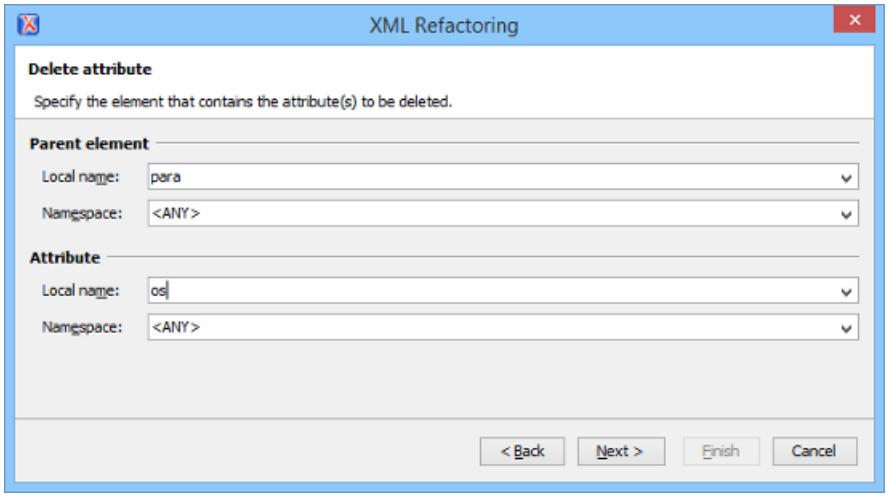


Figure 253: XML Refactoring 2nd Wizard Page (Delete Attribute Operation)

Scope and Filters

The last wizard page allows you to select the set of files that represent the input of the operation.

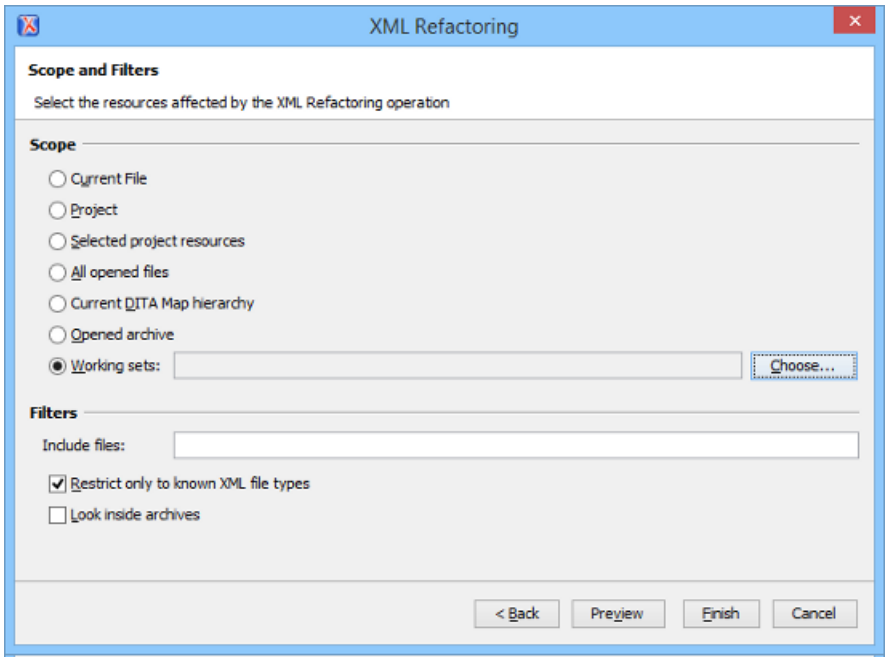


Figure 254: XML Refactoring - Scope and Filters Wizard Page

Scope section

In the **Scope** section, you can select from predefined resource sets (such as the current file, your whole project, the current *DITA map* hierarchy for DITA projects, etc.) or you can define your own set of resources by creating a *working set*.

Filters

The **Filters** section includes the following options:

- **Include files** - Allows you to filter the selected resources by using a file pattern. For example, to restrict the operation to only analyze build files you could use `build*.xml` for the file pattern.
- **Restrict only to known XML file types** - When selected, only resources with a known XML file type will be affected by the operation.
- **Look inside archives** - When selected, the resources inside archives will also be affected.

Preview

You can use the **Preview** button to open a comparison panel where you can review all the changes that will be made by the refactoring operation before applying the changes.

Finish

After clicking the **Finish** button, the operation will be processed and Oxygen XML Editor provides no automatic means for reverting the operations. Any **Undo** action will only revert changes on the current document.

Tip: If an operation takes longer than expected you can use the **Stop** button in the progress bar to cancel the operation.

Predefined Refactoring Operations

The XML Refactoring tool includes a variety of predefined operations that can be used for common refactoring tasks. They are grouped by category in the **Refactoring operations** wizard page. You can also access the operations from the **Refactoring** submenu in the contextual menu of **Author** or **Text** mode. The operations are also grouped by category in this submenu. When selecting the operations from the contextual menu, Oxygen XML Editor considers the editing context to get the names and namespaces of the current element or attribute, and uses this information to preconfigure some of the parameter values for the selected refactoring operation.

Tip: Each operation includes a link in the lower part of the wizard that opens the **XML / XSLT-FO-XQuery / XPath** preferences page where you can configure XPath options and declare namespace prefixes.

The following predefined operations are available:

Refactoring Operations for *Attributes*

Add/Change attribute

Use this operation to change the value of an attribute or insert a new one. This operation allows you to specify the following parameters:

- **Parent element** section
  - **Element** - The parent element of the attribute to be changed, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.
- **Attribute** section
  - **Local name** - The local name of the affected attribute.
  - **Namespace** - The namespace of the affected attribute.
  - **Value** - The value for the affected attribute.
- **Options** section
  - You can choose between one of the following options for the **Operation mode**:
    - **Add the attribute in the parent elements where it is missing**
    - **Change the value in the parent elements where the atrribute already exists**
    - **Both**

Delete attribute

Use this operation to remove one or more attributes. This operation requires you to specify the following parameters:

- **Element** - The parent element of the attribute to be deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.
- **Attribute** - The name of the attribute to be deleted.

Rename attribute

Use this operation to rename an attribute. This operation requires you to specify the following parameters:

- **Element** - The parent element of the attribute to be renamed, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.
- **Attribute** - The name of the attribute to be renamed.
- **New local name** - The new local name of the attribute.

Replace in attribute value

Use this operation to search for a text fragment inside an attribute value and change the fragment to a new value. This operation allows you to specify the following parameters:

- **Target attribute** section
  - **Element** - The parent element of the attribute to be modified, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.
  - **Attribute** - The name of the attribute to be modified.
- **Find / Replace** section
  - **Find** - The text fragments to find. You can use Perl-like regular expressions.
  - **Replace with** - The text fragment to replace the target with. This parameter can bind regular expression capturing groups ($1 , $2, etc.) from the find pattern.

Refactoring Operations for *Comments*

Delete comments

Use this operation to delete comments from one or more elements. This operation requires you specify the following parameter:

- **Element** - The target element (or elements) for which comments will be deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

Note: Comments that are outside the root element will not be deleted because the *serializer* preserves the content before and after the root.

Refactoring Operations for *DITA*

Change topic ID to file name

Use this operation to change the ID of a topic to be the same as its file name.

Convert conrefs to conkeyrefs

Use this operation to convert `conref` attributes to `conkeyref` attributes. For more information and instructions for using this operation, see *Converting Conrefs to Conkeyrefs* on page 1658.

Convert simple tables to CALS tables

Use this operation to convert DITA simple tables to CALS tables.

Convert to Concept

Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept). For more information, see *Converting DITA Topics to Another Type* on page 1624.

Convert to Reference

Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference). For more information, see *Converting DITA Topics to Another Type* on page 1624.

Convert to Task

Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task). For more information, see *Converting DITA Topics to Another Type* on page 1624.

Convert to Topic

Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic). For more information, see *Converting DITA Topics to Another Type* on page 1624.

Convert to Troubleshooting

Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting). For more information, see *Converting DITA Topics to Another Type* on page 1624.

All of these DITA refactoring actions allow you to choose a scope for the operation and some filters:

- **Scope** - Select from a variety of options to define the scope for which resources will be affected by the operation. For example, you can choose to affect all resources in the **Project**, **All opened files**, **Current DITA map hierarchy**, or just the **Current file**.
- **Filters** section
  - **Include files** - Specifies files to be excluded from the operation. You can specify multiple files by separating them with commas and the patterns can include wildcards (such as ***** or **?**).
  - **Restrict to known XML file types only** - Excludes non-XML file types from the operation.
  - **Look inside archives** - If this option is selected, the scope of the operation will include files inside archives.

Refactoring Operations for *Elements*

Delete element

Use this operation to delete elements. This operation requires you to specify the following parameter:

- **Element** - The target element to be deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

Delete element content

Use this operation to delete the content of elements. This operation requires you to specify the following parameter:

- **Element** - The target element whose content is to be deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

Insert element

Use this operation to insert new elements. This operation allows you to specify the following parameters:

- **Element** section
  - **Local name** - The local name of the element to be inserted.
  - **Namespace** - The namespace of the element to be inserted.
- **Location** section
  - **XPath**- An XPath expression that identifies an existing element to which the new element is relative, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.
  - **Position** - The position where the new element will be inserted, in relation to the specified existing element. The possible selections in the drop-down menu are: **After**, **Before**, **First child**, or **Last child**.

Rename element

Use this operation to rename elements. This operation requires you to specify the following parameters:

- **Target elements (XPath)** - The target elements to be renamed, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.
- **New local name** - The new local name of the element.

Unwrap element

Use this operation to remove the surrounding tags of elements, while keeping the content unchanged. This operation requires you to specify the following parameter:

- **Target elements (XPath)** - The target elements whose surrounding tags will be removed, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

Wrap element

Use this operation to surround elements with element tags. This operation allows you to specify the following parameters:

- **Target elements (XPath)** - The target elements to be surrounded with tags, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.
- **Wrapper element** section
  - **Local name** - The local name of the *Wrapper element*.
  - **Namespace** - The namespace of the *Wrapper element*.

Wrap element content

Use this operation to surround the content of elements with element tags. This operation allows you to specify the following parameters:

- **Target elements (XPath)** - The target elements whose content will be surrounded with tags, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.
- **Wrapper element** section
  - **Local name** - The local name of the *Wrapper element* that will surround the content of the target.
  - **Namespace** - The namespace of the *Wrapper element* that will surround the content of the target.

Refactoring Operations for *Fragments*

Insert XML fragment

Use this operation to insert an XML fragment. This operation allows you to specify the following:

- **XML Fragment** - The XML fragment to be inserted.
- **Location** section
  - **XPath** - An XPath expression that identifies an existing element to which the inserted fragment is relative, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

- **Position** - The position where the fragment will be inserted, in relation to the specified existing element. The possible selections in the drop-down menu are: **After**, **Before**, **First child**, or **Last child**.

Replace element content with XML fragment

Use this operation to replace the content of elements with an XML fragment. This operation allows you to specify the following parameters:

- **Target elements (XPath)** - The target elements whose content will be replaced, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.
- **XML Fragment** - The XML fragment with which to replace the content of the target element.

Replace element with XML fragment

Use this operation to replace elements with an XML fragment. This operation allows you to specify the following parameters:

- **Target elements (XPath)** - The target elements to be replaced, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.
- **XML Fragment** - The XML fragment with which to replace the target element.

Refactoring Operations for *JATSKit*

Add BITS DOCTYPE - NLM/NCBI Book Interchange 2.0

Use this operation to add an NLM 'BITS' 2.0 DOCTYPE declaration.

Add Blue DOCTYPE - NISO JATS Publishing 1.1

Use this operation to add a JATS 'Blue' 1.1 DOCTYPE declaration.

Normalize IDs

Use this operation to normalize assigned IDs and assigned IDs to elements that are missing them.

All of these JATSKit refactoring actions allow you to choose a scope for the operation and some filters:

- **Scope** - Select from a variety of options to define the scope for which resources will be affected by the operation. For example, you can choose to affect all resources in the **Project**, **All opened files**, or just the **Current file**.
- **Filters** section
  - **Include files** - Specifies files to be excluded from the operation. You can specify multiple files by separating them with commas and the patterns can include wildcards (such as ***** or **?**).
  - **Restrict to known XML file types only** - Excludes non-XML file types from the operation.
  - **Look inside archives** - If this option is selected, the scope of the operation will include files inside archives.

Additional Notes

Note: There are some operations that allow `<ANY>` for the **local name** and **namespace** parameters. This value can be used to select an element or attribute regardless of its local name or namespace. Also, the `<NO_NAMESPACE>` value can be used to select nodes that do not belong to a namespace.

Note: Some operations have parameters that accept XPath expressions to match elements or attributes. In these XPath expressions you can only use the prefixes declared in the ***Options > Preferences > XML > XSLT-FO-XQUERY > XPath*** page. This preferences page can be easily opened by clicking the link in the note (**Each prefix used in an XPath expression must be declared in the Default prefix-namespace mappings section**) at the bottom of the **Configure Operation Parameters** wizard page.

Custom Refactoring Operations

While Oxygen XML Editor includes a variety of predefined XML refactoring operations to help you accomplish particular tasks, you can also create custom operations according to your specific needs. For example, you could create a custom refactoring operation to convert an attribute to an element and insert the element as the first child of the parent element.

An XML Refactoring operation is defined as a pair of resources:

- An *XQuery Update script* or *XSLT stylesheet* that Oxygen XML Editor will run to refactor the XML files.

- An *XML Operation Descriptor* file that contains information about the operation (such as the name, description, and parameters).

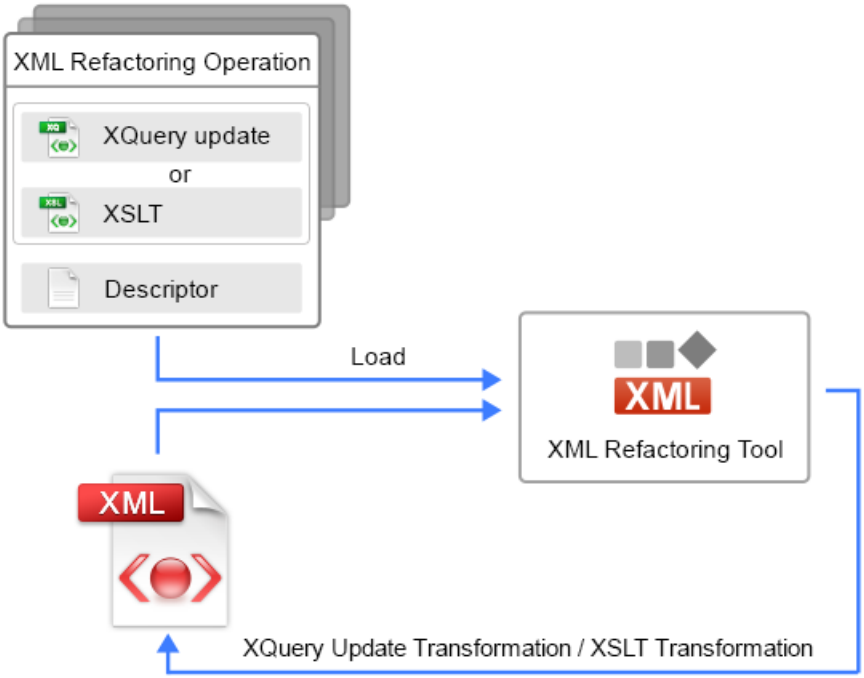


Figure 255: Diagram of an XML Refactoring Operation

All the defined custom operations are loaded by the **XML Refactoring Tool** and presented in *the **Refactoring Operations** wizard page*, along with the predefined built-in operations.

After the user chooses an operation and specifies its parameters, Oxygen XML Editor processes an XQuery Update or XSLT transformation over the input file. This transformation is executed in a **safe mode**, which implies the following:

- When loading the document:
  - The **XInclude** mechanism is disabled. This means that the resources included by using XInclude will not be visible in the transformation.
  - The DTD entities will be processed without being expanded.
  - The associated DTD will be not loaded, so the default attributes declared in the DTD will not be visible in the transformation.
- When saving the updated XML document:
  - The `DOCTYPE` will be preserved.

    Note: This can be changed using *Saxon extension functions in XSLT*.
  - The DTD entities will be preserved as they are in the original document when the document is saved.
  - The attribute values will be kept in their original form without being normalized.
  - The spaces between attributes are preserved. Basically, the spaces are lost by a regular XML serialization since they are not considered important.

The result of this transformation overrides the initial input file.

Note: To achieve some of the previous goals, the XML Refactoring mechanism adds several attributes that are interpreted internally. The attributes belong to the `http://www.oxygenxml.com/ns/xmlRefactoring/additional_attributes` namespace. These attributes should not be taken into account when processing the input XML document since they are discarded when the transformed document is serialized.

Restriction: *Comments* or *processing instructions* that are in any node before or after the root element cannot be modified by an XML Refactoring operation. In other words, XML Refactoring operations can only be applied on the

root element and the nodes inside it. However, as a work around to this limitation, you can use *Saxon extension functions and the XSLT stylesheet method* to implement the new custom XML refactoring operation.

Creating a Custom Refactoring Operation

To create a custom refactoring operation, follow these steps:

1. Create an *XQuery Update script* or *XSLT stylesheet file*.
2. Create an **XML Refactoring Operation Descriptor** file contains the path to the *XQuery Update script* or *XSLT stylesheet*.
3. Store both files in *one of the locations that Oxygen XML Editor* scans when loading the custom operations.

   Result: Once you run the **XML Refactoring** tool again, the custom operation appears in *the **Refactoring Operations** wizard page*.

Related Information:

Storing and Sharing Refactoring Operations on page 508

Custom Refactoring Script

The first step in creating a custom refactoring operation is to create an *XQuery Update script* or *XSLT stylesheet* that is needed to process the refactoring operations. The easiest way to create this script file is to use the **New** document wizard to create a new **XQuery** or **XSLT** file and you can use our *XQuery method example* or *XSLT method example* to help you with the content.

There are cases when it is necessary to add parameters in the *XQuery script* or *XSLT stylesheet*. For instance, if you want to rename an element, you may want to declare an external parameter associated with the name of the element to be renamed. To allow you to specify the value for these parameters, they need to be declared in the *refactoring operation descriptor file* that is associated with this operation.

Note: The XQuery Update processing is disabled by default in Oxygen XML Editor. Thus, if you want to create or edit an XQuery Update script you have to enable this mechanism by creating an *XQuery transformation scenario* and choose **Saxon EE** as the transformation engine. Also, you need to make sure the ***Enable XQuery update** option is selected in the Saxon processor advanced options*.

Note: If you are using an XSLT file, XPath expressions that are passed as parameters will automatically be rewritten to conform with the mapping of the namespace prefixes declared in the ***XML /XSLT-FO-XQuery / XPath** preferences page*.

The next step in creating a custom refactoring operation is to create an **XML Refactoring Operation Descriptor** file contains the path to the *XQuery Update script* or *XSLT stylesheet*.

Related Information:

XQuery Update Script for Creating a Custom Operation on page 502

XSLT Stylesheet for Creating a Custom Operation on page 504

Custom Refactoring Operation Descriptor File

The second step in creating a custom refactoring operation is to create an operation descriptor file. The easiest way to do this is to use the **New** document wizard and choose the **XML Refactoring Operation Descriptor** template.

Introduction to the Descriptor File

This file contains information (such as `name`, `description`, and `id`) that is necessarily when loading an XML Refactoring operation . It also contains the path to the *XQuery Update script* or *XSLT stylesheet* that is associated with the particular operation through the `script` element.

You can specify a `category` for your custom operations to logically group certain operations. The `category` element is optional and if it is not included in the descriptor file, the default name of the category for the custom operations is *Other operations*.

The descriptor file is edited and validated against the following schema: `frameworks/xml_refactoring/operation_descriptor.xsd`.

Declaring Parameters in the Descriptor File

If the XQuery Update script or XSLT stylesheet includes parameters, they should be declared in the **parameters** section of the descriptor file. All the parameters specified in this section of the descriptor file will be displayed in the **XML Refactoring** tool within *the **Configure Operation Parameters** wizard page* for that particular operation.

The value of the first `description` element in the **parameters** section will be displayed at the top of *the **Configure Operation Parameters** wizard page*.

To declare a parameter, specify the following information:

- **`label`** - This value is displayed in the user interface for the parameter.
- **`name`** - The parameter name used in the XQuery Update script or XSLT stylesheet and it should be the same as the one declared in the script.
- **`type`** - Defines the type of the parameter and how it will be rendered. There are several types available:
  - `TEXT` - Generic type used to specify a simple text fragment.
  - `XPATH` - Type of parameter whose value is an XPATH expression. For this type of parameter, Oxygen XML Editor will use a text input with corresponding content completion and syntax highlighting.

    Note: The value of this parameter is transferred as plain text to the XQuery Update or XSLT transformation without being evaluated. You should evaluate the XPath expression inside the XQuery Update script or XSLT stylesheet. For example, you could use the `saxon:evaluate` Saxon extension function.

    Note: A relative XPath expression is converted to an absolute XPath expression by adding `//` before it (`//XPathExp`). This conversion is done before transferring the XPath expression to the XML refactoring engine.

    Note: When writing XPath expressions, you can only use prefixes declared in the ***Options* > *Preferences* > *XML* > *XSLT-FO-XQUERY* > *XPath*** options page.
  - `NAMESPACE` - Used for editing namespace values.
  - `REG_EXP_FIND` - Used when you want to match a certain text by using Perl-like regular expressions.
  - `REG_EXP_REPLACE` - Used along with `REG_EXP_FIND` to specify the replacement string.
  - `XML_FRAGMENT` - This type is used when you want to specify an XML fragment. For this type, Oxygen XML Editor will display a text area specialized for inserting XML documents.
  - `NC_NAME` - The parameter for `NC_NAME` values. It is useful when you want to specify the local part of a *QName* for an element or attribute.
  - `BOOLEAN` - Used to edit boolean parameters.
  - `TEXT_CHOICE` - It is useful for parameters whose value should be from a list of possible values. Oxygen XML Editor renders each possible value as a radio button option.
- **`description`** - The description of the parameter. It is used by the application to display a tooltip when you hover over the parameter.
- **`possibleValues`** - Contains the list with possible values for the parameter and you can specify the default value, as in the following example:

```
<possibleValues onlyPossibleValuesAllowed="true">
    <value name="before">Before</value>
    <value name="after"default="true">After</value>
    <value name="firstChild">First child</value>
    <value name="lastChild">Last child</value>
</possibleValues>
```

Specialized Parameters to Match Elements or Attributes

If you want to match elements or attributes, you can use some specialized parameters, in which case Oxygen XML Editor will propose all declared elements or attributes based on the schema associated with the currently edited file. The following specialized parameters are supported:

`elementLocation`

This parameter is used to match elements. For this type of parameter, the application displays a text field where you can enter the element name or an XPath expression. The text from the `label` attribute is displayed in the application as the label of the text field. The `name` attribute is used to specify the name of the parameter from the XQuery Update script or XSLT stylesheet. If the value of the `useCurrentContext`

attribute is set to `true`, the element name from the cursor position is used as proposed values for this parameter.

Example of an `elementLocation`:

```
<elementLocation name="elem_loc" useCurrentContext="false">
                <element label="Element location">
                    <description>Element location description.</description>
                </element>
</ ElementLocation>
```

`attributeLocation`

This parameter is used to match attributes. For this type of parameter, the application displays two text fields where you can enter the parent element name and the attribute name (both text fields accept XPath expressions for a finer match). The text from the `label` attributes is displayed in the application as the label of the associated text fields. The `name` attribute is used to specify the name of the parameter from the XQuery Update script or XSLT stylesheet. The value of this parameter is an XPath expression that is computed by using the values of the expression from the `element` and `attribute` text fields. For example, if `section` is entered for the element and a `title` is entered for the attribute, the XPath expression would be computed as `//section/@title`. If the value of the `useCurrentContext` attribute is set to `true`, the element and attribute name from the cursor position is used as proposed values for the operation parameters.

Example of an `attributeLocation`:

```
<attributeLocation name="attr_xpath" useCurrentContext="true">
                <element label="Element path">
                    <description>Element path description.</description>
                </element>
                <attribute label="Attribute" >
                    <description>Attribute path description.</description>
                </attribute>
</ AttributeLocation>
```

`elementParameter`

This parameter is used to specify elements by local name and namespace. For this type of parameter, the application displays two combo boxes with elements and namespaces collected from the associated schema of the currently edited file. The text from the `label` attribute is displayed in the application as label of the associated combo. The `name` attribute is used to specify the name of the parameter from the XQuery Update script or XSLT stylesheet. If you specify the `allowsAny` attribute, the application will propose *<ANY>* as a possible value for the **Name** and **Namespace** combo boxes. You can also use the `useCurrentContext` attribute and if its value is set to `true`, the element name and namespace from the cursor position is used as proposed values for the operation parameters.

Example of an `elementParameter`:

```
<elementParameter id="elemID">
    <localName label="Name" name="element_localName" allowsAny="true"
 useCurrentContext="true">
        <description>Local name of the parent element.</description>
    </localName>
    <namespace label="Namespace" name="element_namespace" allowsAny="true">
        <description>Local name of the parent element</description>
    </namespace>
</elementParameter>
```

`attributeParameter`

This parameter is used to specify attributes by local name and namespace. For this type of parameter, the application displays two combo boxes with attributes and their namespaces collected from the associated schema of the currently edited file. The text from the `label` attribute is displayed in the application as the label of the associated combo box. You can also use the `useCurrentContext` attribute and if its value is set to `true`, the attribute name and namespace from the cursor position is used as proposed values for the operation parameters.

Note: An `attributeParameter` is dependant upon an `elementParameter`. The list of attributes and namespaces are computed based on the selection in the `elementParameter` combo boxes.

Example of an `attributeParameter`:

```
<attributeParameter dependsOn="elemID">
    <localName label="Name" name="attribute_localName" useCurrentContext="true">
        <description>The name of the attribute to be converted.</description>
    </localName>
    <namespace label="Namespace" name="attribute_namespace" allowsAny="true">
        <description>Namespace of the attribute to be converted.</description>
    </namespace>
</attributeParameter>
```

Note: All predefined operations are loaded from the *[OXYGEN_INSTALL_DIR]*`/refactoring` folder.

Related Information:

Example of an Operation Descriptor File with an XSLT Stylesheet on page 506
Example of an Operation Descriptor File with an XQuery Update script on page 503

XQuery Update Script for Creating a Custom Operation

To demonstrate creating a custom operation, consider that we have a task where we need to convert an attribute into an element and insert it inside another element. A specific example would be if you have a project with a variety of `image` elements where a deprecated `alt` attribute was used for the description and you want to convert all instances of that attribute into an element with the same name and insert it as the first child of the `image` element.

Thus, our task is to convert this attribute into an element with the same name and insert it as the first child of the image element.

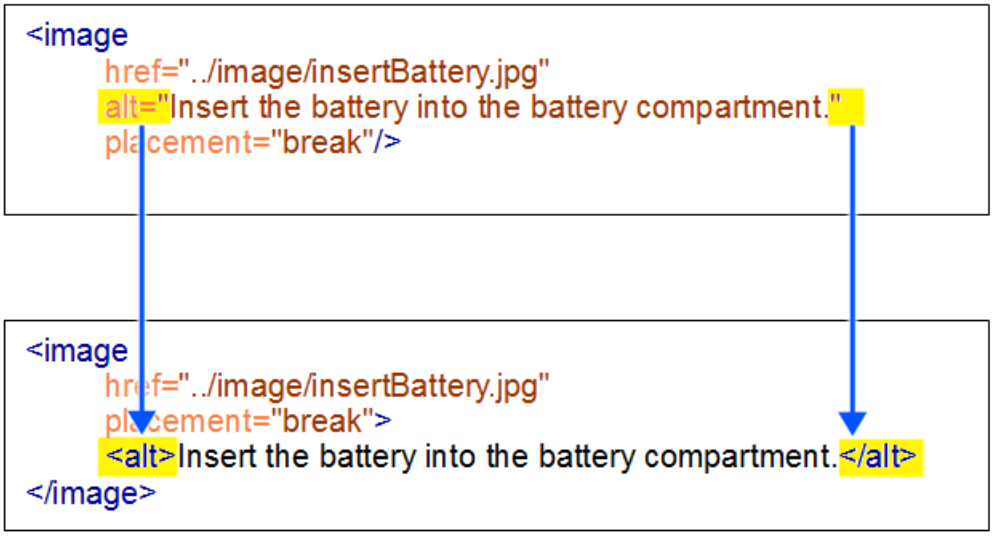


Figure 256: Example: Custom XML Refactoring Operation

We can use an XQuery Update script to implement the new custom XML refactoring operation. The second requirement is an *XML Refactoring operation descriptor file* that contains the path to the XQuery Update script.

Restriction: There is a limitation to using an XQuery script in that *comments* or *processing instructions* that are in any node before or after the root element cannot be modified by an XML Refactoring operation. In other words, XML Refactoring operations can only be performed on *comments* or *processing instructions* that are inside the root element. However, as a work around to this limitation, you can use *Saxon extension functions and the XSLT stylesheet method* to implement the new custom XML refactoring operation.

Example of an XQuery Update Script for Creating a Custom Operation to *Convert an Attribute to an Element*

The XQuery Update script does the following:

- Iterates over all elements from the document that have the specified local name and namespace.
- Finds the attribute that will be converted to an element.

- Computes the *QName* of the new element to be inserted and inserts it as the first child of the parent element.

```
(:
    XQuery document used to implement 'Convert attribute to element'
      operation from XML Refactoring tool.
:)

declare namespace output = "http://www.w3.org/2010/xslt-xquery-serialization";
declare option output:method   "xml";
declare option output:indent   "no";

(: Local name of the attribute's parent element. :)
declare variable $element_localName as xs:string external;

(: Namespace of the attribute's parent element. :)
declare variable $element_namespace as xs:string external;

(: The local name of the attribute to be converted :)
declare variable $attribute_localName as xs:string external;

(: The namespace of the attribute to be converted :)
declare variable $attribute_namespace as xs:string external;

(: Local name of the new element. :)
declare variable $new_element_localName as xs:string external;

(: Namespace of the new element. :)
declare variable $new_element_namespace as xs:string external;

(: Convert attribute to element:)
for $node in //*
(: Find the attribute to convert :)
let $attribute :=
    $node/@*[local-name() = $attribute_localName and
    ($attribute_namespace = '<ANY>' or $attribute_namespace = namespace-uri())]

(: Compute the prefix for the new element to insert :)
let $prefix :=
    for $p in in-scope-prefixes($node)
      where $new_element_namespace = namespace-uri-for-prefix($p, $node)
return $p

(: Compute the qname for the new element to insert :)
let $new_element_qName :=
    if (empty($prefix) or $prefix[1] = '') then $new_element_localName
    else $prefix[1] || ':' || $new_element_localName

 where ('<ANY>' = $element_localName or local-name($node) = $element_localName)
   and
 ($element_namespace = '<ANY>' or $element_namespace = namespace-uri($node))

  return
    if (exists($attribute)) then
      (insert node element {QName($new_element_namespace, $new_element_qName)}
      {string($attribute)} as first into $node,
      delete node $attribute)
      else ()
```

Example of an Operation Descriptor File That References the XQuery Script for Creating a Custom Operation to *Convert an Attribute to an Element*

After you have developed the XQuery script, you have to create an XML Refactoring operation descriptor. This descriptor is used by the application to load the operation details such as name, description, or parameters.

```
<?xml version="1.0" encoding="UTF-8"?>

<refactoringOperationDescriptor
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://www.oxygenxml.com/ns/xmlRefactoring"
 id="convert-attribute-to-element"
 name="Convert attribute to element">
 <description>Converts the specified attribute to an element.
           The new element will be inserted as first child of the attribute's
           parent element.</description>
 <!-- For the XSLT stylesheet option uncomment the following line and comment
           the line referring the XQuery Update script -->
 <!-- <script type="XSLT" href="convert-attribute-to-element.xsl"/> -->
 <script type="XQUERY_UPDATE" href="convert-attribute-to-element.xq"/>
  <parameters>
   <description>Specify the attribute to be converted to element.</description>
    <section label="Parent element">
     <elementParameter id="elemID">
      <localName label="Name" name="element_localName" allowsAny="true">
       <description>Local name of the parent element.</description>
        </localName>
      <namespace label="Namespace" name="element_namespace" allowsAny="true">
        <description>Local name of the parent element</description>
      </namespace>
     </elementParameter>
```

```
    </section>
    <section label="Attribute">
     <attributeParameter dependsOn="elemID">
      <localName label="Name" name="attribute_localName">
       <description>Name of the attribute to be converted.</description>
      </localName>
     <namespace label="Namespace" name="attribute_namespace" allowsAny="true">
       <description>Namespace of the attribute to be converted.</description>
     </namespace>
     </attributeParameter>
    </section>
    <section label="New element">
        <elementParameter>
           <localName label="Name" name="new_element_localName">
               <description>The name of the new element.</description>
           </localName>
           <namespace label="Namespace" name="new_element_namespace">
               <description>The namespace of the new element.</description>
           </namespace>
        </elementParameter>
    </section>
  </parameters>
</refactoringOperationDescriptor>
```

Results

After you have created these files, copy them into a folder *scanned by Oxygen XML Editor when it loads the custom operation*. When the XML Refactoring tool is started again, you will see the created operation.

Since various parameters can be specified, this custom operation can also be used for other similar tasks. The following image shows the parameters that can be specified in our example of the custom operation to convert an attribute to an element:

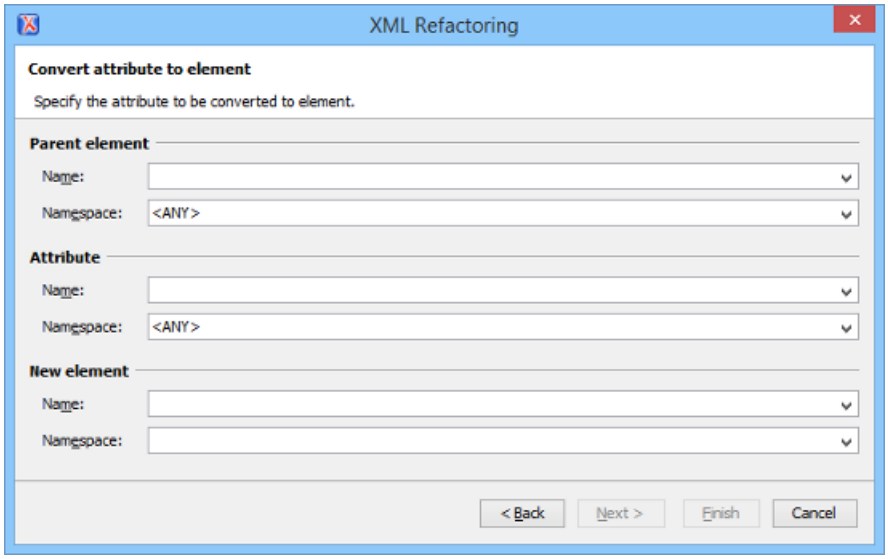


Figure 257: Example: XML Refactoring Wizard for a Custom Operation

XSLT Stylesheet for Creating a Custom Operation

To demonstrate creating a custom operation, consider that we have a task where we need to convert an attribute into an element and insert it inside another element. A specific example would be if you have a project with a variety of `image` elements where a deprecated `alt` attribute was used for the description and you want to convert all instances of that attribute into an element with the same name and insert it as the first child of the `image` element.

Thus, our task is to convert this attribute into an element with the same name and insert it as the first child of the image element.

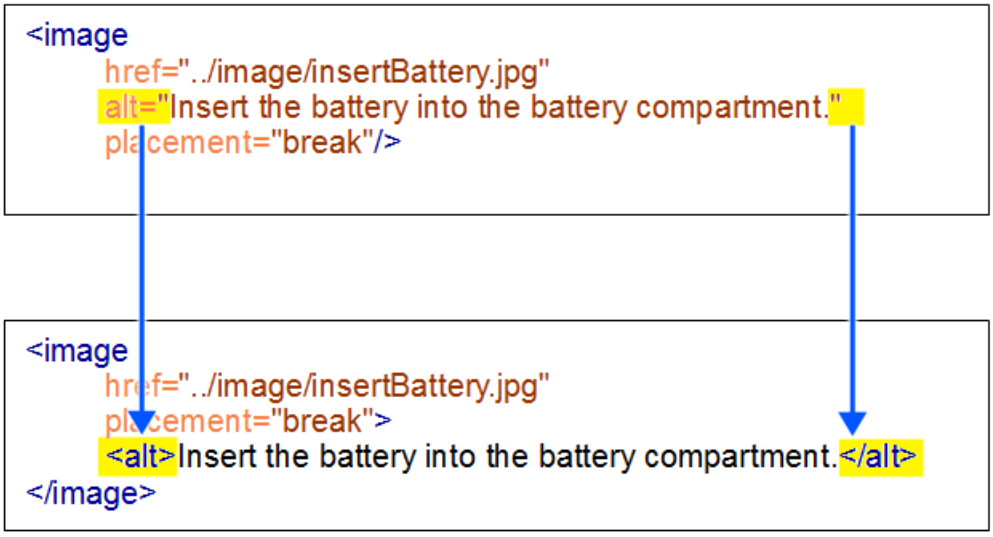


Figure 258: Example: Custom XML Refactoring Operation

We can use an XSLT stylesheet to implement the new custom XML refactoring operation. The second requirement is an *XML Refactoring operation descriptor file* that contains the path to the XSLT stylesheet.

Example of an XSLT Script for Creating a Custom Operation to *Convert an Attribute to an Element*

The XSLT stylesheet does the following:

- Iterates over all elements from the document that have the specified local name and namespace.
- Finds the attribute that will be converted to an element.
- Adds the new element as the first child of the parent element.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  exclude-result-prefixes="xs"
  xmlns:xr="http://www.oxygenxml.com/ns/xmlRefactoring"
  version="2.0">

  <xsl:import
href="http://www.oxygenxml.com/ns/xmlRefactoring/resources/commons.xsl"/>

  <xsl:param name="element_localName" as="xs:string" required="yes"/>
  <xsl:param name="element_namespace" as="xs:string" required="yes"/>
  <xsl:param name="attribute_localName" as="xs:string" required="yes"/>
  <xsl:param name="attribute_namespace" as="xs:string" required="yes"/>
  <xsl:param name="new_element_localName" as="xs:string" required="yes"/>
  <xsl:param name="new_element_namespace" as="xs:string" required="yes"/>

  <xsl:template match="node() | @*">
      <xsl:copy>
          <xsl:apply-templates select="node() | @*"/>
      </xsl:copy>
  </xsl:template>
  <xsl:template match="//*[xr:check-local-name($element_localName, ., true())
     and
        xr:check-namespace-uri($element_namespace, .)]">

      <xsl:variable name="attributeToConvert"
         select="@*[xr:check-local-name($attribute_localName, ., true())
     and
        xr:check-namespace-uri($attribute_namespace, .)]"/>

      <xsl:choose>
          <xsl:when test="empty($attributeToConvert)">
              <xsl:copy>
                  <xsl:apply-templates select="node() | @*"/>
              </xsl:copy>
          </xsl:when>
      <xsl:otherwise>
          <xsl:copy>
              <xsl:for-each select="@*[empty(. intersect $attributeToConvert)]">
                   <xsl:copy-of select="."/>
              </xsl:for-each>
                <!-- The new element namespace -->
```

```
                <xsl:variable name="nsURI" as="xs:string">
                    <xsl:choose>
            <xsl:when test="$new_element_namespace eq $xr:NO-NAMESPACE">
                            <xsl:value-of select="''"/>
                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:value-of select="$new_element_namespace"/>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:variable>
          <xsl:element name="{$new_element_localName}" namespace="{$nsURI}">
                    <xsl:value-of select="$attributeToConvert"/>
                </xsl:element>
             <xsl:apply-templates select="node()"/>
         </xsl:copy>
       </xsl:otherwise>
     </xsl:choose>
   </xsl:template>
 </xsl:stylesheet>
```

Note: The XSLT stylesheet imports a module library that contains utility functions and variables. The location of this module is resolved via an *XML Catalog* set in the XML Refactoring *framework*.

Example of an Operation Descriptor File That References the XSLT Stylesheet for Creating a Custom Operation to *Convert an Attribute to an Element*

After you have developed the XSLT stylesheet, you have to create an XML Refactoring operation descriptor. This descriptor is used by the application to load the operation details such as name, description, or parameters.

```
<?xml version="1.0" encoding="UTF-8"?>

<refactoringOperationDescriptor
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://www.oxygenxml.com/ns/xmlRefactoring"
 id="convert-attribute-to-element"
 name="Convert attribute to element">
 <description>Converts the specified attribute to an element.
           The new element will be inserted as first child of the attribute's
           parent element.</description>
 <!-- For the XSLT stylesheet option uncomment the following line and comment
           the line referring the XQuery Update script -->
 <!-- <script type="XSLT" href="convert-attribute-to-element.xsl"/> -->
 <script type="XQUERY_UPDATE" href="convert-attribute-to-element.xq"/>
  <parameters>
   <description>Specify the attribute to be converted to element.</description>
    <section label="Parent element">
     <elementParameter id="elemID">
      <localName label="Name" name="element_localName" allowsAny="true">
       <description>Local name of the parent element.</description>
        </localName>
       <namespace label="Namespace" name="element_namespace" allowsAny="true">
         <description>Local name of the parent element</description>
       </namespace>
     </elementParameter>
    </section>
    <section label="Attribute">
     <attributeParameter dependsOn="elemID">
      <localName label="Name" name="attribute_localName">
       <description>Name of the attribute to be converted.</description>
      </localName>
     <namespace label="Namespace" name="attribute_namespace" allowsAny="true">
       <description>Namespace of the attribute to be converted.</description>
     </namespace>
     </attributeParameter>
    </section>
    <section label="New element">
        <elementParameter>
           <localName label="Name" name="new_element_localName">
               <description>The name of the new element.</description>
           </localName>
           <namespace label="Namespace" name="new_element_namespace">
               <description>The namespace of the new element.</description>
           </namespace>
        </elementParameter>
    </section>
  </parameters>
</refactoringOperationDescriptor>
```

Note: If you are using an XSLT file, the line with the `script` element would look like this:

```
 <script type="XSLT" href="convert-attribute-to-element.xsl"/>
```

Results

After you have created these files, copy them into a folder *scanned by Oxygen XML Editor when it loads the custom operation*. When the XML Refactoring tool is started again, you will see the created operation.

Since various parameters can be specified, this custom operation can also be used for other similar tasks. The following image shows the parameters that can be specified in our example of the custom operation to convert an attribute to an element:

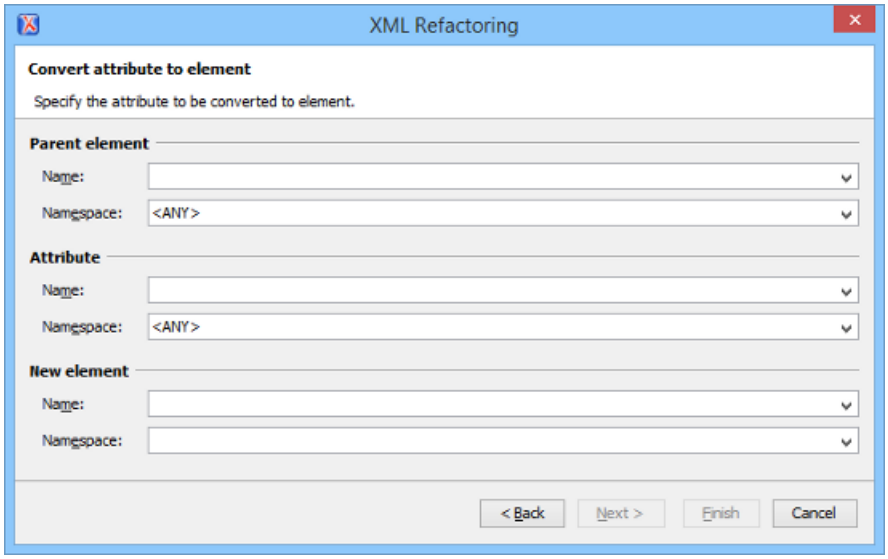


Figure 259: Example: XML Refactoring Wizard for a Custom Operation

Using Saxon Extension Functions to Allow Custom Refactoring Operations to Read and Modify Content Outside the Root Node

One advantage to using an XSLT stylesheet is that there is limitation when using an *XQuery Update script* in that refactoring operations can only be performed on *comments* or *processing instructions* that are inside the root element. Thus, using the XQuery method, *comments* or *processing instructions* that are in any node before or after the root element cannot be modified by an XML Refactoring operation.

The XSLT stylesheet method offers a work around to this limitation through the use of some Saxon extension functions.

To illustrate the use of these functions, consider the following sample XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- comment before root -->
<?pi before root ?>
<root>
    <child></child>
</root>
<!-- comment after root -->
<?pi after root ?>
```

The following Saxon extension functions can be used to read and modify content outside the root node:

Note: They belong to the `http://www.oxygenxml.com/ns/xmlRefactoring/functions` namespace.

- **get-content-after-root()** - Returns the content after root as `xs:string`.

  For the XML above, the call of this function will return the following string value:

  ```
  <!-- comment after root -->
  <?pi after root ?>
  ```

- **set-content-after-root(xs:string)** - Updates the content that will be serialized in the refactored document after the root node.

The function call `set-content-after-root('<!-- Inserted comment -->')` will result in replacing the nodes after the root element with the comment passed as string argument. The XML document will be modified as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- comment before root -->
<?pi before root ?>
<root>
    <child></child>
</root><!-- Inserted comment -->
```

- **get-content-before-root()** - Returns the content before root as `xs:string`.

  For the XML above, the call of this function will return the following string value:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- comment before root -->
<?pi before root ?>
```

- **set-content-before-root(xs:string)** - Updates the content that will be serialized in the refactored document after the root node.

  The function call `set-content-before-root('<!-- Inserted comment -->')` will result in replacing the nodes before the root element with the comment passed as string argument. The XML document will be modified as follows:

```
<!-- Inserted comment --><root>
    <child></child>
</root>
<!-- comment after root -->
<?pi after root ?>
```

XSLT Example:

To process content after the root node, the XSLT would look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" exclude-result-prefixes="xs"
    xmlns:xrf="http://www.oxygenxml.com/ns/xmlRefactoring/functions" version="3.0">
    <xsl:template match="/">
        <!-- The comment content that will be inserted after the root element -->
        <xsl:variable name="commentAsText">&lt;!-- COMMENT ADDED FROM XR OPERATION-->
        </xsl:variable>
        <!-- Retrieve the content after the root element as is -->
        <xsl:variable name="after-root-content" as="xs:string"
                            select="xrf:get-content-after-root()"/>

        <xsl:variable name="processedContent"
                            select="concat($after-root-content, $commentAsText)"/>

        <!-- Update the content after the root element -->
        <xsl:value-of select="xrf:set-content-after-root($processedContent)"/>

        <xsl:apply-templates/>
    </xsl:template>

    <xsl:template match="node() | @*">
        <xsl:copy>
            <xsl:apply-templates select="node() | @*"/>
        </xsl:copy>
    </xsl:template>
</xsl:stylesheet>
```

Note: The above XSLT retrieves the nodes after the root element as string, appends a new comment, and then sets back the updated content into the XML document.

Storing and Sharing Refactoring Operations

Oxygen XML Editor scans the following locations when looking for XML Refactoring operations to provide flexibility:

- A `refactoring` folder, created inside a directory that is associated to a *framework* you are customizing.
- A folder that you specify in the ***Load additional refactoring operations from*** *text box* in the ***XML Refactoring*** *preferences page*.

Note: If you share a project with your team, you can also share the custom operation by doing the following: Then by doing the following:

1. Save the custom operation in a folder that is part of your project.
2. Switch the **XML Refactoring** option page to *project level*:
   a. *Open the **Preferences** dialog box (**Options** > **Preferences**)* and go to ***XML** > **XML Refactoring***.
   b. Select ***Project Options*** at the bottom of the dialog box.
3. In the ***Load additional refactoring operations from** text box*, use the *`${pd}` editor variable* so that the folder path is declared relative to the project.

- A folder specified by the *XML Refactoring Operations Plugin Extension*.
- The `refactoring` folder from the Oxygen XML Editor installation directory (*`[OXYGEN_INSTALL_DIR]`*`/refactoring/`).

Sharing Custom Refactoring Operations

The purpose of Oxygen XML Editor scanning multiple locations for the XML Refactoring operations is to provide more flexibility for developers who want to share the refactoring operations with the other team members. Depending on your particular use case, you can attach the custom refactoring operations to other resources, such as *framework* or projects.

After storing custom operations, you can share them with other users by sharing the resources.

Localizing XML Refactoring Operations

Oxygen XML Editor includes localization support for the XML refactoring operations.

The translation keys for the built-in refactoring operations are located in *`[OXYGEN_INSTALL_DIR]`*`/refactoring/i18n/translation.xml`.

The localization support is also available for custom refactoring operations. The following information can be translated:

- The operation name, `description`, and `category`.
- The `description` of the `parameters` element.
- The `label`, `description`, and `possibleValues` for each `parameter`.

Translated refactoring information uses the following form:

```
${i18n(translation_key)}
```

Oxygen XML Editor scans the following locations to find the `translation.xml` files that are used to load the translation keys:

- A `refactoring/i18n` folder, created inside a directory that is associated to a customized *framework*.
- A `i18n` folder, created inside a directory that is associated to a customized *framework*.
- An `i18n` folder inside any specified folder. In this case, you need to *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **XML** > **XML Refactoring**, and specify the folder in the **Load additional refactoring operations from** text box.
- An `i18n` folder located in directories specified through the *XML Refactoring Operations Plugin Extension*.
- The `refactoring/i18n` folder from the Oxygen XML Editor installation directory (*`[OXYGEN_INSTALL_DIR]`*`/refactoring/i18n`).

Example: Refactoring Operation Descriptor File with *i18n* Support

```
<?xml version="1.0" encoding="UTF-8"?>

<refactoringOperationDescriptor
    xmlns="http://www.oxygenxml.com/ns/xmlRefactoring" id="remove_text_content"
           name="${i18n(Remove_text_content)}">
    <description>${i18n(Remove_text_content_description)}</description>
    <script type="XQUERY_UPDATE" href="remove_text_content.xq"/>
    <parameters>
        <description>${i18n(parameters_description)}</description>
        <parameter label="${i18n(Element_name)}" name="element_localName"
                      type="NC_NAME">
            <description>${i18n(Element_name_descriptor)}</description>
            <possibleValues>
```

```
                <value default="true" name="value1">${i18n(value_1)}</value>
                <value name="value2">${i18n(value_2)}</value>
            </possibleValues>
        </parameter>
    </parameters>
</refactoringOperationDescriptor>
```

Editing XSLT Stylesheets

Oxygen XML Editor includes a built-in editor for XSLT stylesheets. This section presents the features of the XSLT editor and how these features can be used. The features of the XSLT editor include:

- **Create new XSLT files and templates** - You can use the built-in new file wizards to *create new XSLT documents or templates*.
- **Open and Edit XSLT files** - XSLT files can be opened and edited in the source editor (***Text** mode*).
- **Visual Editing** - XSLT stylesheets are rendered, and can be edited, in the *visual **Author** editing mode*.
- **Validation** - Presents validation errors in XSLT files.
- **Content completion** - Offers proposals for properties and the values that are available for each property.
- **Syntax highlighting** - The syntax highlighting in Oxygen XML Editor makes XSLT files more readable.

To watch our video demonstration about basic XSLT editing and transformation scenarios in Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/XSL_Editing.html*.

Editing XSLT Stylesheets in the Master Files Context

Smaller interrelated modules that define a complex stylesheet cannot be correctly edited or validated individually, due to their interdependency with other modules. For example, a function defined in a main stylesheet is not visible when you edit an included or imported module. Oxygen XML Editor provides the support for defining the main module (or modules), allowing you to edit any of the imported/included files in the context of the larger stylesheet structure.

You cat set a main XSLT stylesheet either using the *master files support from the **Project** view*, or using a validation scenario.

To set a *master file* using a validation scenario, add validation units that point to the main modules. Oxygen XML Editor warns you if the current module is not part of the dependencies graph computed for the main stylesheet. In this case, it considers the current module as the main stylesheet.

The advantages of editing in the context of *master file* include:

- Correct validation of a module in the context of a larger stylesheet structure.
- *Content Completion Assistant* displays all components valid in the current context.
- The ***Outline** view* displays the components collected from the entire stylesheet structure.

To watch our video demonstration about editing XSLT stylesheets in the *master files* context, go to *https://www.oxygenxml.com/demo/MasterFilesSupport.html*.

Related Information:

XSLT Resource Hierarchy/Dependencies View on page 525

XSLT Component Dependencies View on page 528

Validating XSLT Stylesheets

Numerous XSLT code quality assurance checks are done during automatic validation to help you keep your stylesheets valid and well formed. Oxygen XML Editor performs the validation of XSLT documents with the help of an XSLT processor *that you can configure in the preferences pages* according to the XSLT version.

For XSLT 1.0, the options are: Xalan, Saxon 6.5.5, Saxon 9.7.0.15 and *a JAXP transformer specified by the main Java class.* For XSLT 2.0, the options are: Saxon 9.7.0.15 and *a JAXP transformer specified by the main Java class.* For XSLT 3.0, the options are Saxon 9.7.0.15 and *a JAXP transformer specified by the main Java class.*

To access the *XSLT preferences* quickly, use the **Validation options** action from the **Document** > **Validate** menu.

Creating a Validation Scenario for XSLT Stylesheets

You can validate an XSLT document using the engine defined in the transformation scenario, or a custom validation scenario. If you choose to validate using the engine from transformation scenario, and a transformation scenario is not associated with the current document or the engine has no validation support, the default engine is used. To set the default engine, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XSLT/FO/XQuery** > **XSLT**.

You can also create new validation scenarios or edit existing ones, and you can add *JARS* and classes that contain extension functions. To create or edit a validation scenario for an XSLT stylesheet, follow these steps:

1. With the XSLT file opened in Oxygen XML Editor, select the **Configure Validation Scenario(s)** from the **Document** > **Validate** menu, or the **Validation** toolbar drop-down menu, or from the **Validate** submenu when invoking the contextual menu on the XSLT file in the ***Project** view*.
   The **Configure Validation Scenario(s)** dialog box is displayed. It contains the existing scenarios, organized in categories depending on the type of file they apply to. You can use the options in the **Settings** drop-down menu to filter which scenarios are shown.
2. To edit an existing scenario, select the scenario and press the **Edit** button. If you try to edit one of the *read-only* predefined scenarios, Oxygen XML Editor creates a customizable duplicate (you can also use the **Duplicate** button).
3. To add a new scenario, press the **New** button.
   The **New scenarios** dialog box is displayed. It lists all validation units of the scenario.

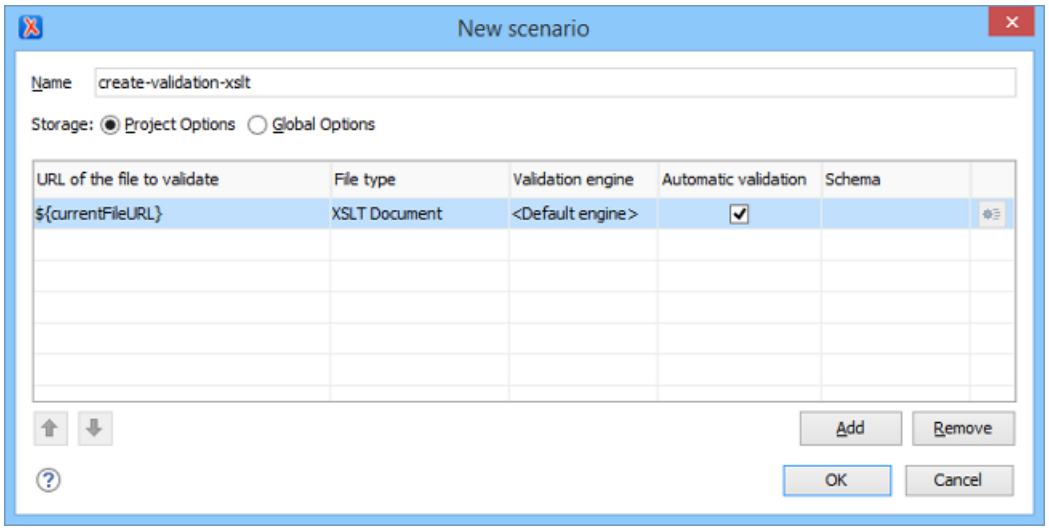


Figure 260: Add / Edit a Validation Unit

4. Configure the following information in this dialog box:
   a) **Name** - The name of the validation scenario.
   b) **Storage** - You can choose between storing the scenario in the ***Project Options*** or ***Global Options***.
   c) **URL of the file to validate** - In most cases, leave this field as the default selection (the URL of the current file). If you want to specify a different URL, double-click its cell and enter the URL in the text field, select it from the drop-down list, or use the **Browse** drop-down menu or ***Insert Editor Variable*** button.
   d) **File type** - The file type should be `XSLT Document`.
   e) **Validation engine** - Click the cell to select a validation engine. You must select an engine to be able to add or edit extensions.
   f) **Automatic validation** - If this option is selected, the validation operation defined by this row is also used by *the automatic validation feature*.
5. To add or edit extensions, click the **Edit extensions** button. This button is only available if the **File type** is set as `XSLT Document` and a **Validation engine** is chosen.
   The **Libraries** dialog box is opened. It is used to specify the *JARS* and classes that contain extension functions called from the XSLT file of the current validation scenario. They will be searched, in the specified extensions,

in the order displayed in this dialog box. To change the order of the items, select the item and press the **Move up** or **Move down** buttons.

6. Press **OK** to close the **New scenario** dialog box.
   The newly created validation scenario is now included in the list of scenarios in the **Configure Validation Scenario(s)** dialog box. You can select the scenario in this dialog box to associate it with the current XSLT document and press the **Apply associated** button to run the validation scenario.

Validating XSLT Stylesheets with Custom Engines

If you need to validate an XSLT stylesheet with a validation engine that is different from the built-in engine, you can configure external engines as custom XSLT validation engines in the Oxygen XML Editor preferences. After a custom validation engine is *properly configured*, it can be applied on the current document by selecting it from the list of custom validation engines in the **Validation** toolbar drop-down menu. The document is validated against the schema declared in the document.

By default, there are two validators that are configured for XSLT stylesheets:

- **MSXML 4.0** - included in Oxygen XML Editor (Windows edition). It is associated to the XSL Editor type in *Preferences page.*
- **MSXML.NET** - included in Oxygen XML Editor (Windows edition). It is associated to the XSL Editor type in *Preferences page.*

XSLT Quick Fix Support

The Oxygen XML Editor *Quick Fix support* helps you resolve various errors that appear in a stylesheet by proposing *Quick Fixes* to problems such as missing templates, misspelled template names, missing functions, or references to an undeclared variable or parameter.

To activate this feature, hover over or place the cursor in the highlighted area of text where a validation error or warning occurs. If a *Quick Fix* is available for that particular error or warning, you can access the *Quick Fix* proposals with any of the following methods:

- When hovering over the error or warning, the proposals are presented in a tooltip pop-up window.
- If you place the cursor in the highlighted area where a validation error or warning occurs, a *Quick Fix* icon ( ) is displayed in the stripe on the left side of the editor. If you click this icon, Oxygen XML Editor displays the list of available fixes.
- With the cursor placed in the highlighted area of the error or warning, you can also invoke the *Quick Fix* menu by pressing **Alt + 1 (Command + Alt + 1 on OS X)** on your keyboard.

Note: The *Quick Fixes* are available only when validating an XSLT file with Saxon HE/PE/EE.

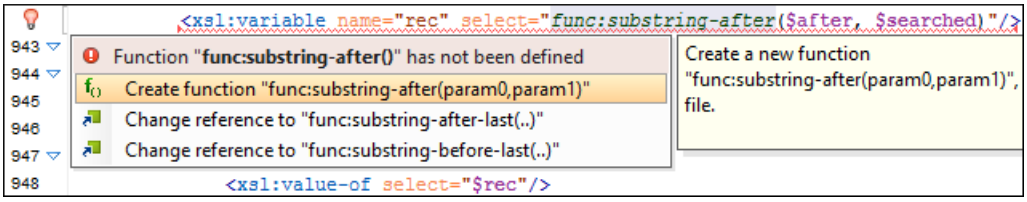


Figure 261: Example of an Undefined XSLT Functions Quick Fix

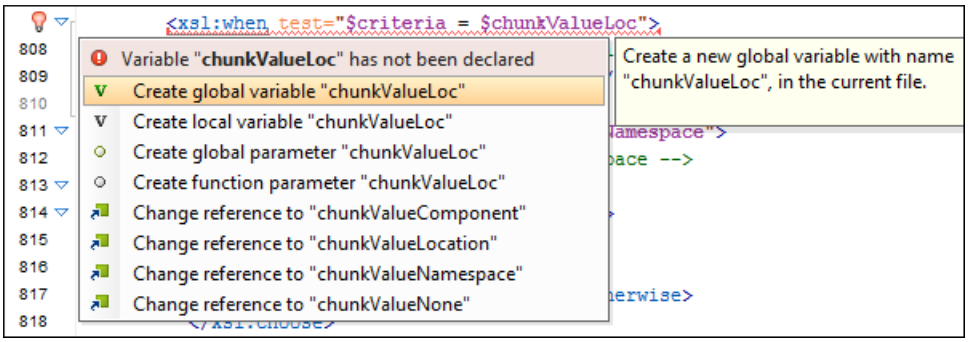


Figure 262: Example of an Undeclared XSLT Variables/Parameters Quick Fix

Oxygen XML Editor provides XSLT *Quick Fixes* for the following types of instances:

- **Template does not exist**, when the template name referenced in a `call-template` element does not exist. The following fixes are available:
  - **Create template "templateName"** - creates a template and generates its corresponding parameters. The template name and parameter names and types are collected from the `call-template` element.
  - **Change reference to "newTemplateName"** - changes the name of the missing template referenced in the `call-template` element. The proposed new names are the existing templates with names similar with the missing one.
- **Variable/Parameter not declared**, when a parameter or variable reference cannot be found. The following fixes are available:
  - **Create global variable "varName"** - creates a global variable with the specified name in the current stylesheet. The new variable is added at the beginning of the stylesheet after the last global variable or parameter declaration.
  - **Create global parameter "paramName"** - creates a global parameter with the specified name in the current stylesheet. The new parameter is added at the beginning of the stylesheet after the last global parameter or variable declaration.
  - **Create local variable "varName"** - creates a local variable with the specified name before the current element.
  - **Create template parameter "paramName"** - creates a new parameter with the specified name in the current template. This fix is available if the error is located inside a template.
  - **Create function parameter "paramName"** - creates a new parameter with the specified name in the current function. This fix is available if the error is located inside a function.
  - **Change reference to "varName"** - changes the name of the referenced variable/parameter to an existing local or global variable/parameter, that has a similar name with the current one.
- **Parameter from a called template is not declared**, when a parameter referenced from a `call-template` element is not declared. The following fixes are available:
  - **Create parameter "paramName" in the template "templateName"** - creates a new parameter with the specified name in the referenced template.
  - **Change "paramName" parameter reference to "newParamName"** - changes the parameter reference from the `call-template` element to a parameter that is declared in the called template.
  - **Remove parameter "paramName" from call-template** - removes the parameter with the specified name from the `call-template` element.
- **No value supplied for required parameter**, when a required parameter from a template is not referenced in a `call-template` element. The following quick-fix is available::
  - **Add parameter "paramName" in call-template** - creates a new parameter with the specified name in call-template element.
- **Function "prefix:functionName()" has not been defined**, when a function declaration is not found. The following *Quick Fixes* are available:
  - **Create function "prefix:functionName(param1, param2)"** - creates a new function with the specified signature, after the current top level element from stylesheet.

- **Change function to "newFunctionName(..)"** - changes the referenced function name to an already defined function. The proposed names are collected from functions with similar names and the same number of parameters.
- **Attribute-set "attrSetName" does not exist**, when the referenced attribute set does not exist. The following *Quick Fixes* are available:
  - **Create attribute-set "attrSetName"** - creates a new attribute set with the specified name, after the current top level element from stylesheet.
  - **Change reference to "attrSetName"** - changes the referenced attribute set to an already defined one.
- **Character-map "chacterMap" has not been defined**, when the referenced character map declaration is not found. The following *Quick Fixes* are available:
  - **Create character-map "characterMapName"** - creates a new character map with the specified name, after the current top level element from stylesheet.
  - **Change reference to "characterMapName"** - changes the referenced character map to an already defined one.

Content Completion in XSLT Stylesheets

The list of proposals offered by the *Content Completion Assistant* in XSLT are context-sensitive and includes proposals that are valid at the current cursor position. It can be manually activated with the **Ctrl + Space (Command + Space on OS X)** shortcut.

You can enhance the list of proposals by specifying an additional schema. This schema is defined in the ***Content Completion / XSLT** preferences* page and can be any of the following: XML Schema, DTD, RELAX NG schema, or NVDL schema.

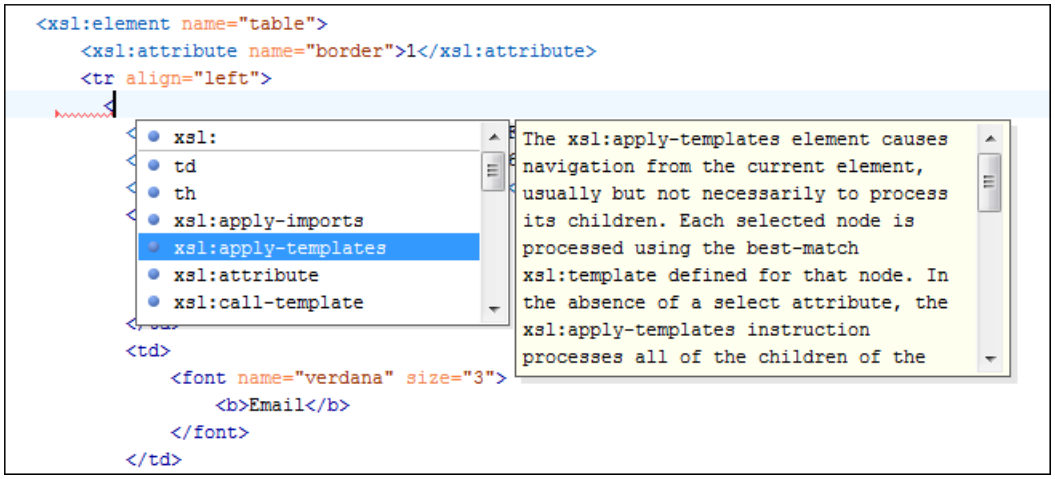


Figure 263: XSLT Content Completion Assistant

The feature is activated in **Text** mode in the following situations:

- After you press the **<** character when inserting an element, it is automatically activated after a short delay. You can adjust the activation delay with the ***Activation delay of the proposals window (ms)** option* from the **Content Completion** preferences page.
- After typing a partial element or attribute name, you can manually activate it by pressing **Ctrl + Space (Command + Space on OS X)** or **Alt + ForwardSlash (Command + Alt + ForwardSlash on OS X)**. If there is only one valid proposal at the current location, it is inserted without displaying the list of proposals.

The *Content Completion Assistant* proposes numerous item types (such as templates, variables, parameters, keys, etc.) that are defined in the current stylesheet, and in the imported and included XSLT stylesheets. The *Content Completion Assistant* also includes *code templates that can be used to quickly insert code fragments* into stylesheets.

Note: For XSL and XSD resources, the *Content Completion Assistant* collects its components starting from the *master files*. The *master files* can be defined in the project or in the associated validation scenario. For further details about the *Master Files* support go to *Defining Master Files at Project Level*.

The extension functions included in the Saxon 6.5.5 and 9.7.0.15 transformation engines are presented in the content completion list only if the Saxon namespace (*http://saxon.sf.net* for XSLT version 2.0 / 3.0 or *http://icl.com/saxon* for XSLT version 1.0) is declared and one of the following conditions is true:

- The edited file has a transformation scenario that uses as transformation engine Saxon 6.5.5 (for XSLT version 1.0), Saxon 9.7.0.15 PE or Saxon 9.7.0.15 EE (for XSLT version 2.0 / 3.0).
- The edited file has a validation scenario that uses as validation engine Saxon 6.5.5 (for version 1.0), Saxon 9.7.0.15 PE or Saxon 9.7.0.15 EE (for version 2.0 / 3.0).
- The validation engine specified in *Options* page is Saxon 6.5.5 (for version 1.0), Saxon 9.7.0.15 PE or Saxon 9.7.0.15 EE (for version 2.0 / 3.0).

Additionally. the Saxon-CE-specific extension functions and instructions are presented in the list of content completion assistance proposals only if the `http://saxonica.com/ns/interactiveXSLT` namespace is declared.

Namespace prefixes in the scope of the current context are presented at the top of the content completion assistance window to speed up the insertion into the document of prefixed elements.

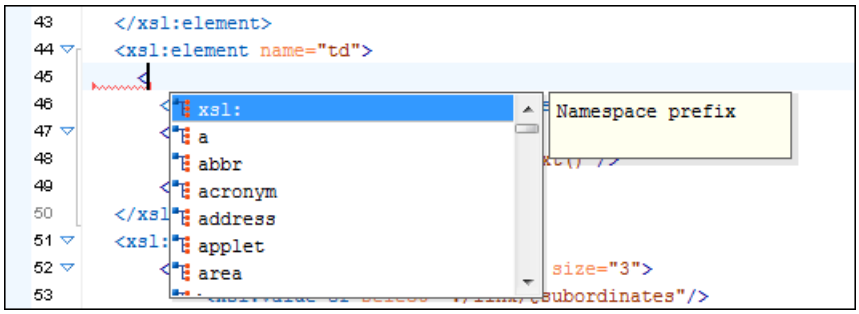


Figure 264: Namespace Prefixes in the Content Completion Assistant

For the common namespaces such as XSL namespace (`http://www.w3.org/1999/XSL/Transform`), XML Schema namespace (`http://www.w3.org/2001/XMLSchema`), or Saxon namespace (`http://icl.com/saxon` for version 1.0, `http://saxon.sf.net/` for version 2.0 / 3.0), Oxygen XML Editor provides an easy mode to declare them by proposing a prefix for these namespaces.

Content Completion in XPath Expressions

In XSLT stylesheets, the *Content Completion Assistant* provides *all the features available in the XML editor* and also adds some enhancements. In XPath expressions used in attributes of XSLT stylesheets (elements such as `match`, `select`, and `test`), the *Content Completion Assistant* offers the names of XPath and XSLT functions, XSLT axes, and user-defined functions (the name of the function and its parameters). If a transformation scenario was defined and associated to the edited stylesheet, the *Content Completion Assistant* computes and presents elements and attributes based on:

- The input XML document selected in the scenario.
- The current context in the stylesheet.

The associated document is displayed in the ***XSLT/XQuery Input*** *view*.

Content completion for XPath expressions is started:

- On XPath operators detected in one of the `match`, `select` and `test` attributes of XSLT elements: ", ', /, //, (, [, |, :, ::, $
- For attribute value templates of non-XSLT elements, that is the { character when detected as the first character of the attribute value.
- On request, if the combination **Ctrl + Space (Command + Space on OS X)** is pressed inside an edited XPath expression.

The proposals presented in the *Content Completion Assistant* are dependent on:

- The context of the current XSLT element.
- The XML document associated with the edited stylesheet in the stylesheet transformation scenario.
- The XSLT version of the stylesheet (1.0, 2.0, or 3.0).

Note: The XSLT 3.0 content completion list of proposals includes specific elements and attributes for the 3.0 version.

For example, if the document associated with the edited stylesheet is:

```
<personnel>
    <person id="Big.Boss">
        <name>
            <family>Boss</family>
            <given>Big</given>
        </name>
        <email>chief@oxygenxml.com</email>
        <link subordinates="one.worker"/>
    </person>
    <person id="one.worker">
        <name>
            <family>Worker</family>
            <given>One</given>
        </name>
        <email>one@oxygenxml.com</email>
        <link manager="Big.Boss"/>
    </person>
</personnel>
```

If you enter an `xsl:template` element using the *Content Completion Assistant*, the following actions are triggered:

- The `match` attribute is inserted automatically.
- The cursor is placed between the quotes.
- The XPath *Content Completion Assistant* automatically displays a pop-up window with all the XSLT axes, XPath functions and elements and attributes from the XML input document that can be inserted in the current context.

The set of XPath functions depends on the XSLT version declared in the root element `xsl:stylesheet`: 1.0, 2.0, or 3.0.

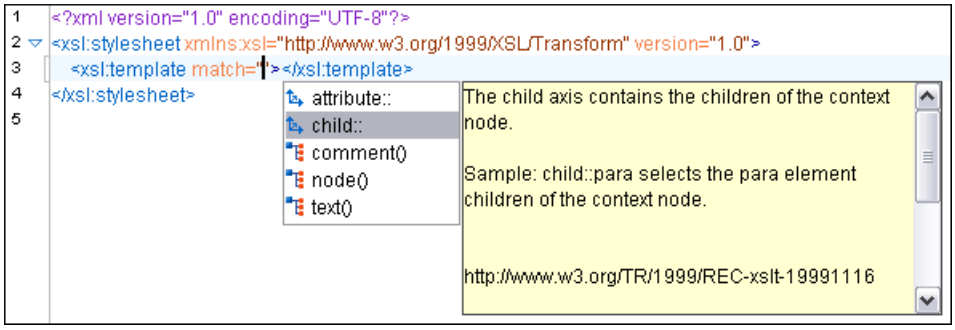


Figure 265: Content Completion in the `match` Attribute

If the cursor is inside the `select` attribute of an `xsl:for-each`, `xsl:apply-templates`, `xsl:value-of` or `xsl:copy-of` element the content completion proposals depend on the path obtained by concatenating the XPath expressions of the parent XSLT elements `xsl:template` and `xsl:for-each` as shown in the following figure:

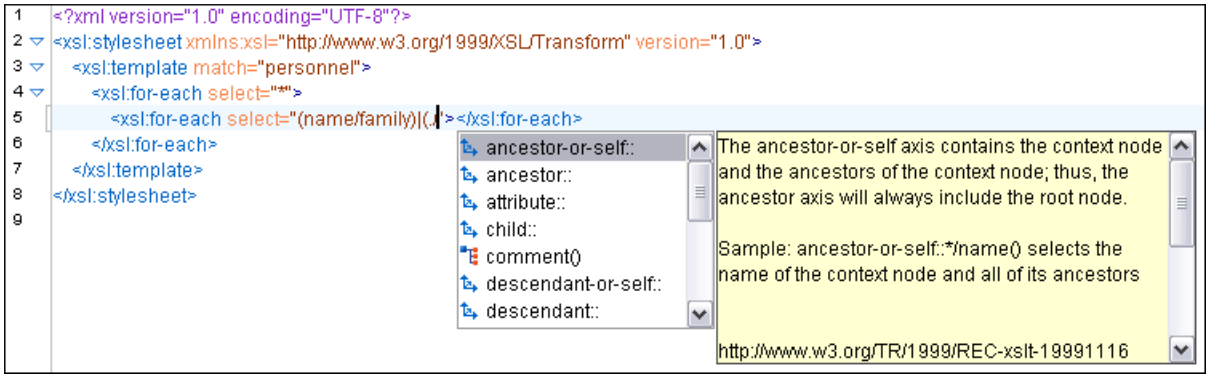


Figure 266: Content Completion in the `select` Attribute

Also XPath expressions typed in the `test` attribute of an `xsl:if` or `xsl:when` element benefit of the assistance of the content completion.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
3    <xsl:template match="personnel">
4      <xsl:for-each select="*">
5        <xsl:for-each select="name|link">
6          <xsl:if test="family|given|"></xsl:if>
7        </xsl:for-each>
8      </xsl:for-each>
9    </xsl:template>
10 </xsl:stylesheet>
11
```

```
name
QName
abs
adjust-date-to-timezone
adjust-dateTime-to-timezone
adjust-time-to-timezone
ancestor-or-self::
```

```
fn:QName($paramURI as xs:string?,
$paramQName as xs:string) as xs:QName

Returns an xs:QName with the namespace URI
given in $paramURI. If $paramURI is the
zero-length string or the empty sequence, it
represents "no namespace"; in this case, if the
value of $paramQName contains a colon (:), an
error is raised [err:FOCA0002]. The prefix (or
```

Figure 267: Content Completion in the `test` Attribute

XSLT variable references are easier to insert in XPath expressions with the help of the content completion pop-up triggered by the $ character, which signals the start of such a reference in an XPath expression.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
3    <xsl:template match="personnel">
4      <xsl:variable name="manager" select="*[1]/link/@manager"></xsl:variable>
5
6      <xsl:variable name="subord" select="*[1]/link/@subordinates"></xsl:variable>
7      <xsl:for-each select="*">
8        <xsl:value-of select="name/given"></xsl:value-of>
9        <xsl:value-of select="$"></xsl:value-of>
10     </xsl:for-each>
11
12   </xsl:template>
13 </xsl:stylesheet>
14
```

```
manager
subord
```

Figure 268: Content Completion in the `test` Attribute

If the { character is the first one in the value of the attribute, the same *Content Completion Assistant* is available also in attribute value templates of non-XSLT elements.

```
3    <xsl:template match="myTag">
4      <xsl:variable name="var" select="*[1]/@type"/>
5      <a href="{"></a>
6    </xsl:template
7  </xsl:stylesheet>
8
```

```
name
ancestor-or-self::
ancestor::
attribute::
boolean
ceiling
child::
```

```
name(node-set?) as string

The name function returns a string containing a
QName representing the expanded-name of the
node in the argument node-set that is first in
document order. The QName must represent the
expanded-name with respect to the namespace
declarations in effect on the node whose
expanded-name is being represented. Typically,
```

Figure 269: Content Completion in Attribute Value Templates

The time delay (*configured in the **Content Completion** preferences page*) for all content completion assistance windows is also applied for the content completion in XPath expressions.

Related Information:

Working with XPath Expressions on page 1057

Tooltip Helper for the XPath Functions Arguments

When editing the arguments of an XPath/XSLT function, Oxygen XML Editor tracks the current entered argument by displaying a tooltip containing the function signature. The currently edited argument is highlighted with a bolder font.

When moving the cursor through the expression, the tooltip is updated to reflect the argument found at the cursor position.

Examples:

If you want to concatenate the absolute values of two variables, named *v1* and *v2*:

```
<xsl:template match="/">
    <xsl:value-of select="concat(abs($v1), abs($v2))"></xsl:value-of>
</xsl:template>
```

When moving the cursor before the first `abs` function, Oxygen XML Editor identifies it as the first argument of the `concat` function. The tooltip shows in bold font the following information about the first argument:

- Its name is `$arg1`.
- Its type is `xdt:anyAtomicType`.
- It is optional (note the ? sign after the argument type).

The function also takes other arguments that have the same type and returns a `xs:string`.

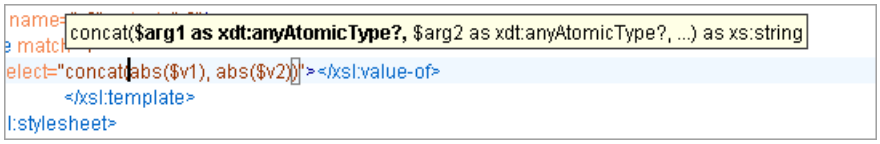


Figure 270: XPath Tooltip Helper - Identify the `concat` Function's First Argument

Moving the cursor on the first variable `$v1`, the editor identifies the `abs` as context function and shows its signature:

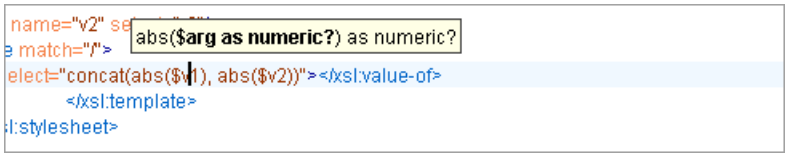


Figure 271: XPath Tooltip Helper - Identify the `abs` Function's Argument

Further, clicking the second `abs` function name, the editor detects that it represents the second argument of the `concat` function. The tooltip is repainted to display the second argument in bold font.

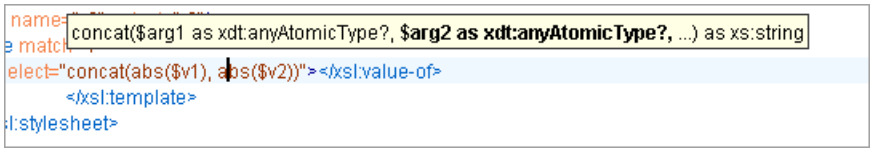


Figure 272: XPath Tooltip Helper - Identify the `concat` Function's Second Argument

Note: The tooltip helper is also available in the ***XPath Builder*** *view* and *XPath toolbar*.

Related Information:

Working with XPath Expressions on page 1057

Syntax Highlighting in XSLT

Oxygen XML Editor supports syntax highlighting in XSLT documents to improve the readability of the content and reduce the time it takes to internalize the semantics of the structured content.

To customize the colors or styles used for the syntax highlighting colors for XSLT files, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)***.
2. Go to ***Editor*** > ***Syntax Highlight***.
3. Select and expand the **XML** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the **XSL** tab in the **Preview** pane to see the effects of your changes.

Tip: Oxygen XML Editor also allows you to specify syntax highlighting colors for specific XML elements and attributes with specific namespace prefixes. This can be done in the ***Editor*** > ***Syntax Highlight*** > ***Elements/ Attributes by Prefix*** *preferences page*.

Related Information:

Customize Syntax Highlight colors on page 104

XSLT Outline View

The **Outline** view for XSLT stylesheets displays the list of all the components (templates, attribute-sets, character-maps, variables, functions, keys, outputs) from both the edited stylesheet and its imports or includes. For XSL and XSD resources, the **Outline** view collects its components starting from the *master files*. The master files can be defined in the project or in the associated validation scenario. For further details about the *Master Files* support go to *Defining Master Files at Project Level*.

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

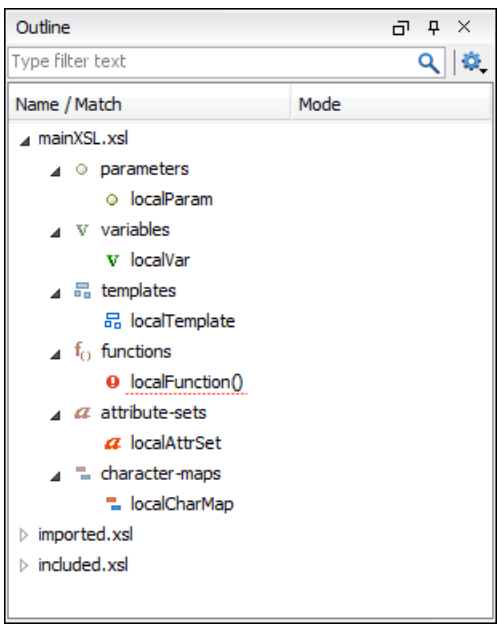


Figure 273: XSLT Outline View

The following actions are available in the **Settings** menu on the **Outline** view toolbar:

Filter returns exact matches

The text filter of the **Outline** view returns only exact matches;

Selection update on cursor move

Controls the synchronization between **Outline** view and source document. The selection in the **Outline** view can be synchronized with the cursor moves or the changes in the XSLT editor. Selecting one of the components from the **Outline** view also selects the corresponding item in the source document.

When the **Show components** option is selected, the following actions are available:

Show XML structure

Displays the XML document structure in a tree-like structure.

Show all components

Displays all components that were collected starting from the *master file*. This option is set by default.

Show only local components

Displays the components defined in the current file only.

Group by location/type

The stylesheet components can be grouped by location and type.

When the **Show XML structure** option is selected, the following actions are available:

Show components

Switches the **Outline** view to the components display mode.

Flat presentation mode of the filtered results

When active, the application flattens the filtered result elements to a single level.

Show comments and processing instructions

Show/hide comments and processing instructions in the **Outline** view.

Show element name

Show/hide element name.

Show text

Show/hide additional text content for the displayed elements.

Show attributes

Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from *the Outline preferences panel*.

Configure displayed attributes

Displays the ***XML Structured Outline*** *preferences page*.

The following contextual menu actions are also available when the **Show components** option is selected in the **Settings** menu:

Edit Attributes

Opens a small in-place editor that allow you to edit the attributes of the selected node.

Cut

Cuts the currently selected node.

Copy

Copies the currently selected node.

Delete

Deletes the currently selected node.

Search References Ctrl + Shift + R (Command + Shift + R on OS X)

Searches all references of the item found at current cursor position in the defined scope, if any. See *Finding XSLT References and Declarations* for more details.

Search References in

Searches all references of the item found at current cursor position in the specified scope. See *Finding XSLT References and Declarations* for more details.

Component Dependencies

Opens the ***Component Dependencies*** *view* that allows you to see the dependencies for the current selected component.

Resource Hierarchy

Opens the ***Resource Hierarchy/Dependencies*** *view* that displays the hierarchy for the currently selected resource.

Resource Dependencies

Opens the ***Resource Hierarchy/Dependencies*** *view* that displays the dependencies of the currently selected resource.

Rename Component in

Renames the selected component. See *XSLT Refactoring Actions* for more details.

The following contextual menu actions are available in the **Outline** view when the **Show XML structure** option is selected in the **Settings** menu:

Append Child

Displays a list of elements that you can insert as children of the current element.

Insert Before

Displays a list of elements that you can insert as siblings of the current element, before the current element.

Insert After

Displays a list of elements that you can insert as siblings of the current element, after the current element.

Edit Attributes

Opens a small in-place editor that allow you to edit the attributes of the selected node.

Toggle Comment

Comments/uncomments the currently selected element.

Search references

Searches for the references of the currently selected component.

Search references in

Searches for the references of the currently selected component in the context of a scope that you define.

Component dependencies

Opens the ***Component Dependencies*** *view* that displays the dependencies of the currently selected component.

Rename Component in

Renames the currently selected component in the context of a scope that you define.

Cut

Cuts the currently selected component.

Copy

Copies the currently selected component.

Delete

Deletes the currently selected component.

Expand More

Expands the structure of a component in the **Outline** view.

Collapse All

Collapses the structure of all the component in the **Outline** view.

The stylesheet components information is presented on two columns: the first column presents the `name` and `match` attributes, the second column the `mode` attribute. If you know the component name, match or mode, you

can search it in the **Outline** view by typing one of these pieces of information in the filter text field from the top of the view or directly on the tree structure. When you type de component name, match or mode in the text field, you can switch to the tree structure using:

- Keyboard arrow keys
- **Enter** key
- **Tab** key
- **Shift-Tab** key combination

To switch from tree structure to the filter text field, you can use **Tab** and **Shift-Tab**.

Tip: The search filter is case insensitive. The following wildcards are accepted:

- ***** - any string
- **?** - any character
- **,** - patterns separator

If no wildcards are specified, the string to search is used as a partial match.

The content of the **Outline** view and the editing area are synchronized. When you select a component in the **Outline** view, its definition is highlighted in the editing area.

Oxygen XML Editor allows you to sort the components of the tree in the **Outline** view.

Note: Sorting groups in the **Outline** view is not supported.

Oxygen XML Editor has a predefined order of the groups in the **Outline** view:

- For location, the names of the files are sorted alphabetically. The file you are editing is located at the top of the list.
- For type, the order is: parameters, variables, templates, functions, set attributes, character-map.

  Note: When no grouping is available and the table is not sorted, Oxygen XML Editor sorts the components depending on their order in the document. Oxygen XML Editor also takes into account the name of the file that the components are part of.

XSLT/XQuery Input View

The structure of the XML document associated to the edited XSLT stylesheet, or the structure of the source documents of the edited XQuery is displayed in a tree form in a view called the **XSLT/XQuery Input** view. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu. The tree nodes represent the elements of the documents.

XSLT

If you click a node in the **XSLT/XQuery Input** view, the corresponding template from the stylesheet is highlighted. A node can be dragged from this view and dropped in the editor area for quickly inserting `xsl:template`, `xsl:for-each`, or other XSLT elements that have the `match/select/test` attribute already completed. The value of the attribute is the correct XPath expression that refers to the dragged tree node. This value is based on the current editing context of the drop spot.

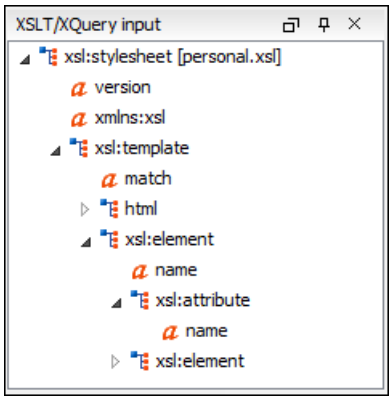


Figure 274: XSLT Input View

XSLT Example:

For the following XML document:

```
<personnel>
    <person id="Big.Boss">
        <name>
            <family>Boss</family>
            <given>Big</given>
        </name>
        <email>chief@oxygenxml.com</email>
        <link subordinates="one.worker"/>
    </person>
    <person id="one.worker">
        <name>
            <family>Worker</family>
            <given>One</given>
        </name>
        <email>one@oxygenxml.com</email>
        <link manager="Big.Boss"/>
    </person>
</personnel>
```

and the following XSLT stylesheet:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        version="2.0">
    <xsl:template match="personnel">
        <xsl:for-each select="*">

        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

if you drag the `given` element and drop it inside the `xsl:for-each` element, the following pop-up menu is displayed:

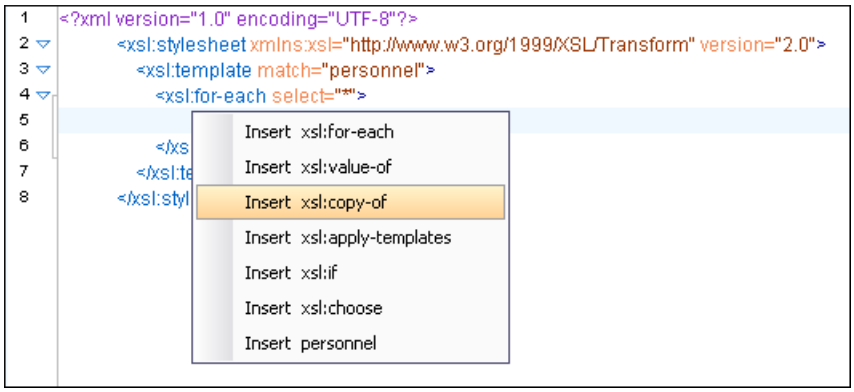


Figure 275: XSLT Input Drag and Drop Pop-up Menu

If you select **Insert xsl:copy-of** (for example), the resulting document will look like this:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
    <xsl:template match="personnel">
        <xsl:for-each select="*">
            <xsl:copy-of  select="name/given"/>
        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

Figure 276: XSLT Input Drag and Drop Result

XQuery

You can also use the **XSLT/XQuery Input** view to drag and drop a node into the editing area to quickly insert XQuery expressions.

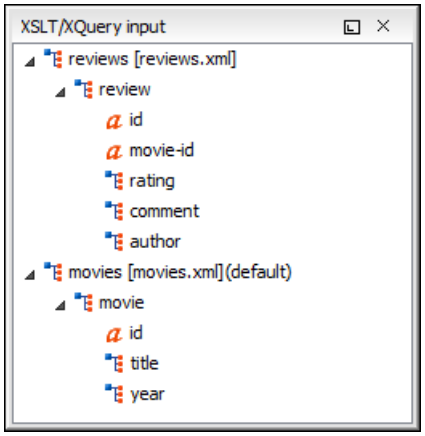


Figure 277: XQuery Input View

XQuery Example:

For the following XML documents:

```
<movies>
<movie id="1">
<title>The Green Mile</title>
<year>1999</year>
</movie>
<movie id="2">
<title>Taxi Driver</title>
<year>1976</year>
</movie>
</movies>
```

```
<reviews>
<review id="100" movie-id="1">
<rating>5</rating>
<comment>It is made after a great Stephen King book.
</comment>
<author>Paul</author>
</review>
<review id="101" movie-id="1">
<rating>3</rating>
<comment>Tom Hanks does a really nice acting.</comment>
<author>Beatrice</author>
</review>
<review id="104" movie-id="2">
<rating>4</rating>
<comment>Robert De Niro is my favorite actor.</comment>
<author>Maria</author>
</review>
</reviews>
```

and the following XQuery:

```
let $review := doc("reviews.xml")
for $movie in doc("movies.xml")/movies/movie
let $movie-id := $movie/@id
return
```

```
                    <movie id="{$movie/@id}">
                    {$movie/title}
                    {$movie/year}
                    <maxRating>
                    {

                    }
                    </maxRating>
                    </movie>
```

If you drag the **review** element and drop it between the braces, the following pop-up menu is displayed:

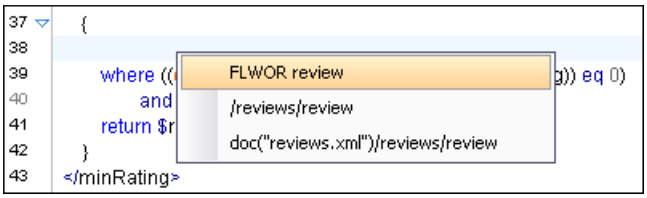


Figure 278: XQuery Input Drag and Drop Pop-up Menu

Select **FLWOR review**, the resulting document will look like this:

```
37        {
38            for $review in doc("reviews.xml")/reviews/review
39            return
40            where ((compare($rev/rating/text(), string($minRating)) eq 0)
41                    and ($rev/@movie-id = $movie/@id))
42            return $rev/author
43        }
```

Figure 279: XQuery Input Drag and Drop Result

XSLT Resource Hierarchy/Dependencies View

The **Resource Hierarchy/Dependencies** view allows you to see the hierarchy/dependencies for a stylesheet. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

If you want to see the hierarchy of a stylesheet, select the desired stylesheet in the ***Project*** *view* and choose **Resource Hierarchy** from the contextual menu.

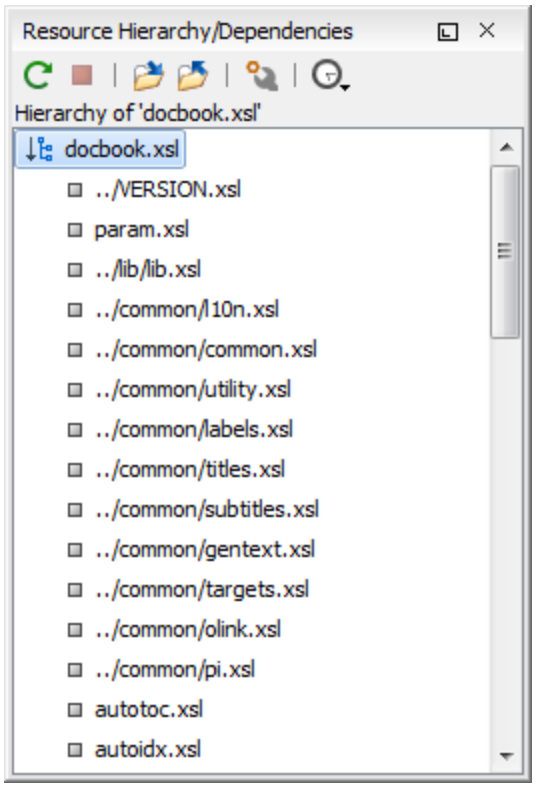


Figure 280: Resource Hierarchy/Dependencies View - Hierarchy for `docbook.xsl`

If you want to see the dependencies of a stylesheet, select the desired stylesheet in the ***Project*** *view* and choose **Resource Dependencies** from the contextual menu.

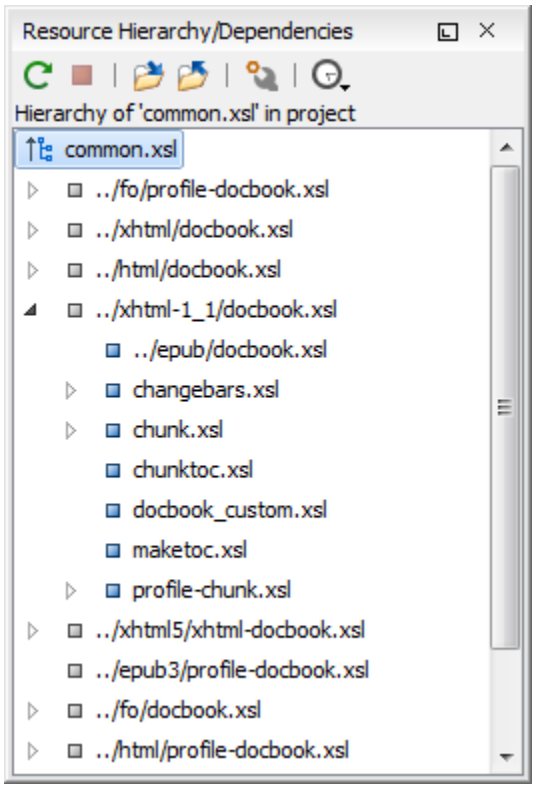


Figure 281: Resource Hierarchy/Dependencies View - Dependencies for `common.xsl`

The following actions are available in the **Resource Hierarchy/Dependencies** view:

Refresh

Refreshes the Hierarchy/Dependencies structure.

Stop

Stops the hierarchy/dependencies computing.

Show Hierarchy

Allows you to choose a resource to compute the hierarchy structure.

Show Dependencies

Allows you to choose a resource to compute the dependencies structure.

Configure

Allows you to configure a scope to compute the dependencies structure. There is also an option for automatically using the defined scope for future operations.

History

Provides access to the list of previously computed dependencies. Use the **Clear history** button to remove all items from this list.

The contextual menu contains the following actions:

Open

Opens the resource. You can also double-click a resource in the Hierarchy/Dependencies structure to open it.

Copy location

Copies the location of the resource.

Move resource

Moves the selected resource.

Rename resource

Renames the selected resource.

Show Resource Hierarchy

Shows the hierarchy for the selected resource.

Show Resource Dependencies

Shows the dependencies for the selected resource.

Add to Master Files

Adds the currently selected resource in *the **Master Files** directory*.

Expand All

Expands all the children of the selected resource from the Hierarchy/Dependencies structure.

Collapse All

Collapses all children of the selected resource from the Hierarchy/Dependencies structure.

Tip: When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon .

Related Information:

Working with Modular XML Files in the Master Files Context on page 481
Search and Refactor Operations Scope on page 480

Moving/Renaming XSLT Resources

You can move and rename a resource presented in the **Resource/Hierarchy Dependencies** view, using the **Rename resource** and **Move resource** refactoring actions from the contextual menu.

When you select the **Rename** action in the contextual menu of the **Resource/Hierarchy Dependencies** view, the **Rename resource** dialog box is displayed. The following fields are available:

- **New name** - Presents the current name of the edited resource and allows you to modify it.

- **Update references** - Select this option to update the references to the resource you are renaming.

When you select the **Move** action from the contextual menu of the **Resource/Hierarchy Dependencies** view, the **Move resource** dialog box is displayed. The following fields are available:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, in accordance with the new location and name.

If the **Update references of the moved resource(s)** option is selected, a **Preview** option (which opens the **Preview** dialog box) is available for both actions. The **Preview** dialog box presents a list with the resources that are updated.

XSLT Component Dependencies View

The **Component Dependencies** view allows you to see the dependencies for a selected XSLT component. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

If you want to see the dependencies of an XSLT component, select the desired component in the editor and choose the **Component Dependencies** action from the contextual menu. The action is available for all named components (templates, variables, parameters, attribute sets, keys, functions, outputs).

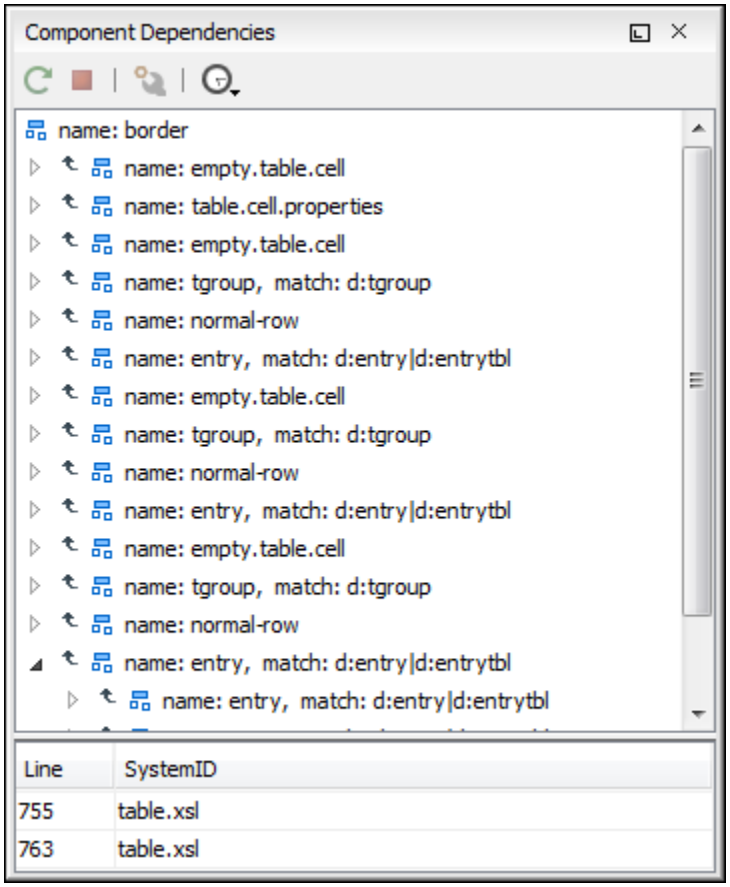


Figure 282: Component Dependencies View - Hierarchy for table.xsl

In the Component Dependencies view you have several actions in the toolbar:

Refresh

Refreshes the dependencies structure.

Stop

Stops the dependencies computing.

Configure

Allows you to configure a search scope to compute the dependencies structure. You can decide to use automatically the defined scope for future operations by selecting the corresponding checkbox.

History

Allows you to repeat a previous dependencies computation.

The following actions are available on the contextual menu:

Go to First Reference

Selects the first reference of the referenced component from the current selected component in the dependencies tree.

Go to Component

Shows the definition of the current selected component in the dependencies tree.

Tip: If a component contains multiple references to another, a small table is displayed that contains all references. When a recursive reference is encountered, it is marked with a special icon .

Related Information:

Search and Refactor Operations Scope on page 480

Highlight Component Occurrences

When a component (for example variable or named template) is found at current cursor position, Oxygen XML Editor performs a search over the entire document to find the component declaration and all its references. When found, they are highlighted both in the document and in the stripe bar, at the right side of the document.

Note: Oxygen XML Editor also supports occurrences highlight for template modes.

Customizable colors are used: one for the component definition and another one for component references. Occurrences are displayed until another component is selected and a new search is performed. All occurrences are removed when you start to edit the document.

This feature is enabled by default. To configure it, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Mark Occurrences**. A search can also be triggered with the **Search** > **Search Occurrences in File (Ctrl + Shift + U (Command + Shift + U on OS X))** contextual menu action. Matches are displayed in separate tabs of the ***Results** view*.

Finding XSLT References and Declarations

The following search actions related with XSLT references and declarations are available from the **Search** submenu of the contextual menu and from the **Document** > **References** menu:

- **Search References** - Searches all references of the item found at current cursor position in the defined scope, if any. If a scope is defined but the currently edited resource is not part of the range of determined resources, a warning dialog box is displayed that allows you to define another search scope.
- **Search References in** - Searches all references of the item found at current cursor position in the file or files that you specify when a scope is defined.
- **Search Declarations** - Searches all declarations of the item found at current cursor position in the defined scope, if any. If a scope is defined but the current edited resource is not part of the range of resources determined by this scope, a warning dialog box is displayed that allows you to define another search scope.
- **Search Declarations in** - Searches all declarations of the item found at current cursor position in the file or files that you specify when a scope is defined.
- **Search Occurrences in File** - Searches all occurrences of the item at the cursor position in the currently edited file.

The following action is available from the contextual menu and the **Document** > **Schema** menu:

- **Show Definition** - Moves the cursor to the location of the definition of the current item.

  Note: You can also use the **Ctrl + Single-Click (Command + Single-Click on OS X)** shortcut on a reference to display its definition.

Related Information:

Search and Refactor Operations Scope on page 480

XSLT Stylesheet Component Documentation Support

Oxygen XML Editor offers built-in support for documenting XSLT stylesheets. If the expanded *QName* of the element has a non-null namespace URI, the `xsl:stylesheet` element may contain any element not from the XSLT namespace. Such elements are referenced as user-defined data elements. Such elements can contain the documentation for the stylesheet and its elements (top-level elements whose names are in the XSLT namespace). Oxygen XML Editor offers its own XML schema that defines such documentation elements. The schema is named `stylesheet_documentation.xsd` and can be found in *[OXYGEN_INSTALL_DIR]*`/frameworks/stylesheet_documentation`. The user can also specify a custom schema in *XSL Content Completion options*.

When content completion is invoked inside an XSLT editor by pressing **Ctrl + Space (Command + Space on OS X)**, it offers elements from the XSLT documentation schema (either the built-in one or one specified by user).

In **Text** mode, to add documentation blocks, press **Ctrl + Alt + D (Command + Alt + D on OS X)** or select **Add component documentation** from the contextual menu.

In **Author** mode, the following stylesheet documentation actions are available in the contextual menu, **Component Documentation** submenu:

- **Add component documentation** - Adds documentation blocks for the component at cursor position.
- **Paragraph** - Inserts a new documentation paragraph.
- **Bold** - Makes the selected documentation text bold.
- **Italic** - Makes the selected documentation text italic.
- **List** - Inserts a new list.
- **List Item** - Inserts a list item.
- **Reference** - Inserts a documentation reference.

If the cursor is positioned inside the `xsl:stylesheet` element context, documentation blocks are generated for all XSLT elements. If the cursor is positioned inside a specific XSLT element (such as a template or function), a documentation block is generated for that element only.

Example: A documentation block using Oxygen XML Editor built-in schema

```
<xd:doc>
  <xd:desc>
    <xd:p>Search inside parameter <xd:i>string</xd:i>
        for the last occurrence of parameter
    <xd:i>searched</xd:i>. The substring starting from the 0 position
        to the identified last occurrence will be returned.
    <xd:ref name="f:substring-after-last" type="function"
        xmlns:f="http://www.oxygenxml.com/doc/xsl/functions">See also
    </xd:ref>
    </xd:p>
  </xd:desc>
  <xd:param name="string">
    <xd:p>String to be analyzed</xd:p>
    </xd:param>
  <xd:param name="searched">
    <xd:p>Marker string. Its last occurrence will be identified</xd:p>
    </xd:param>
  <xd:return>
    <xd:p>A substring starting from the beginning of <xd:i>string</xd:i>
        to the last occurrence of <xd:i>searched</xd:i>.
        If no occurrence is found an empty string will be returned.
    </xd:p>
  </xd:return>
</xd:doc>
```

Related Information:

Generating Documentation for an XSLT Stylesheet on page 536

XSLT Refactoring Actions

Oxygen XML Editor offers a set of actions that allow you to change the structure of an XSLT stylesheet without changing the results of running it in an XSLT transformation. Depending on the selected text, the following XSLT refactoring actions are available from the **Refactoring** submenu of the contextual menu (or from the **Document** > **Refactoring** menu):

- **Extract template** (Active only when the selection contains well-formed elements) - Extracts the selected XSLT instructions sequence into a new template. It opens a dialog box that allows you to specify the name of the new template to be created. The possible changes to perform on the document can be previewed before altering the document. After pressing OK, the template is created and the selection is replaced with a `<xsl:call-template>` instruction referencing the newly created template.

  Note: The newly created template is indented and its name is highlighted in the `<xsl:call-template>` element.
- **Extract function** - Extracts the selected XSLT instructions sequence into a new function. It opens a dialog box that allows you to specify the name of the new function. It then moves the selected lines to a newly created XSLT function and inserts a function call in the place of the selected lines. You can also use parts of an XPath expression to create the new functions.
- **Create local variable** - Creates an XSLT variable, wrapped around the selection. It opens a dialog box that allows you to specify the name of the new variable. It then wraps the selection in the variable and you can reference it at anytime in the code.
- **Move to another stylesheet** (Active only when entire components are selected) - Allows you to move one or more XSLT global components (templates, functions, or parameters) to another stylesheet. It opens a dialog box that allows you to specify where the selected components will be moved to. Follow these steps when using the dialog box:

  1. Choose whether you want to move the selected components to a new stylesheet or an existing one.
  2. If you choose to move the components to an existing one, select the destination stylesheet. Press the **Choose** button to select the destination stylesheet file. Oxygen XML Editor will automatically check if the destination stylesheet is already contained by the hierarchy of the current stylesheet. If it is not contained, choose whether or not the destination stylesheet will be referenced (imported or included) from the current stylesheet. The following options are available:
     - **Include** - The current stylesheet will use an `xsl:include` instruction to reference the destination stylesheet.
     - **Import** - The current stylesheet will use an `xsl:import` instruction to reference the destination stylesheet.
     - **None** - There will be created no relation between the current and destination stylesheets.
  3. Press the **Move** button to move the components to the destination. The moved components are highlighted in the destination stylesheet.
- **Convert attributes to xsl:attributes** - Converts the attributes from the selected element and represents each of them with an `<xsl:attribute>` instruction. For example, the following element:

  ```
  <person id="Big{test}Boss"/>
  ```

  is converted to:

  ```
  <person>
      <xsl:attribute name="id">
          <xsl:text>Big</xsl:text>
          <xsl:value-of select="test"/>
          <xsl:text>Boss</xsl:text>
      </xsl:attribute>
  </person>
  ```
- **Convert xsl:if into xsl:choose/xsl:when** - Converts one or more `xsl:if` element blocks into one or more `xsl:when` blocks surrounded by an `xsl:choose` element. If it is invoked on a selection, the selection must contain a well-formed fragment. If there is no selection, the `xsl:if` element that surrounds the content at the current cursor position is converted.

For example, the following block:

```
<xsl:if test="a">
  <!-- XSLT code -->
</xsl:if>
```

is converted to:

```
<xsl:choose>
    <xsl:when test="a">
        <!-- XSLT code -->
    </xsl:when>
    <xsl:otherwise>
        |
    </xsl:otherwise>
</xsl:choose>
```

(where the | character is the current cursor position)

- **Convert xsl:choose/xsl:when into xsl:if** - Converts each `xsl:when` block into an `xsl:if` block. For the *otherwise* branch, it also adds an *and* statement to each negated form of the conditions. For example, the following block:

```
<xsl:choose>
    <xsl:when test="c1">
        <!-- XSLT statement 1 -->
    </xsl:when>
    <xsl:when test="c2">
        <!-- XSLT statement 2 -->
    </xsl:when>
    <xsl:when test="c3">
        <!-- XSLT statement 3 -->
    </xsl:when>
    <xsl:otherwise>
        <!-- XSLT "otherwise" statement-->
    </xsl:otherwise>
</xsl:choose>
```

is converted to:

```
<xsl:if test="c1">
    <!-- XSLT statement 1 -->
</xsl:if>
<xsl:if test="c2">
    <!-- XSLT statement 2 -->
</xsl:if>
<xsl:if test="c3">
    <!-- XSLT statement 3 -->
</xsl:if>
<xsl:if test="not(c1) and not(c2) and not(c3)">
    <!-- XSLT "otherwise" statement-->
</xsl:if>
```

- **Extract local variable** (Active on a selection made inside an attribute that contains an XPath expression) - Allows you to create a new local variable by extracting the selected XPath expression. After creating the new local variable before the current element, Oxygen XML Editor allows you to edit the name of the variable.
- **Extract global variable** (Active on a selection made inside an attribute that contains an XPath expression) - Allows you to create a new global variable by extracting the selected XPath expression. After creating the new global variable, Oxygen XML Editor allows you to edit the name of the variable.

  Note: Oxygen XML Editor checks if the selected expression depends on local variables or parameters that are not available in the global context where the new variable is created.
- **Extract template parameter** (Active on a selection made inside an attribute that contains an XPath expression) - Allows you to create a new template parameter by extracting the selected XPath expression. After creating the new parameter, Oxygen XML Editor allows you to edit the name of the parameter.
- **Extract global parameter** (Active on a selection made inside an attribute that contains an XPath expression) - Allows you to create a new global parameter by extracting the selected XPath expression. After creating the new parameter, Oxygen XML Editor allows you to edit the name of the parameter.

  Note: Oxygen XML Editor checks if the selected expression depends on local variables or parameters that are not available in the global context where the new parameter is created.
- **Rename Component** - Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at

the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the **Esc** or **Enter** key on your keyboard.

- **Rename Component in** - Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the **Preview** button, you can view the files to be affected by the action.

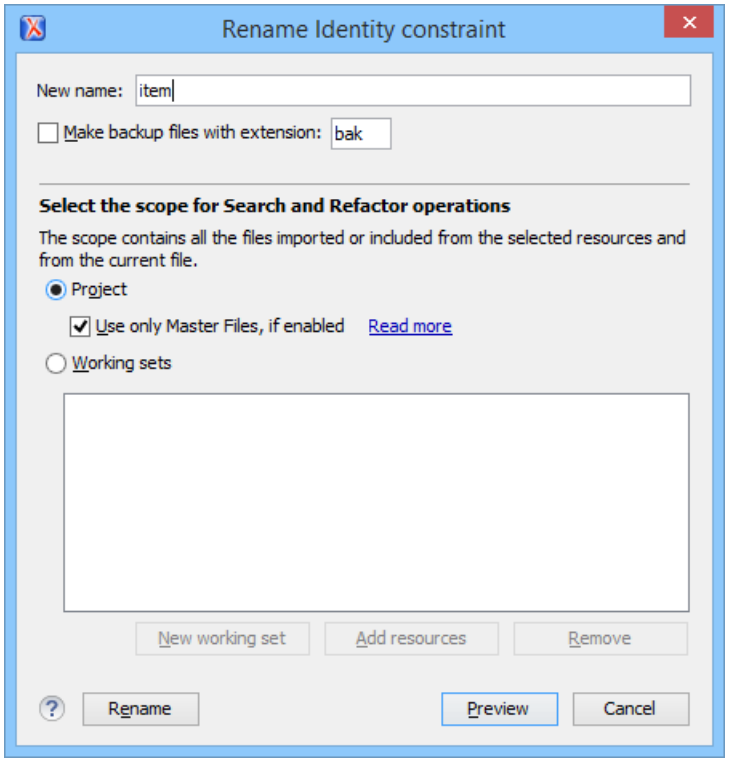


Figure 283: Rename Identity Constraint Dialog Box

Note: Many of these refactoring actions are also proposed by the *Quick Assist support*.

To watch our video demonstration about XSLT refactoring, go to *https://www.oxygenxml.com/demo/XSL_Refactoring.html*.

XSLT Quick Assist Support

The *Quick Assist support* helps you to rapidly access search and refactoring actions. If one or more actions are available in the current context, they are accessible via a yellow bulb help ( ) placed at the current line in the stripe on the left side of the editor. Also, you can invoke the *Quick Assist* menu by using the **Alt + 1** (**Meta + Alt + 1** on Mac OS X) keyboard shortcuts.

Two categories of actions are available in the *Quick Assist* menu:

- Actions available on a selection made inside an attribute that contains an XPath expression:

  Extract template

  Extracts the selected XSLT instructions sequence into a new template.

  Move to another stylesheet

  Allows you to move one or more XSLT global components (templates, functions, or parameters) to another stylesheet.

  Extract local variable

  Allows you to create a new local variable by extracting the selected XPath expression.

  Extract global variable

  Allows you to create a new global variable by extracting the selected XPath expression.

Extract template parameter

Allows you to create a new template parameter by extracting the selected XPath expression.

Extract global parameter

Allows you to create a new global parameter by extracting the selected XPath expression.

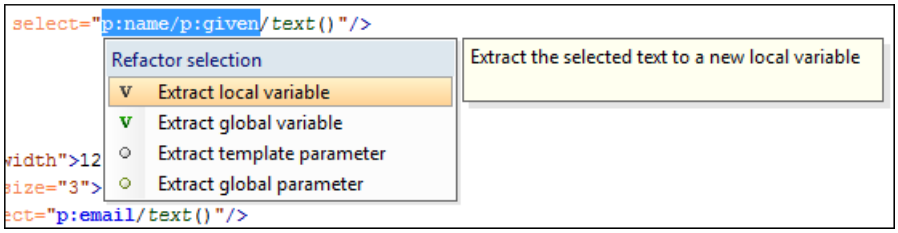


Figure 284: XSLT Quick Assist Support - Refactoring Actions

- Actions available when the cursor is positioned over the name of a component:

Rename Component in

Renames the component and all its dependencies.

Search Declarations

Searches the declaration of the component in a predefined scope. It is available only when the context represents a component name reference.

Search References

Searches all references of the component in a predefined scope.

Component Dependencies

Searches the component dependencies in a predefined scope.

Change Scope

Configures the scope that will be used for future search or refactor operations.

Rename Component

Allows you to rename the current component in-place.

Search Occurrences

Searches all occurrences of the component within the current file.

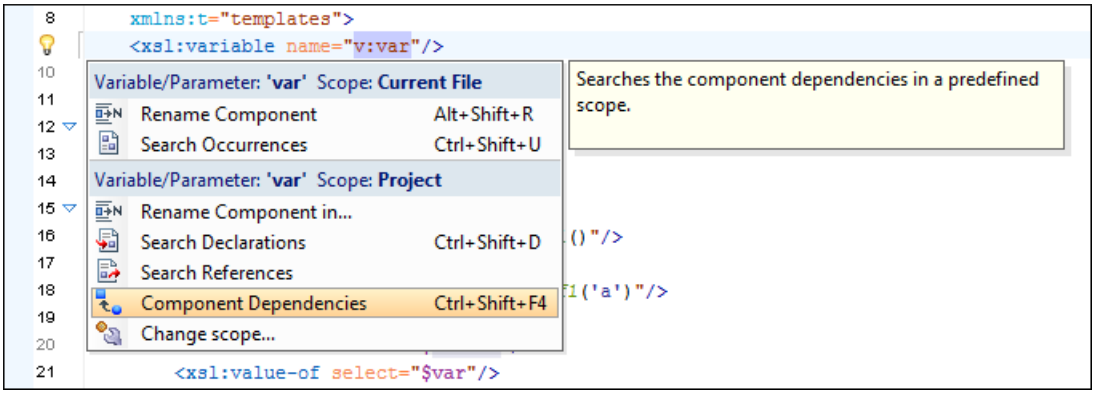


Figure 285: XSLT Quick Assist Support - Component Actions

Related Information:

Component Dependencies View on page 528
XSLT Hierarchy View on page 525
XSLT Refactoring Actions on page 531

Search and Refactor Operations Scope on page 480

XSLT Unit Test (XSpec)

XSpec is a behavior driven development (BDD) *framework* for XSLT and XQuery. XSpec consists of a syntax for describing the behavior of your XSLT or XQuery code, and some code that enables you to test your code against those descriptions.

Creating an XSLT Unit Test

To create an XSLT Unit Test, go to **File** > **New** > **XSLT Unit Test**. You can also create an XSLT Unit Test from the contextual menu of an XSL file in the ***Project*** *view*. Oxygen XML Editor allows you to customize the XSpec document when you create it. In the customization dialog box, you can enter the path to an XSL document or to a master XSL document.

When you create an XSpec document based on an XSL document, Oxygen XML Editor uses information from the validation and transformation scenarios associated with the XSL file. From the transformation scenario Oxygen XML Editor uses extensions and properties of Saxon 9.7.0.15, improving the Ant scenario associated with the XSpec document.

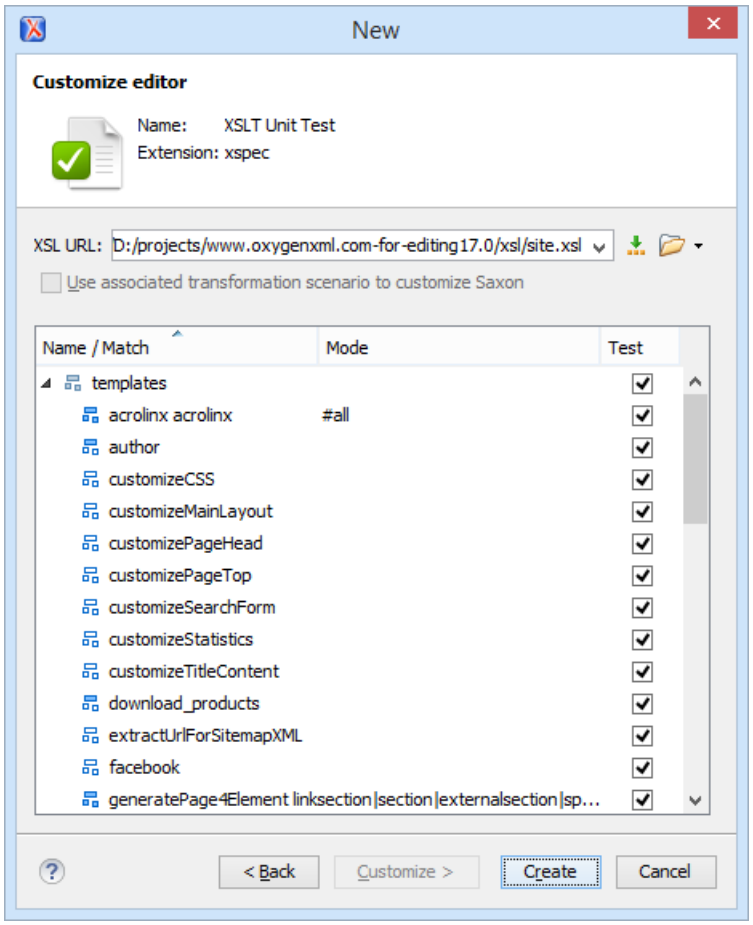


Figure 286: New XSLT Unit Test Wizard

Running an XSLT Unit Test

To run an XSLT Unit Test, open the XSpec file in an editor and click **Apply Transformation Scenario(s)** on the main toolbar. This will run the built-in **XSpec for XSLT** transformation scenario that is defined in the XSpec *framework*.

Testing a Stylesheet

An XSpec file contains one or more test scenarios. You can test a stylesheet in one of the following ways:

- Test an entire stylesheet.

  Testing is performed in a certain context. You can define a context as follows:

  - Inline context, building the test based on a string.

    ```
    <x:scenario label="when processing a para element">
        <x:context>
            <para>...</para>
        </x:context>
        ...
    </x:scenario>
    ```

  - Based on an external file, or on a part of an external file extracted with an XPath expression.

    ```
    <x:scenario label="when processing a para element">
        <x:context href="source/test.xml" select="/doc/body/p[1]" />
        ...
    </x:scenario>
    ```

- Test a function.

  ```
  <x:scenario label="when capitalising a string">
    <x:call function="eg:capital-case">
      <x:param select="'an example string'" />
      <x:param select="true()" />
    </x:call>
    ...
  </x:scenario>
  ```

- Test a template with a name.

  ```
  <x:scenario label="when creating a table">
    <x:call template="createTable">
      <x:param name="nodes">
        <value>A</value>
        <value>B</value>
      </x:param>
      <x:param name="cols" select="2" />
    </x:call>
  </x:scenario>
  ```

You can reference test files between each other, which allows you to define a suite of tests. For further details about test scenarios, go to *https://github.com/expath/xspec/wiki/Writing-Scenarios*.

Adding a Catalog to an XSpec Transformation

If your XSLT needs a catalog, you can add one to the XSpec transformation by doing one of the following:

- If you are using a *project* in Oxygen XML Editor, create a `catalog.xml` file in the project directory. This catalog will then be loaded automatically.
- *Edit* the **XSpec for XSLT** transformation scenario, go to the ***Parameters** tab*, and set the value of the `catalog` parameter to the location of your catalog file.

Generating Documentation for an XSLT Stylesheet

You can use Oxygen XML Editor to generate detailed documentation in HTML format for the elements (top-level elements whose names are in the XSLT namespace) of an XSLT stylesheet. You can select what XSLT elements to include in the generated documentation and also the level of details to present for each of them. The elements are hyperlinked. To generate documentation in a *custom output format*, you can edit the XSLT stylesheet used to generate the documentation, or create your own stylesheet.

To open the **XSLT Stylesheet Documentation** dialog box, select **XSLT Stylesheet Documentation** from the **Tools** > **Generate Documentation** menu or from the **Generate Documentation** submenu in the contextual menu of the ***Project** view*.

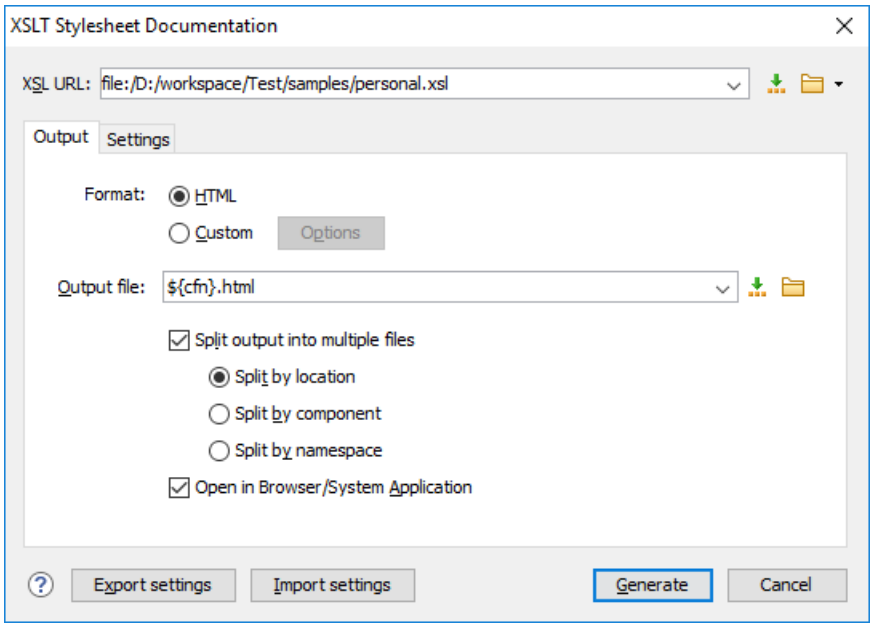


Figure 287: XSLT Stylesheet Documentation Dialog Box

The **XSL URL** field of the dialog box must contain the full path to the XSL Stylesheet file you want to generate documentation for. The stylesheet may be a local or a remote file. You can specify the path to the stylesheet by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.

Output Tab

The following options are available in the **Output** tab:

- **Format** - Allows you to choose between the following formats:
  - **HTML** - The documentation is generated in *HTML output format*.
  - **Custom** - The documentation is generated in a *custom output format*, allowing you to control the output. Click the **Options** button to open a **Custom format options** dialog box where you can specify a custom stylesheet for creating the output. There is also an option to **Copy additional resources to the output folder** and you can select the path to the additional **Resources** that you want to copy. You can also choose to keep the intermediate XML files created during the documentation process by deselecting the **Delete intermediate XML file** option.
- **Output file** - You can specify the path of the output file by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.
- **Split output into multiple files** - Instructs the application to split the output into multiple files. For large XSLT stylesheets, choosing another split criterion may generate smaller output files, providing faster documentation browsing. You can choose to split them by namespace, location, or component name.
- **Open in Browser/System Application** - Opens the result in the system application associated with the output file type.

  Note: To set the browser or system application that will be used, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.

Settings Tab

When you generate documentation for an XSLT stylesheet you can choose what XSLT elements to include in the output (templates, functions, global parameters, global variables, attribute sets, character maps, keys, decimal formats, output formats, XSLT elements from referenced stylesheets) and the details to include in the documentation.

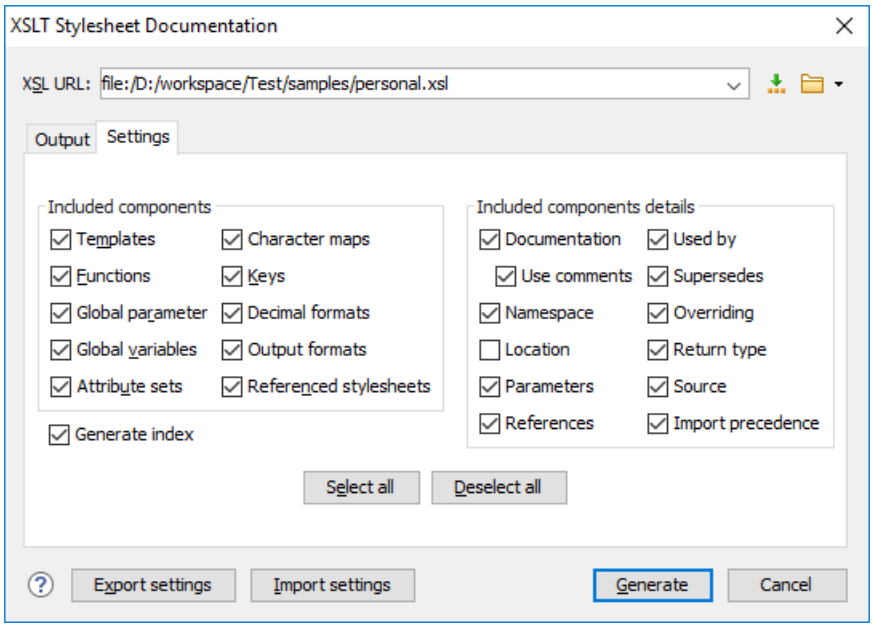


Figure 288: Settings Tab of the XSLT Stylesheet Documentation Dialog Box

The **Settings** tab allows you to choose whether or not to include the following components: **Templates**, **Functions**, **Global parameters**, **Global variables**, **Attribute sets**, **Character maps**, **Keys**, **Decimal formats**, **Output formats**, **Referenced stylesheets**.

You can choose whether or not to include the following other details:

- **Documentation** - Shows the documentation for each XSLT element. For HTML format, the user-defined data elements that are recognized and transformed in documentation blocks of the XSLT elements they precede, are the ones from the following schemas:
  - Oxygen XML Editor built-in XSLT documentation schema.
  - A subset of DocBook 5 elements. The recognized elements are: `section`, `sect1` to `sect5`, `emphasis`, `title`, `ulink`, `programlisting`, `para`, `orderedlist`, `itemizedlist`.
  - A subset of DITA elements. The recognized elements are: `concept`, `topic`, `task`, `codeblock`, p, b, `i`, `ul`, `ol`, `pre`, `sl`, `sli`, `step`, `steps`, `li`, `title`, `xref`.
  - Full XHTML 1.0 support.
  - XSLStyle documentation environment. XSLStyle uses DocBook or DITA languages inside its own user-defined data elements. The supported DocBook and DITA elements are the ones mentioned above.
  - DOXSL documentation *framework*. Supported elements are : `codefrag`, `description`, `para`, `docContent`, `documentation`, `parameter`, `function`, `docSchema`, `link`, `list`, `listitem`, `module`, `parameter`, `template`, `attribute-set`;

    Other XSLT documentation blocks that are not recognized will just be serialized inside an HTML `pre` element. You can change this behavior by using a *custom format* instead of the built-in *HTML format* and providing your own XSLT stylesheets.
- **Use comments** - Controls whether or not the comments that precede an XSLT element is treated as documentation for the element they precede. Comments that precede or succeed the `xsl:stylesheet` element, are treated as documentation for the whole stylesheet. Note that comments that precede an import or include directive are not collected as documentation for the imported/included module. Also, comments from within the body of the XSLT elements are not collected at all.
- **Namespace** - Shows the namespace for named XSLT elements.
- **Location** - Shows the stylesheet location for each XSLT element.
- **Parameters** - Shows parameters of templates and functions.
- **References** - Shows the named XSLT elements that are referenced from within an element.
- **Used by** - Shows the list of all the XSLT elements that reference the current named element.
- **Supersedes** - Shows the list of all the XSLT elements that are superseded the current element.

- **Overriding** - Shows the list of all the XSLT elements that override the current element.
- **Return type** - Shows the return type of the function.
- **Source** - Shows the text stylesheet source for each XSLT element.
- **Import precedence** - Shows the computed import precedence as declared in the XSL transformation specifications.
- **Generate index** - Creates an index with all the XSLT elements included in the documentation.

Export settings - Save the current settings in a settings file for further use (for example, with the exported settings file you can generate the same *documentation from the command-line interface*.)

Import settings - Reloads the settings from the exported file.

Generate - Use this button to generate the XSLT documentation.

Related Information:

XSLT Stylesheet Component Documentation Support on page 530

Generate XSLT Documentation in HTML Format

The XSLT documentation generated in HTML format is presented in a visual diagram style with various sections, hyperlinks, and options.

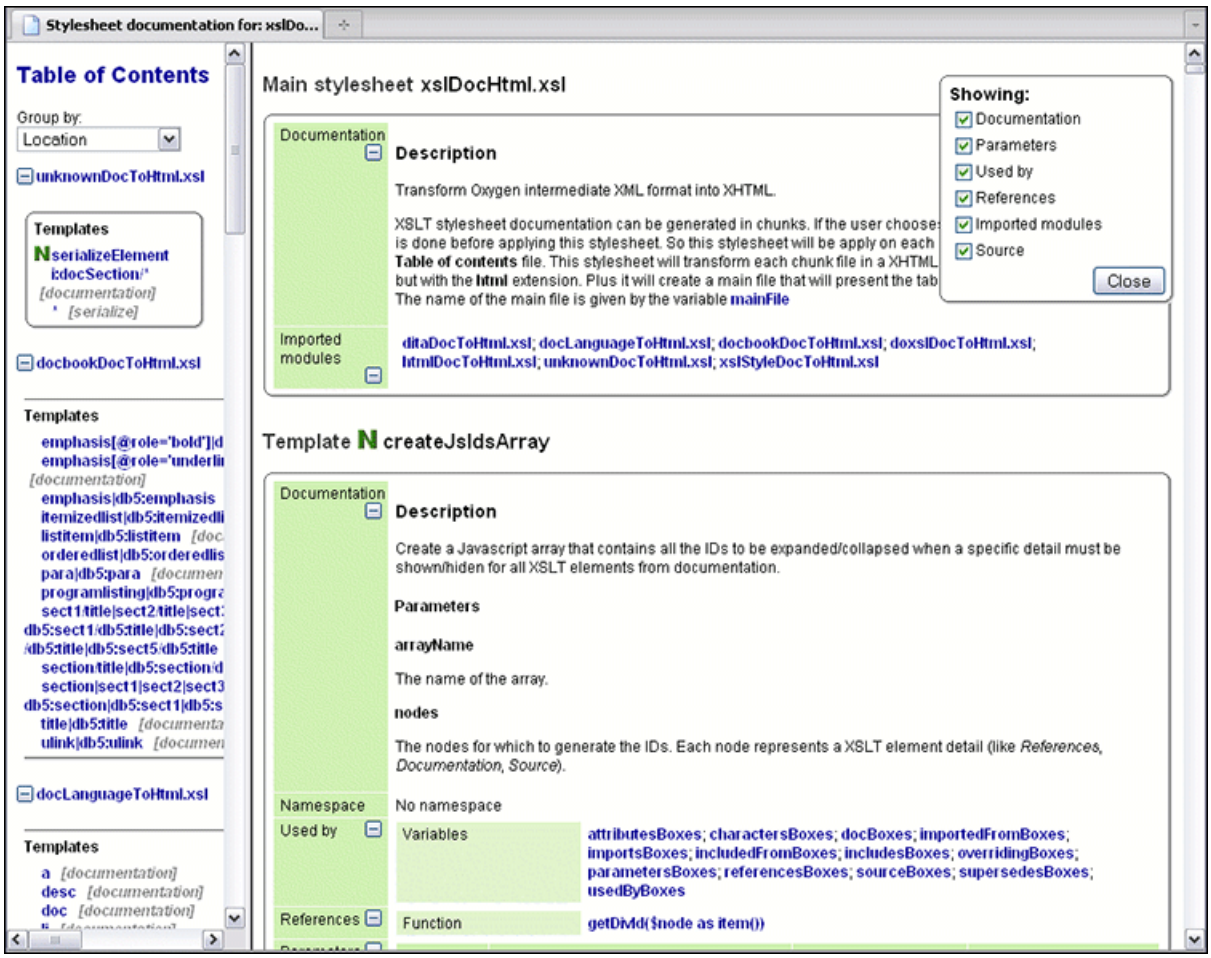


Figure 289: XSLT Stylesheet Documentation Example

The generated documentation includes the following:

- Table of Contents - You can group the contents by namespace, location, or component type. The XSLT elements from each group are sorted alphabetically (named templates are presented first and the `match` ones second).
- Information about main, imported, and included stylesheets. This information consists of:
  - XSLT modules included or imported by the current stylesheet.

- The XSLT stylesheets where the current stylesheet is imported or included.
- The stylesheet location.

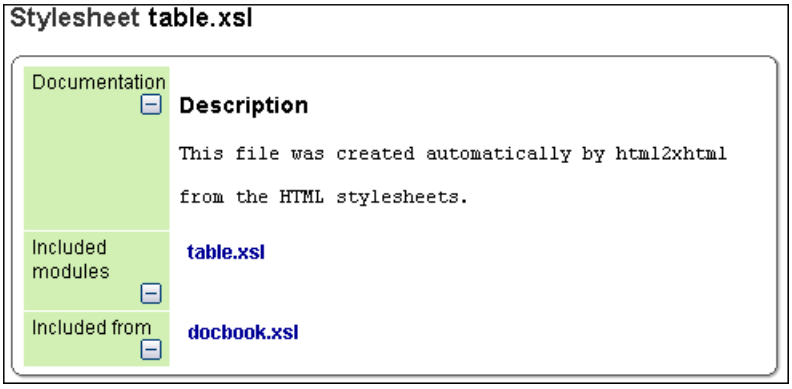


Figure 290: Information About an XSLT Stylesheet

If you choose to split the output into multiple files, the table of contents is displayed in the left frame. The contents are grouped using the same criteria as the split.

After the documentation is generated, you can collapse or expand details for some stylesheet XSLT elements by using the **Showing** options or the **Collapse** or **Expand** buttons.

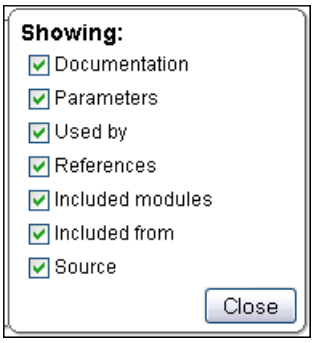


Figure 291: Showing Options

For each element included in the documentation, the section presents the element type followed by the element name (value of the `name` or `match` attribute for match templates).

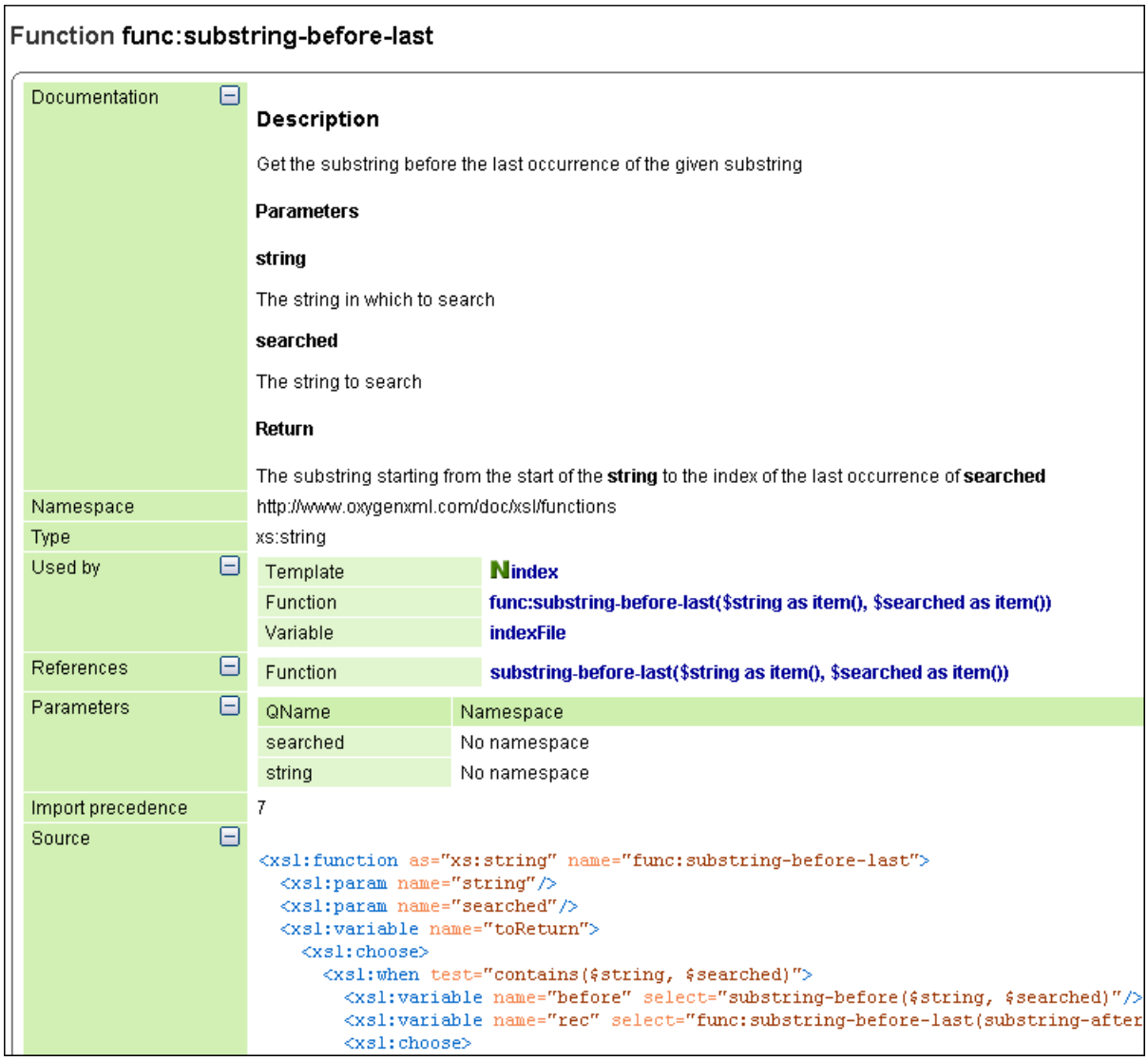


Figure 292: Documentation for an XSLT Element

Generate XSLT Documentation in a Custom Format

XSLT stylesheet documentation can be also generated in a custom format. You can choose the format from the *XSLT Stylesheet Documentation dialog box*. Specify your own stylesheet to transform the intermediary XML generated in the documentation process. You must write your stylesheet based on the schema `xslDocSchema.xsd` from *`[OXYGEN_INSTALL_DIR]`*`/frameworks/stylesheet_documentation`. You can create a custom format starting from one of the stylesheets used in the predefined HTML, PDF, and DocBook formats. These stylesheets are available in *`[OXYGEN_INSTALL_DIR]`*`/frameworks/stylesheet_documentation/xsl`.

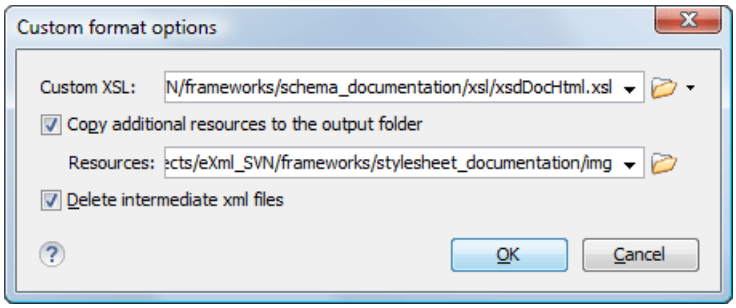


Figure 293: Custom Format Options Dialog Box

When using a custom format, you can also copy additional resources into the output folder or choose to keep the intermediate XML files created during the documentation process.

Generating XSLT Documentation From the Command-Line Interface

You can export the settings of the **XSLT Stylesheet Documentation** dialog box to an XML file by pressing the **Export settings** button. With the exported settings file, you can generate the same documentation from the command line by running the script `stylesheetDocumentation.bat` (on Windows) / `stylesheetDocumentation.sh` (on OS X / Unix / Linux) located in the Oxygen XML Editor installation folder. The script can be integrated in an external batch process launched from the command-line interface.

The command-line parameter of the script is the relative path to the exported XML settings file. The files that are specified with relative paths in the exported XML settings are resolved relative to the script directory.

Example:

```
<serialized>
    <map>
        <entry>
            <String xml:space="preserve">xsd.documentation.options</String>
            <xsdDocumentationOptions>
                <field name="outputFile">
                    <String xml:space="preserve">${cfn}.html</String>
                </field>
                <field name="splitMethod">
                    <Integer xml:space="preserve">1</Integer>
                </field>
                <field name="openOutputInBrowser">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="format">
                    <Integer xml:space="preserve">1</Integer>
                </field>
                <field name="customXSL">
                    <null/>
                </field>
                <field name="deleteXMLFiles">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeIndex">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeGlobalElements">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeGlobalAttributes">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeLocalElements">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeLocalAttributes">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeSimpleTypes">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeComplexTypes">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeGroups">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeAttributesGroups">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeRedefines">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeReferencedSchemas">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsDiagram">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsNamespace">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsLocation">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsType">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsTypeHierarchy">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsModel">
```

```
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsChildren">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsInstance">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsUsedby">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsProperties">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsFacets">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsAttributes">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsIdentityConstr">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsEscapeAnn">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsSource">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsAnnotations">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
            </xsdDocumentationOptions>
        </entry>
    </map>
</serialized>
```

Compiling an XSL Stylesheet for Saxon

As of Saxon 9.7, it is possible to export a compiled form of a stylesheet as an XML file (called a *stylesheet export file*). Oxygen XML Editor includes a simple tool called **Compile XSL Stylesheet for Saxon** that does this for you. A *stylesheet export file* is also needed if you want to use the Saxon-JS product to run transformations in a browser, as in the following example:

```
<script type="text/javascript" src="SaxonJS/SaxonJS.min.js"></script>
<script>
    window.onload = function() {
      SaxonJS.transform({
         stylesheetLocation: "books.sef",
         sourceLocation: "books.xml"
         });
    }
</script>
```

The **Compile XSL Stylesheet for Saxon** tool can be found in the **Tools** menu and it compiles a specified stylesheet as an XML file (*stylesheet export file* with a file extension of `.sef`).

Selecting this tool opens the **Compile XSL Stylesheet for Saxon** dialog box that allows you to configure some options for conversion.

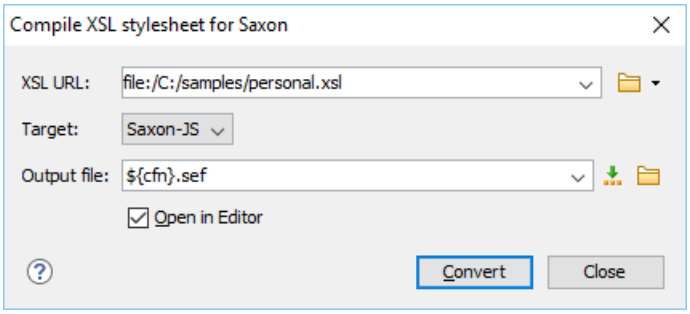


Figure 294: Compile XSL Stylesheet for Saxon Dialog Box

This dialog box includes the following options:

XSL URL

Allows you to select URL of the source XSL stylesheet. You can specify the URL by using the text field, the history drop-down, or the browsing tools in the **Browse** drop-down list.

Target

Allows you to select the type of Saxon product that the export file will be used with. You can choose **Saxon-JS**, **Saxon-EE**, **Saxon-PE**, or **Saxon-HE**.

Output file

You can specify the path where the output file will be saved by entering it in the text field, using the **Insert Editor Variables** button, or using the browsing tool button.

Open in Editor

Select this option to open the resulting *stylesheet export file* in the main Oxygen XML Editor editing pane.

Convert

Use this button to generate the *stylesheet export file* according the options selected in this dialog box.

Editing Ant Build Files

Oxygen XML Editor includes an Ant editor that provides a variety of specialized features to assist you with creating and editing Ant build files. The editor includes some specialized views, content completion assistance, automatic validation, syntax highlighting, *Quick Assist* and *Quick Fix* support, as well as numerous common editing and search features.

Related Information:

Editing XML Documents in Text Mode on page 290

Editing Ant Build Files in the Context of Master Files

Smaller interrelated modules that define a complex Ant build file cannot be correctly edited or validated individually due to their interdependency with other modules. For example, a *target* defined in a main build file is not visible when you edit an included or imported module. Oxygen XML Editor provides support for defining the main module (or modules), allowing you to edit any of the imported/included files in the context of the larger Ant build structure.

You cat set a main Ant build file either by using the *master files support from the **Project** view*, or *a validation scenario*.

To set a *master file* using a validation scenario, add validation units that point to the main modules. Oxygen XML Editor warns you if the current module is not part of the dependencies graph computed for the main build file. In this case, it considers the current module as the main build file.

The advantages of editing in the context of *master file* include:

- Correct validation of a module in the context of a larger build structure.
- *Content Completion Assistant* displays all components valid in the current context.
- The ***Outline** view* displays the components collected from the entire build file structure.

Validating Ant Build Files

Oxygen XML Editor performs the validation of Ant build files with the help of a built-in processor, which is largely based on the *Apache Ant* libraries. The path to these libraries can be configured in the ***Ant** preferences page*. The validation processor accesses the *parameters set in the associated Ant transformation scenario* and uses them as Ant properties when validating the current build script.

Oxygen XML Editor automatically validates Ant build files as you type. You can also validate the currently edited file by selecting the **Validate** action from the **Validation** toolbar drop-down menu or the **Document** > **Validate** menu.

Tip: To make a custom task available in the Ant validation engine, add a *JAR* file that contains the task implementation to the library directory of the built-in Ant distribution that comes bundled with Oxygen XML Editor (for example, `[OXYGEN_INSTALL_DIR]/tools/ant/lib` folder).

Create a Validation Scenario for Ant Build Files

If you want to customize the validation process for Ant build files, you can create a new validation scenario (or configure an existing one). For example, if you want to validate interrelated modules that define a complex Ant build file, you can specify the main Ant file by configuring a validation scenario. To create or configure a validation scenario, select **Configure Validation Scenario(s)** from the **Validation** toolbar drop-down menu or the **Document** > **Validate** menu.

Passing parameters to the Ant validation engine

Ant validation scenarios cannot be configured to accept custom Ant parameters. However, you can specify values for the parameters in your Ant document using an Ant transformation scenario:

1. Create a new *Ant transformation scenario*.
2. Edit the transformation scenario and *set all parameters* you need to pass to your Ant document.
3. Associate the new scenario with your Ant document (in the ***Configure Transformation Scenarios(s)** dialog box*). You do not need to run the transformation scenario. Every time a validation operation is triggered, the built-in validation engine uses the parameters set in the transformation scenario.

   Note: This behavior is available only for the validation scenarios that use the built-in validation engine. The custom defined validation engines do not benefit from this functionality.

Ant Quick Fix Support

The Oxygen XML Editor *Quick Fix support* helps you resolve missing target reference errors that may occur when developing Ant build documents.

To activate this feature, hover over or place the cursor in the highlighted area of text where a validation error or warning occurs. If a *Quick Fix* is available for that particular error or warning, you can access the *Quick Fix* proposals with any of the following methods:

- When hovering over the error or warning, the proposals are presented in a tooltip pop-up window.
- If you place the cursor in the highlighted area where a validation error or warning occurs, a *Quick Fix* icon ( ) is displayed in the stripe on the left side of the editor. If you click this icon, Oxygen XML Editor displays the list of available fixes.
- With the cursor placed in the highlighted area of the error or warning, you can also invoke the *Quick Fix* menu by pressing **Alt + 1 (Command + Alt + 1 on OS X)** on your keyboard.

Oxygen XML Editor provides the following types of *Quick Fixes* for Ant build files:

- **Create new target** - Creates a new target with the specified name.
- **Change reference to "targetName"** - Corrects the reference to point to an already defined target.
- **Remove target reference** - Removes the erroneous reference.

Content Completion in Ant Build Files

The list of proposals offered by the *Content Completion Assistant* in Ant build files are context-sensitive and includes proposals that are valid at the current cursor position. It can be manually activated with the **Ctrl + Space (Command + Space on OS X)** shortcut.

The *Content Completion Assistant* proposes various item types that are defined in the current Ant build and in the imported and included builds. The proposals include:

- Element names
- Attribute names
- Property names

  Note: In addition to the user-defined properties, the *Content Completion Assistant* offers the following values:

- The system properties set in the Java Virtual Machine.
- The built-in properties that Ant provides.
- Target names
- Task and type reference IDs

  Tip: To make a custom task available in the *Content Completion Assistant*, add a *JAR* file that contains the task implementation to the library directory of the built-in Ant distribution that comes bundled with Oxygen XML Editor (for example, `[OXYGEN_INSTALL_DIR]/tools/ant/lib` folder).

Note: For Ant resources, the proposals are collected starting from the *master files*. The *master files* can be defined in the project or in the associated validation scenario. For further details about the *Master Files* support go to *Defining Master Files at Project Level*.

Related Information:

Syntax Highlighting in Ant Files

Oxygen XML Editor supports syntax highlighting in Ant files to improve the readability of the content and reduce the time it takes to internalize the semantics of the structured content.

To customize the colors or styles used for the syntax highlighting colors for Ant build files, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)***.
2. Go to ***Editor*** > ***Syntax Highlight***.
3. Select and expand the **Ant Property** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

Related Information:

Syntax Highlight Preferences on page 104

Ant Outline View

The **Outline** view for Ant files displays the list of all the components (properties, targets, extension points, task/type definitions and references) from both the edited Ant build file and its imported and included modules. For Ant resources, the **Outline** view collects its components starting from the *master files*. The *master files* can be defined in the project and the main build file can be specified in a validation scenario. For more details, see *Defining Master Files at Project Level*.

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

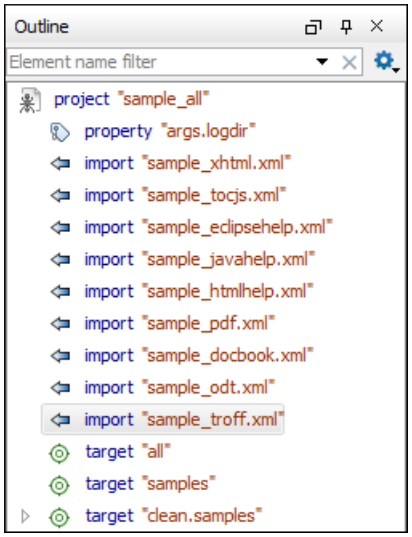


Figure 295: Ant Outline View

The following actions are available in the **Settings** menu on the Outline view toolbar:

Filter returns exact matches

The text filter of the **Outline** view returns only exact matches. By default, this filter is not selected.

Selection update on cursor move

Controls the synchronization between **Outline** view and source document. The selection in the **Outline** view can be synchronized with the cursor moves or the changes in the Ant editor. Selecting one of the components from the outline view also selects the corresponding item in the source document.

When the **Show components** option is selected, the following actions are available:

Show XML structure

Displays the XML document structure in a tree-like manner.

Sort

Sorts the components in the **Outline** view alphabetically.

Show all components

Displays all components that were collected starting from the *master file*. This option is set by default.

Show only local components

Displays the components defined in the current file only.

Group by location/type

The build file components can be grouped by location and type.

When the **Show XML structure** option is selected, the following actions are available:

Show components

Switches the **Outline** view to the components display mode.

Flat presentation mode of the filtered results

When active, the application flattens the filtered result elements to a single level.

Show comments and processing instructions

Show/hide comments and processing instructions in the **Outline** view.

Show element name

Show/hide element name.

Show text

Show/hide additional text content for the displayed elements.

Show attributes

Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from *the Outline preferences panel*.

Configure displayed attributes

Displays the ***XML Structured Outline*** *preferences page*.

The following actions are available in the contextual menu of the **Outline** view (when the **Show XML structure** option is selected in the **Settings** menu:

Append Child

Displays a list of elements that can be inserted as children of the current element.

Insert Before

Displays a list of elements that can be inserted as siblings of the current element, before the current element.

Insert After

Displays a list of elements that can be inserted as siblings of the current element, after the current element.

Edit Attributes

Displays an in-place attribute editing window.

Toggle Comment

Comments/uncomments the currently selected element.

Search References Ctrl + Shift + R (Command + Shift + R on OS X)

Searches all references of the item found at current cursor position in the defined scope. See *Find References and Declarations of Ant Components* for more details.

Search References in

Searches all references of the item found at current cursor position in the specified scope. See *Find References and Declarations of Ant Components* for more details.

Component Dependencies

Opens the *Ant **Component Dependencies** view* that allows you to see the dependencies for the current selected component.

Rename Component in

Renames the selected component. See *Ant Refactoring Actions* for more details.

Cut, Copy, Delete

Executes the typical editing actions on the currently selected component.

Expand More

Expands recursively all sub-components of the selected component.

Collapse All

Collapses recursively all sub-components of the selected component.

You can search a component in the **Outline** view by typing its name in the filter text field at the top of the view or directly on the tree structure. When you type the component name in the text field, you can switch to the tree structure using the following:

- **Down** arrow key
- **Tab** key
- **Shift-Tab** key combination

To switch from tree structure to the filter text field, you can use **Tab** and **Shift-Tab**.

Tip: The search filter is case insensitive. The following wildcards are accepted:

- ***** - any string
- **?** - any character
- **,** - patterns separator

If no wildcards are specified, the string to search is used as a partial match (such as ***textToFind***).

The content of the **Outline** view and the editing area are synchronized. When you select a component in the **Outline** view, its definition is highlighted in the editing area.

Oxygen XML Editor has a predefined order for the groups in the **Outline** view:

- For location, the names of the files are sorted alphabetically. The file you are editing is located at the top of the list.
- For type, the order is: properties, targets, references.

  Note: When no grouping is available Oxygen XML Editor sorts the components depending on their order in the document. Oxygen XML Editor also takes into account the name of the file that the components are part of.

Ant Resource Hierarchy/Dependencies View

The **Resource Hierarchy/Dependencies** view allows you to see the hierarchy/dependencies for an Ant build file by analyzing imported or included build files. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

If you want to see the hierarchy of a build file, select it in the ***Project*** *view* and choose **Resource Hierarchy** from the contextual menu.

If you want to see the dependencies of a build file, select it in the ***Project*** *view* and choose **Resource Dependencies** from the contextual menu.

The following actions are available in the **Resource Hierarchy/Dependencies** view:

Refresh

Refreshes the Hierarchy/Dependencies structure.

Stop

Stops the hierarchy/dependencies computing.

Show Hierarchy

Allows you to choose a resource to compute the hierarchy structure.

Show Dependencies

Allows you to choose a resource to compute the dependencies structure.

Configure

Allows you to configure a scope to compute the dependencies structure. There is also an option for automatically using the defined scope for future operations.

History

Provides access to the list of previously computed dependencies. Use the **Clear history** button to remove all items from this list.

The contextual menu contains the following actions:

Open

Opens the resource. You can also double-click a resource in the Hierarchy/Dependencies structure to open it.

Copy location

Copies the location of the resource.

Move resource

Moves the selected resource.

Rename resource

Renames the selected resource.

Show Resource Hierarchy

Shows the hierarchy for the selected resource.

Show Resource Dependencies

Shows the dependencies for the selected resource.

Add to Master Files

Adds the currently selected resource in *the* ***Master Files*** *directory*.

Expand All

Expands all the children of the selected resource from the Hierarchy/Dependencies structure.

Collapse All

Collapses all children of the selected resource from the Hierarchy/Dependencies structure.

Tip: When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon .

Ant Component Dependencies View

The **Component Dependencies** view allows you to see the dependencies for a selected Ant component. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

If you want to see the dependencies of an Ant component, select the desired component in the editor or ***Outline*** *view* and choose the **Component Dependencies** action from the contextual menu. The action is available for the following components: *properties*, *targets*, *extension-points*, and *references* (those that have an ID set).

The following toolbar actions are available in the **Component Dependencies** view:

Refresh

Refreshes the dependencies structure.

Stop

Stops the dependencies computing.

Configure

Allows you to configure a search scope to compute the dependencies structure. You can decide to use automatically the defined scope for future operations by selecting the corresponding checkbox.

History

Allows you to repeat a previous dependencies computation.

The following actions are available from the contextual menu:

Go to First Reference

Selects the first reference of the referenced component from the current selected component in the dependencies tree.

Go to Component

Shows the definition of the current selected component in the dependencies tree.

Tip: If a component contains multiple references to another, these references are listed in a table displayed at the bottom of **Component Dependencies** view. When a recursive reference is encountered, it is marked with a special icon .

Related Information:

Search and Refactor Operations Scope on page 480

Highlight Component Occurrences

When a component (for example *property* or *target*) is found at the current cursor position, they are highlighted in both the document and in the stripe bar at the right side of the document. Oxygen XML Editor also supports occurrences highlight for type and task references.

Customizable colors are used (one for the component definition and another one for component references). Occurrences are displayed until another component is selected and a new search is performed. All highlights are removed when you start to edit the document.

This feature is enabled by default. To configured it, *open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Editor** > **Mark Occurrences**. If your particular type of file is not selected, you can perform this search by going to **Search** > **Search Occurrences in File** **Ctrl + Shift + U (Command + Shift + U on OS X)** in the contextual menu. Matches are displayed in separate tabs of the ***Results*** *view*.

Related Information:

Mark Occurrences Preferences on page 111

Find References and Declarations of Ant Components

The following search actions related to references and declarations of Ant components are available from the **Search** submenu of the contextual menu and from the **Document** > **References** menu::

- **Search References** - Searches all references of the item found at current cursor position in the defined scope.

- **Search References in** - Searches all references of the item found at current cursor position in the file or files that you specify after selecting a scope for the search operation.
- **Search Declarations** - Searches all declarations of the item found at current cursor position in the defined scope.
- **Search Declarations in** - Searches all declarations of the item found at current cursor position in the file or files that you specify when defining a new scope.
- **Search Occurrences in File** - Searches all occurrences of the item at the cursor position in the currently edited file.

The following action is available from the contextual menu and the **Document** > **Schema** menu:

- **Show Definition** - Moves the cursor to the location of the definition of the current item.

  Note: You can also use the **Ctrl + Single-Click (Command + Single-Click on OS X)** shortcut on a reference to display its definition.

Related Information:

Search and Refactor Operations Scope on page 480

Ant Refactoring Actions

The following refactoring actions can be applied on *targets*, *extension-points*, *properties*, and *references* and allow you to consistently rename a component in the entire Ant build file structure. They are available from the **Refactoring** submenu in the contextual menu of the current editor or from the **Document** > **Refactoring** menu:

- **Rename Component** - Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the **Esc** or **Enter** key on your keyboard.
- **Rename Component in** - Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the **Preview** button, you can view the files to be affected by the action.

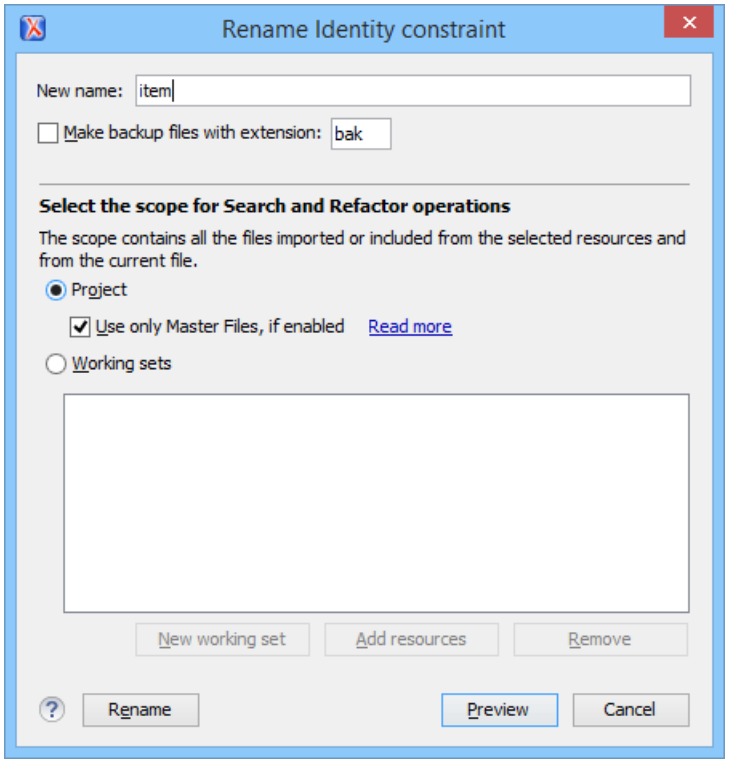


Figure 296: Rename Identity Constraint Dialog Box

Ant Quick Assist Support

The *Quick Assist support* helps you to rapidly access search and refactoring actions. If one or more actions are available in the current context, they are accessible via a yellow bulb icon ( ) placed at the current line in the stripe on the left side of the editor. Also, you can invoke the *Quick Assist* menu by using the **Alt + 1** (**Meta + Alt + 1** on Mac OS X) keyboard shortcuts.

The *Quick Assist* support offers direct access to the following actions:

Rename Component in

Renames the component and all its dependencies.

Search Declarations

Searches the declaration of the component in a predefined scope. It is available only when the context represents a component name reference.

Search References

Searches all references of the component in a predefined scope.

Component Dependencies

Searches the component dependencies in a predefined scope.

Change Scope

Configures the scope that will be used for future search or refactor operations.

Rename Component

Allows you to rename the current component in-place.

Search Occurrences

Searches all occurrences of the component within the current file.

Related Information:

Ant Component Dependencies View on page 550
Ant resource Hierarchy/Dependencies View on page 549
Ant Refactoring Actions on page 551
Search and Refactor Operations Scope on page 480

Editing XML Schemas

An XML Schema describes the structure of an XML document and is used to validate XML document instances against it, to check that the XML instances conform to the specified requirements. If an XML instance conforms to the schema then it is said to be valid. Otherwise, it is invalid.

Oxygen XML Editor offers support for both XML Schema 1.0 and 1.1 and you can edit XML Schema files in the following editing modes:

- ***Text editing mode*** - Allows you to edit XML Schema files in a source editing mode.
- ***Grid editing mode*** - Displays XML Schema files in a structured spreadsheet-like grid.
- ***Design editing mode*** - Visual schema designer that helps you understand the structure and develop complex schemas.
- ***Author editing mode*** - The visual **Author** mode is also available for XML Schema, allowing you to visually edit the schema annotations. It presents a polished and compact view of the XML Schema, with support for links on imported/included schemas.

For information about applying and detecting schemas, see *Associating a Schema to XML Documents* on page 464.

Related Information:

Associating a Schema to XML Documents on page 464

Design Editing Mode (XML Schema Diagram Editor)

XML Schemas allow document designers to specify the allowed structure and content of an XML document and to check if an XML document is valid.

Oxygen XML Editor provides a simple and expressive XML Schema diagram editor (**Design** mode) for editing XML Schemas. The schema diagram helps both the content authors who want to understand a schema and schema designers who develop complex schemas.

The diagram font can be increased using the usual Oxygen XML Editor shortcuts: **(Ctrl (Meta on Mac OS) + "+")** , **(Ctrl (Meta on Mac OS)+"-")** , **(Ctrl (Meta on Mac OS) + 0)** or **(Ctrl (Meta on Mac OS) - mouse wheel)**. The whole diagram can also be zoomed with one of the predefined factors *available in the Schema preferences panel*. The same zoom factor is applied for the print and save actions.

To switch to this mode, select **Design** at the bottom of the editing area.

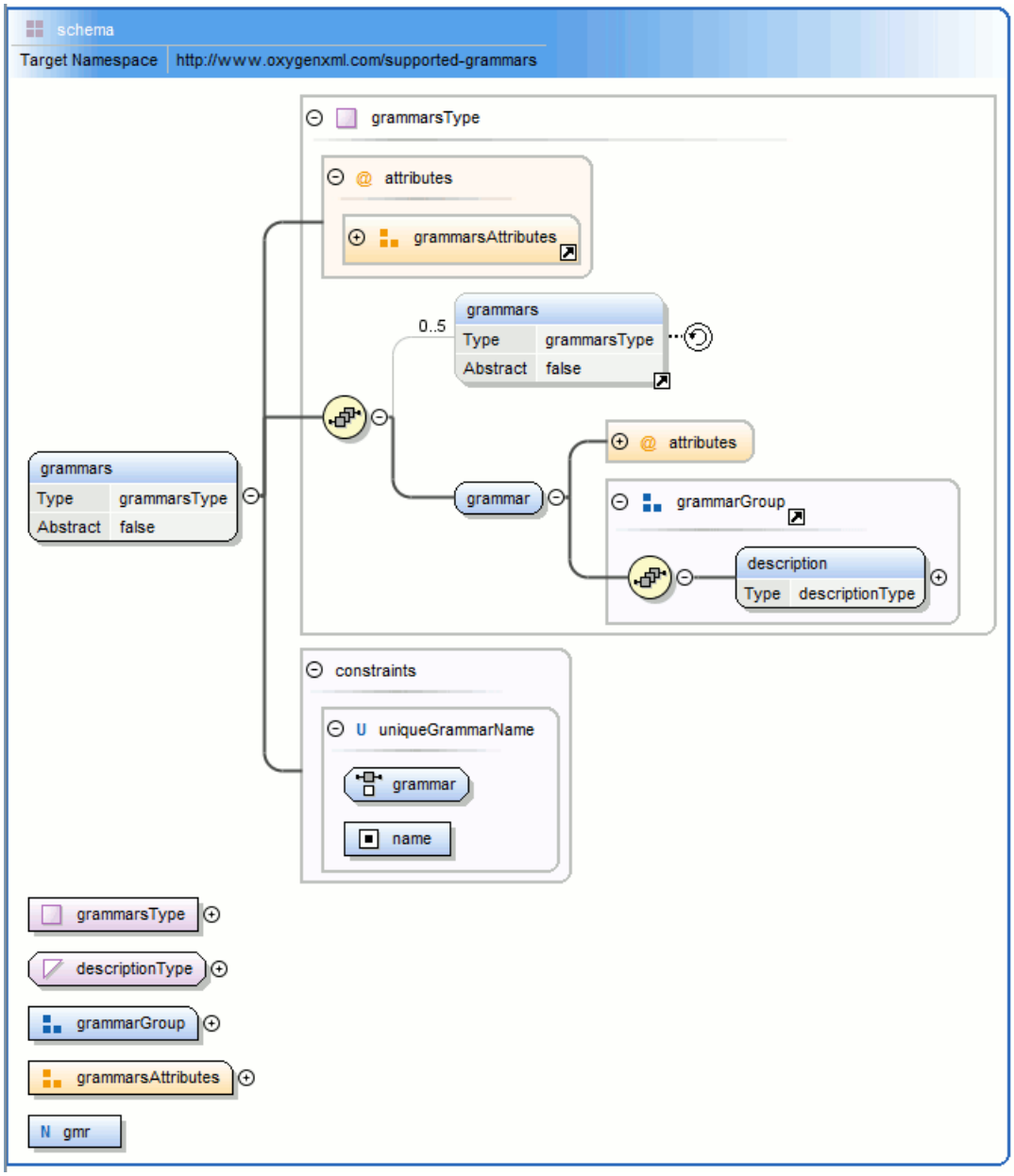


Figure 297: XML Schema Diagram

To watch our video demonstration about the basic aspects of designing an XML Schema using the new Schema Editor, go to *https://www.oxygenxml.com/demo/XML_Schema_Editing.html*.

Navigation in the XML Schema Design Mode

The following editing and navigation features work for all types of schema components in the XML Schema **Design** mode:

- Move/reference components in the diagram using drag-and-drop actions.
- Select consecutive components on the diagram (components from the same level) using the *Shift* key. You can also make discontinuous selections in the schema diagram using the **Ctrl (Meta on Mac OS)** key. To deselect one of the components, use **Ctrl + Single-Click (Command + Single-Click on OS X)**.
- Use the arrow keys to navigate the diagram vertically and horizontally.
- Use *Home*/*End* keys to jump to the first/last component from the same level. Use **Ctrl + Home (Command + Home on OS X)** key combination to go to the diagram root and **Ctrl + End (Command + End on OS X)** to go to the last child of the selected component.
- You can easily go back to a previously visited component while moving from left to right. The path will be preserved only if you use the left arrow key or right arrow key. For example, if the current selection is on the second attribute from an attribute group and you press the left arrow key to jump to the attribute group, when you press the right arrow key, then the selection will be moved to the second attribute.
- Go back and forward between components viewed or edited in the diagram by selecting them in the ***Outline*** *view*:
  - **Back** (go to previous schema component).
  - **Forward** (go to next schema component).
  - **Go to Last Modification** (go to last modified schema component).
- Copy, reference, or move global components, attributes, and identity constraints to another position and from one schema to another using the **Cut/Copy** and **Paste/Paste as Reference** actions.
- Go to the definition of an element or attribute with the **Show Definition** action.
- Search in the diagram using the *Find/Replace dialog box* or the *Quick find toolbar*. You can find/replace components only in the current file scope.
- You can expand and see the contents of the imports/includes/redefines in the diagram. In order to edit components from other schemas the schema for each component will be opened as a separate file in Oxygen XML Editor.

  Tip: If an XML Schema referenced by the current opened schema was modified on disk, the change will be detected and you will be asked to refresh the current schema contents.
- Recursive references are marked with a *recurse symbol* ( ). Click this symbol to navigate between the element declaration and its reference.

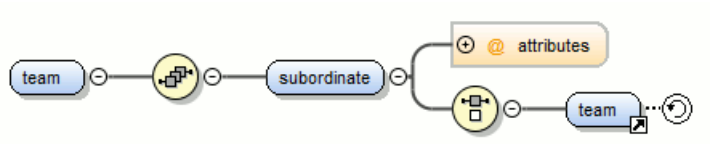


Figure 298: Recursive Reference

Schema Editing Actions

You can edit an XML schema using drag and drop operations or contextual menu actions.

Drag and drop is the easiest way to move the existing components to other locations in an XML schema. For example, you can quickly insert an element reference in the diagram with a drag and drop from the ***Outline*** *view* to a compositor in the diagram. Also, the components order in an `xs:sequence` can be easily changed using drag and drop.

If this property has not been set, you can easily set the attribute/element type by dragging over it a simple type or complex type from the diagram. If the type property for a simple type or complex type is not already set, you can set it by dragging over it a simple or complex type.

Depending on the drop area, various actions are available:

- **move** - Context dependent, the selected component is moved to the destination.
- **reference** - Context dependent, the selected component is referenced from the parent.
- **copy** - If **(Ctrl (Meta on Mac OS))** key is pressed, a copy of the selected component is inserted to the destination.

Visual clues about the operation type are indicated by the mouse pointer shape:

- - When moving a component.
- - When referencing a component.
- - When copying a component.

You can edit some schema components directly in the diagram. For these components, you can edit the name and the additional properties presented in the diagram by double clicking the value you want to edit. If you want to edit the name of a selected component, you can also press **(Enter)**. The list of properties that can be displayed for each component can be customized *in the Preferences*.

When editing references, you can choose from a list of available components. Components from an imported schema for which the target namespace does not have an associated prefix is displayed in the list as `componentName#targetNamespace`. If the reference is from a target namespace that was not yet mapped, you are prompted to add prefix mappings for the inserted component namespace in the current edited schema.

You can also change the compositor by double-clicking it and choose the compositor you want from the proposals list.

There are some components that cannot be edited directly in the diagram: imports, includes, redefines. The editing action can be performed if you double-click or press **(Enter)** on an import/include/redefine component. An edit dialog box is displayed, allowing you to customize the directives.

Related Information:

Searching and Refactoring Actions in XML Schemas

Component Dependencies View for XML Schema

XML Schema Resource Hierarchy / Dependencies View

Generating Sample XML Files

Schema Design Preferences on page 92

Contextual Menu Actions in the Design Mode

The contextual menu of the **Design** mode includes the following actions:

Show Definition (Ctrl + Shift + Enter)

Shows the definition for the current selected component. For references, this action is available by clicking the arrow displayed in its bottom right corner.

Open Schema (Ctrl + Shift + Enter)

Opens the selected schema. This action is available for `xsd:import`, `xsd:include` and `xsd:redefine` elements. If the file you try to open does not exist, a warning message is displayed and you have the possibility to create the file.

Edit Attributes ()

Allows you to edit the attributes of the selected component in a small in-place editor that presents the same attributes as in the ***Attributes*** *view* and the ***Facets*** *view*. The actions that can be performed on attributes in this dialog box are the same actions presented in the two views.

Append child

Offers a list of valid components, depending on the context, and appends your selection as a child of the currently selected component. You can set a name for a named component after it has been added in the diagram.

Insert before

Offers a list of valid components, depending on the context, and inserts your selection before the selected component, as a sibling. You can set a name for a named component after it has been added in the diagram.

Insert after

Offers a list of valid components, depending on the context, and inserts your selection after the selected component, as a sibling. You can set a name for a named component after it has been added in the diagram.

New global

Inserts a global component in the schema diagram. This action does not depend on the current context. If you choose to insert an import you have to specify the URL of the imported file, the target namespace and the import ID. The same information, excluding the target namespace, is requested for an `xsd:include` or `xsd:redefine` element.

Note: If the imported file has declared a target namespace, the field **Namespace** is completed automatically.

Edit Schema Namespaces

When performed on the schema root, it allows you to edit the schema target namespace and namespace mappings. You can also invoke the action by double-clicking the target namespace property from ***Attributes view*** for the schema or by double-clicking the schema component.

Edit Annotations

Allows you to edit the annotation for the selected schema component in the **Edit Annotations** dialog box. You can perform the following operations in the dialog box:

- **Edit all `appinfo/documentation` items for a specific annotation** - All `appinfo/documentation` items for a specific annotation are presented in a table and can be easily edited. Information about an annotation item includes: type (documentation/appinfo), content, source (optional, specify the source of the `documentation/appinfo` element) and `xml:lang`. The content of a `documentation/appinfo` item can be edited in the **Content** area below the table.
- **Insert/Insert before/Remove documentation/appinfo.** The **Add** button allows you to insert a new annotation item (`documentation/appinfo`). You can add a new item before the item selected in table by pressing the **Insert Before** button. Also, you can delete the selected item using the **Remove** button.
- **Move items up/down** - to do this use the **Move up** and **Move down** buttons.
- **Insert/Insert before/Remove annotation** - Available for components that allow multiple annotations such as schemas or redefines.
- **Specify an ID for the component annotation**. An optional identifier for the annotation.

Annotations are rendered by default under the graphical representation of the component. When you have a reference to a component with annotations, these annotations are presented in the diagram also below the reference component. The **Edit Annotations** action invoked from the contextual menu edit the annotations for the reference. If the reference component does not have annotations, you can edit the annotations of the referenced component by double-clicking the annotations area. Otherwise, you can edit the referenced component annotations only if you go to the definition of the component.

Note: For imported/included components that do not belong to the currently edited schema, the **Edit Annotations** dialog box presents the annotation as read-only. To edit its annotation, open the schema where the component is defined.

Extract Global Element

Action that is available for local elements. A local element is made global and is replaced with a reference to the global element. The local element properties that are also valid for the global element declaration are kept.

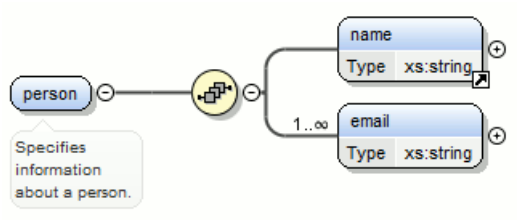


Figure 299: Extracting a Global Element

If you use the **Extract Global Element** action on a `name` element , the result is:

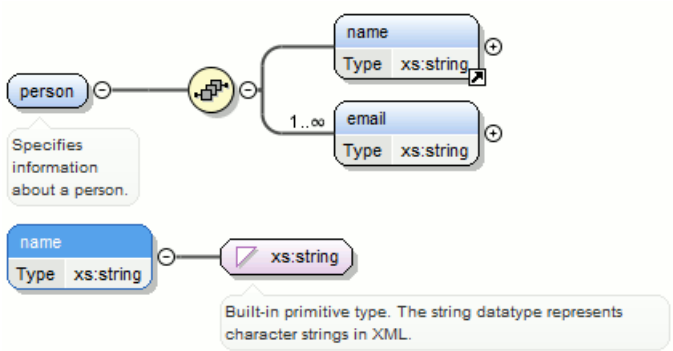


Figure 300: Extracting a Global Element on a `name` Element

Extract Global Attribute

Action available for local attributes. A local attribute is made global and replaced with a reference to the global attribute. The properties of local attribute that are also valid in the global attribute declaration are kept.

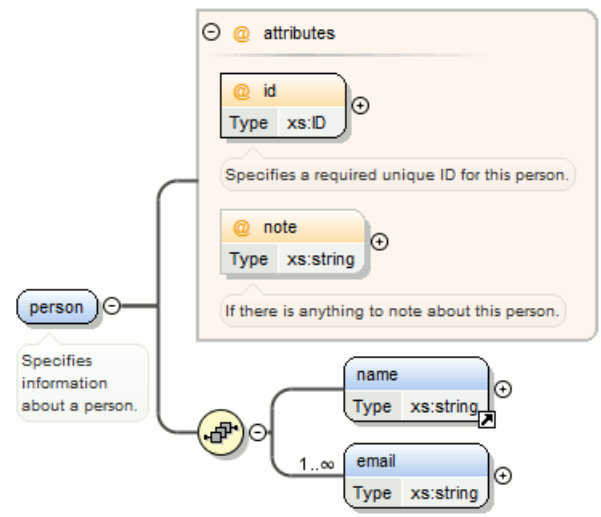


Figure 301: Extracting a Global Attribute

If you use the **Extract Global Attribute** action on a `note` attribute, the result is:

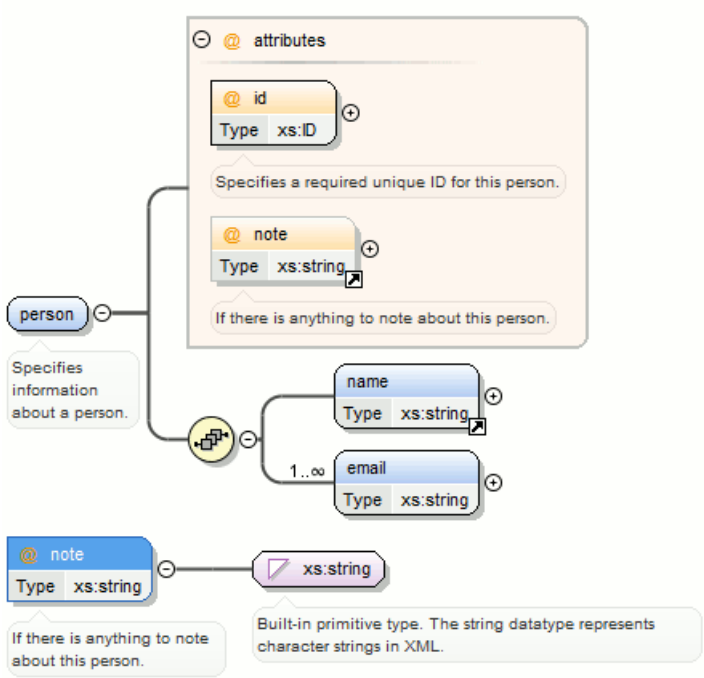
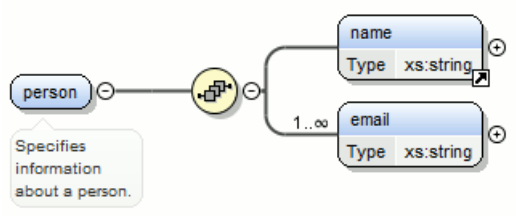


Figure 302: Extracting a Global Attribute on a `note` Attribute

Extract Global Group

Action available for compositors (sequence, choice, all). This action extracts a global group and makes a reference to it. The action is available only if the parent of the compositor is not a group.



If you use the **Extract Global Group** action on the `sequence` element, the **Extract Global Component** dialog box is displayed and you can choose a name for the group. If you type `personGroup`, the result is:

Figure 303: Extracting a Global Group

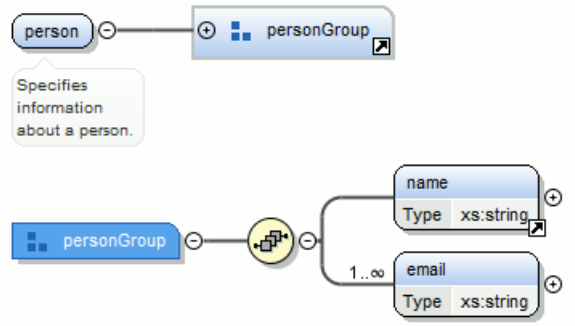


Figure 304: Extracting a Global Gropu on a `sequence` Element

Extract Global Type

Action used to extract an anonymous simple type or an anonymous complex type as global. For anonymous complex types, the action is available on the parent element.

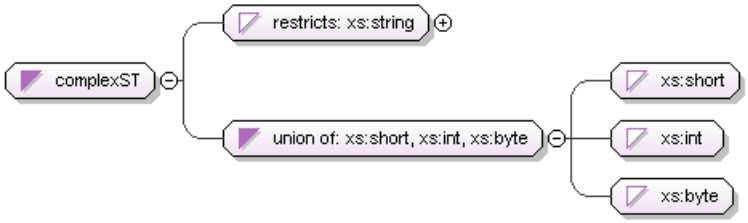


Figure 305: Extracting a Global Simple Type

If you use the action on the `union` component and choose `numericST` for the new global simple type name, the result is:

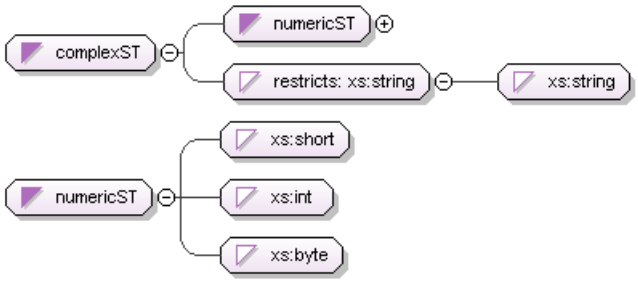


Figure 306: Extracting a Global Simple Type on a `union` Component

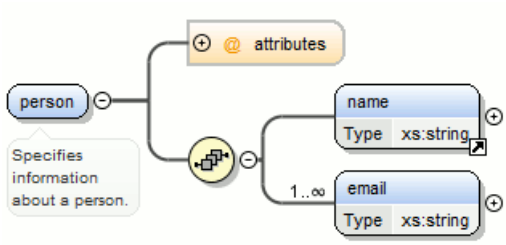


Figure 307: Extracting a Global Complex Type

If you use the action on a `person` element and choose `person_type` for the new complex type name, the result is:

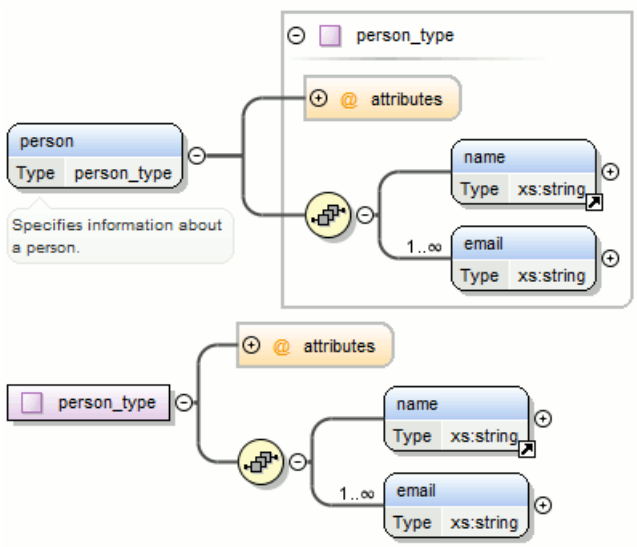


Figure 308: Extracting a Global Complex Type on a `person` Element

Rename Component in

Rename the selected component.

Cut Ctrl + X (Command + X on OS X)

Cut the selected component(s).

Copy Ctrl + C (Command + C on OS X)

Copy the selected component(s).

Copy XPath

This action copies an XPath expression that identifies the selected element or attribute in an instance XML document of the edited schema and places it in the clipboard.

Paste Ctrl + V (Command + V on OS X)

Paste the component(s) from the clipboard as children of the selected component.

Paste as Reference

Create references to the copied component(s). If not possible a warning message is displayed.

Remove (Delete)

Remove the selected component(s).

Override component

Copies the overridden component in the current XML Schema. This option is available for `xs:override` components.

Redefine component

The referenced component is added in the current XML Schema. This option is available for `xs:redefine` components.

Optional

Can be performed on element/attribute/group references, local attributes, elements, compositors, and element wildcards. The `minOccurs` property is set to 0 and the `use` property for attributes is set to `optional`.

Unbounded

Can be performed on element/attribute/group references, local attributes, elements, compositors, and element wildcards. The `maxOccurs` property is set to `unbounded` and the `use` property for attributes is set to `required`.

Search

Can be performed on local elements or attributes. This action makes a reference to a global element or attribute.

Search References

Searches all references of the item found at current cursor position in the defined scope if any.

Search References in

Searches all references of the item found at current cursor position in the specified scope.

Search Occurrences in File

Searches all occurrences of the item found at current cursor position in the current file.

Component Dependencies

Opens the ***Component Dependencies*** *view* that allows you to see the dependencies for the current selected component.

Resource Hierarchy

Opens the ***Resource Hierarchy / Dependencies*** *view* that allows you to see the hierarchy for the current selected resource.

Flatten Schema

Recursively adds the components of included Schema files to the main one. It also flattens every imported XML Schema from the hierarchy.

Resource Dependencies

Allows you to see the dependencies for the current selected resource.

Expand All

Recursively expands all sub-components of the selected component.

Collapse All

Recursively collapses all sub-components of the selected component.

Save as Image

Save the diagram as image, in JPEG, BMP, SVG or PNG format.

Generate Sample XML Files

Generate XML files using the current opened schema. The selected component is the XML document root. See more in the *Generate Sample XML Files* section.

Options

Show the *Schema preferences panel*.

XML Schema Components

A schema diagram contains a series of interconnected components. To quickly identify the relation between two connected components, the connection is represented as:

- A thick line to identify a connection with a required component (in the following image, `family` is a required element).

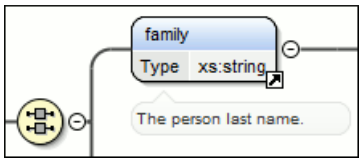


Figure 309: Example: Required Component

- A thin line to identify a connection with an optional component (in the following image, `email` is an optional element).

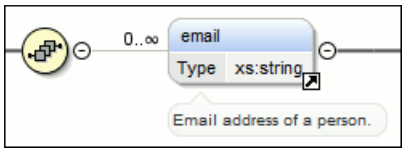


Figure 310: Example: Optional Component

The following topics explain in detail all available components and their symbols as they appear in an XML schema diagram.

xs:schema

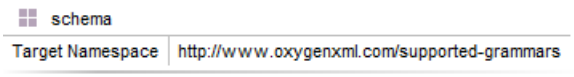


Figure 311: The *xs:schema* Component

Defines the root element of a schema. A schema document contains representations for a collection of schema components, such as type definitions and element declarations, that have a common target namespace. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-schema*.

By default, it displays the *targetNamespace* property when rendered.

Table 11: *xs:schema* Properties

Property Name	Description	Possible Values
Target Namespace	The schema target namespace.	Any URI
Element Form Default	Determining whether or not local element declarations will be namespace-qualified by default.	qualified, unqualified, [Empty]. Default value is unqualified.
Attribute Form Default	Determining whether or not local attribute declarations will be namespace-qualified by default.	qualified, unqualified, [Empty]. Default value is unqualified.
Block Default	Default value of the *block* attribute of *xs:element* and *xs:complexType*.	#all, extension, restriction, substitution, restriction extension, restriction substitution, extension substitution, restriction extension substitution, [Empty].
Final Default	Default value of the *final* attribute of *xs:element* and *xs:complexType*.	#all, restriction, extension, restriction extension, [Empty].
Default Attributes	Specifies a set of attributes that apply to every complex Type in a schema document.	Any.
Xpath Default Namespace	The default namespace used when the XPath expression is evaluated.	##defaultNamespace, ##targetNamespace, ##local.
Version	Schema version	Any token.
ID	The schema ID	Any ID.
Component	The edited component name.	Not editable property.
SystemID	The schema system ID	Not editable property.

xs:element

Figure 312: The *xs:element* Component

Defines an element. An element declaration is an association of a name with a type definition, either simple or complex, an (optional) default value and a (possibly empty) set of identity-constraint definitions. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-element*.

An element by default displays the following properties when rendered in the diagram: *default*, *fixed*, *abstract* and *type*. When referenced or declared locally, the element graphical representation also contains the value for the *minOccurs* and *maxOccurs* properties (for 0..1 and 1..1 occurs the values are implied by the connector style) and the connectors to the element are drawn using dotted lines if the element is optional.

Table 12: *xs:element* Properties

Property Name	Description	Possible Values	Mentions
Name	The element name. Always required.	Any NCName for global or local elements, any *QName* for element references.	If missing, will be displayed as '[element]' in diagram.
Is Reference	When set, the local element is a reference to a global element.	true/false	Appears only for local elements.
Type	The element type.	All declared or built-in types. In addition, the following anonymous types are available: [ST-restriction], [ST-union], [ST-list], [CT-anonymous], [CT-extension SC], [CT-restriction SC], [CT-restriction CC], [CT-extension CC].	For all elements. For references, the value is set in the referenced element.
Base Type	The extended/restricted base type.	All declared or built-in types	For elements with complex type, with simple or complex content.
Mixed	Defines if the complex type content model will be mixed.	true/false	For elements with complex type.
Content	The content of the complex type.	simple/complex	For elements with complex type that extends/restricts a base type. It is automatically detected.
Content Mixed	Defines if the complex content model will be mixed.	true/false	For elements with complex type that has a complex content.
Default	Default value of the element. A default value is automatically assigned to the element when no other value is specified.	Any string	The fixed and default attributes are mutually exclusive.
Fixed	A simple content element may be fixed to a specific value using this attribute. A fixed value is also automatically assigned to the element and you cannot specify another value.	Any string	The fixed and default attributes are mutually exclusive.

Property Name	Description	Possible Values	Mentions
Min Occurs	Minimum number of occurrences of the element.	A numeric positive value. Default value is 1	Only for references/local elements
Max Occurs	Maximum number of occurrences of the element.	A numeric positive value. Default value is 1	Only for references/local elements
Substitution Group	Qualified name of the head of the substitution group that this element belongs to.	All declared elements. For XML Schema 1.1 this property supports multiple values.	For global and reference elements
Abstract	Controls whether or not the element may be used directly in instance XML documents. When set to true, the element may still be used to define content models, but it must be substituted through a substitution group in the instance document.	true/false	For global elements and element references
Form	Defines if the element is "qualified" (belongs to the target namespace) or "unqualified" (doesn't belong to any namespace).	unqualified/qualified	Only for local elements
Nillable	When this attribute is set to true, the element can be declared as nil using an *xsi:nil* attribute in the instance documents.	true/false	For global elements and element references
Target Namespace	Specifies the target namespace for local element and attribute declarations. The namespace URI may be different from the schema target namespace. This property is available for local elements only.	Not editable property.	For all elements.

Property Name	Description	Possible Values	Mentions
Block	Controls if the element can be subject to a type or substitution group substitution. '#all' blocks any substitution, 'substitution' blocks any substitution through substitution groups and 'extension'/'restriction' block any substitution (both through *xsi:type* and substitution groups) by elements or types, derived respectively by extension or restriction from the type of the element. Its default value is defined by the *blockDefault* attribute of the parent *xs:schema*.	#all, restriction, extension,substitution, extension restriction, extension substitution, restriction substitution, restriction extension substitution	For global elements and element references
Final	Controls whether the element can be used as the head of a substitution group for elements whose types are derived by extension or restriction from the type of the element. Its default value is defined by the *finalDefault* attribute of the parent *xs:schema*.	#all, restriction, extension, restriction extension, [Empty]	For global elements and element references
ID	The component ID.	Any ID	For all elements.
Component	The edited component name.	Not editable property.	For all elements.
Namespace	The component namespace.	Not editable property.	For all elements.
System ID	The component system ID.	Not editable property.	For all elements.

xs:attribute

Figure 313: The *xs:attribute* Component

Defines an attribute. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-attribute*.

An attribute by default displays the following properties when rendered in the diagram: *default*, *fixed*, *use* and *type*. Connectors to the attribute are drawn using dotted lines if the attribute use is optional. The attribute name is stroked out if prohibited.

Table 13: *xs:attribute* Properties

Property Name	Description	Possible Value	Mentions
Name	Attribute name. Always required.	Any NCName for global/local attributes, all declared attributes' *QName* for references.	For all local or global attributes. If missing, will be displayed as '[attribute]' in the diagram.
Is Reference	When set, the local attribute is a reference.	true/false	For local attributes.
Type	Qualified name of a simple type.	All global simple types and built-in simple types. In addition another 3 proposals are present: [anonymous restriction], [anonymous list], [anonymous union] for creating anonymous simple types more easily.	For all attributes. For references, the type is set to the referenced attribute.
Default	Default value. When specified, an attribute is added by the schema processor (if it is missing from the instance XML document) and it is given this value. The default and fixed attributes are mutually exclusive.	Any string	For all local or global attributes. For references the value is from the referenced attribute.
Fixed	When specified, the value of the attribute is fixed and must be equal to this value. The default and fixed attributes are mutually exclusive.	Any string	For all local or global attributes. For references the value is from the referenced attribute.
Use	Possible usage of the attribute. Marking an attribute "prohibited" is useful to exclude attributes during derivations by restriction.	optional, required, prohibited	For local attributes
Form	Specifies whether or not the attribute is qualified (must have a namespace prefix in the instance XML document). The default value for this attribute is specified by the *attributeFormDefault* attribute of the *xs:schema* document element.	unqualified/qualified	For local attributes.

Property Name	Description	Possible Value	Mentions
Inheritable	Specifies if the attribute is inheritable. Inheritable attributes can be used by <alternative> element on descendant elements.	true/false	For all local or global attributes. The default value is false. This property is available for XML Schema 1.1.
Target Namespace	Specifies the target namespace for local attribute declarations. The namespace URI may be different from the schema target namespace.	Any URI	Setting a target namespace for local attribute is useful only when restricts attributes of a complex type that is declared in other schema with a different target namespace. This property is available for XML Schema 1.1.
ID	The component ID.	Any ID	For all attributes.
Component	The edited component name.	Not editable property.	For all attributes.
Namespace	The component namespace.	Not editable property.	For all attributes.
System ID	The component system ID.	Not editable property.	For all attributes.

xs:attributeGroup

Figure 314: The *xs:attributeGroup* Component

Defines an attribute group to be used in complex type definitions. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-attributeGroup*.

Table 14: *xs:attributeGroup* Properties

Property Name	Description	Possible Values	Mentions
Name	Attribute group name. Always required.	Any NCName for global attribute groups, all declared attribute groups for reference.	For all global or referenced attribute groups. If missing, will be displayed as '[attributeGroup]' in diagram.
ID	The component ID.	Any ID	For all attribute groups.
Component	The edited component name.	Not editable property.	For all attribute groups.
Namespace	The component namespace.	Not editable property.	For all attribute groups.
System ID	The component system ID.	Not editable property.	For all attribute groups.

xs:complexType

Figure 315: The *xs:complexType* Component

Defines a top level complex type. Complex Type Definitions provide for: See more data at *http://www.w3.org/TR/xmlschema11-1/#element-complexType*.

- Constraining element information items by providing Attribute Declarations governing the appearance and content of attributes.
- Constraining element information item children to be empty, or to conform to a specified element-only or mixed content model, or else constraining the character information item children to conform to a specified simple type definition.
- Using the mechanisms of Type Definition Hierarchy to derive a complex type from another simple or complex type.
- Specifying post-schema-validation infoset contributions for elements.
- Limiting the ability to derive additional types from a given complex type.
- Controlling the permission to substitute, in an instance, elements of a derived type for elements declared in a content model to be of a given complex type.

Tip: A complex type that is a base type to another type will be rendered with yellow background.

Table 15: *xs:complexType* Properties

Property Name	Description	Possible Values	Mentions
Name	The name of the complex type. Always required.	Any NCName	Only for global complex types. If missing, will be displayed as '[complexType]' in diagram.
Base Type Definition	The name of the extended/restricted types.	Any from the declared simple or complex types.	For complex types with simple or complex content.
Derivation Method	The derivation method.	restriction/ extension	Only when base type is set. If the base type is a simple type, the derivation method is always extension.
Content	The content of the complex type.	simple/ complex	For complex types that extend/restrict a base type. It is automatically detected.
Content Mixed	Specifies if the complex content model will be mixed.	true/false	For complex contents.
Mixed	Specifies if the complex type content model will be mixed.	true/false	For global and anonymous complex types.

Property Name	Description	Possible Values	Mentions
Abstract	When set to *true*, this complex type cannot be used directly in the instance documents and needs to be substituted using an *xsi:type* attribute.	true/false	For global and anonymous complex types.
Block	Controls if a substitution (either through a *xsi:type* or substitution groups) can be performed for a complex type, which is an extension or a restriction of the current complex type. This attribute can only block such substitutions (it cannot "unblock" them), which can also be blocked in the element definition. The default value is defined by the *blockDefault* attribute of *xs:schema*.	all, extension, restriction, extension restriction, [Empty]	For global complex types.
Final	Controls whether the complex type can be further derived by extension or restriction to create new complex types.	all, extension, restriction, extension restriction, [Empty]	For global complex types.
Default Attributes Apply	The *schema* element can carry a *defaultAttributes* attribute, which identifies an attribute group. Each *complexType* defined in the schema document then automatically includes that attribute group, unless this is overridden by the *defaultAttributesApply* attribute on the *complexType* element.	true/false	This property is available only for XML Schema 1.1.
ID	The component ID.	Any ID	For all complex types.
Component	The edited component name.	Not editable property.	For all complex types.
Namespace	The component namespace.	Not editable property.	For all complex types.
System ID	The component system ID.	Not editable property.	For all complex types.

xs:simpleType

Figure 316: The *xs:simpleType* Component

Defines a simple type. A simple type definition is a set of constraints on strings and information about the values they encode, applicable to the normalized value of an attribute information item or of an element information item with no element children. Informally, it applies to the values of attributes and the text-only content of elements. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-simpleType*.

Tip: A simple type that is a base type to another type will be rendered with yellow background.

Table 16: *xs:simpleType* Properties

Name	Description	Possible Values	Scope
Name	Simple type name. Always required.	Any NCName.	Only for global simple types. If missing, will be displayed as '[simpleType]' in diagram.
Derivation	The simple type category: restriction, list or union.	restriction,list or union	For all simple types.
Base Type	A simple type definition component. Required if derivation method is set to restriction.	All global simple types and built-in simple types. In addition another 3 proposals are present: [anonymous restriction], [anonymous list], [anonymous union] for easily create anonymous simple types.	For global and anonymous simple types with the derivation method set to restriction.
Item Type	A simple type definition component. Required if derivation method is set to list.	All global simple types and built-in simple types(from schema for schema). In addition another 3 proposals are present: [anonymous restriction], [anonymous list], [anonymous union] for easily create anonymous simple types.	For global and anonymous simple types with the derivation method set to list. Derivation by list is the process of transforming a simple datatype (named the item type) into a whitespace-separated list of values from this datatype. The item type can be defined inline by adding a simpleType definition as a child element of the list element, or by reference, using the itemType attribute (it is an error to use both).
Member Types	Category for grouping union members.	Not editable property.	For global and anonymous simple types with the derivation method set to union.

Name	Description	Possible Values	Scope
Member	A simple type definition component. Required if derivation method is set to union.	All global simple types and built-in simple types(from schema for schema). In addition another 3 proposals are present: [anonymous restriction], [anonymous list], [anonymous union] for easily create anonymous simple types.	For global and anonymous simple types with the derivation method set to union. Deriving a simple datatype by union merges the lexical spaces of several simple datatypes (called member types) to create a new simple datatype. The member types can be defined either by reference (through the memberTypes attribute) or embedded as simple datatype local definitions in the xs:union element. Both styles can be mixed.
Final	Blocks any further derivations of this datatype (by list, union, derivation or all).	#all, list, restriction, union, list restriction, list union, restriction union. In addition, [Empty] proposal is present for set empty string as value.	Only for global simple types.
ID	The component ID.	Any ID.	For all simple types
Component	The name of the edited component.	Not editable property.	Only for global and local simple types
Namespace	The component namespace.	Not editable property.	For global simple types.
System ID	The component system ID.	Not editable property.	Not present for built-in simple types..

xs:alternative

The *type alternatives* mechanism allows you to specify type substitutions on an element declaration.

Note: *xs:alternative* is available for XML Schema 1.1.

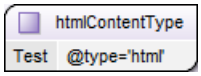


Figure 317: The *xs:alternative* Component

Table 17: *xs:alternative* Properties

Name	Description	Possible Values
Type	Specifies type substitutions for an element, depending on the value of the attributes.	All declared or built-in types. In addition, the following anonymous types are available: [ST-restriction], [ST-union], [ST-list], [CT-anonymous], [CT-extension SC], [CT-restriction SC], [CT-restriction CC], [CT-extension CC].
Test	Specifies an XPath expression. If the XPath condition is valid, the specified type is selected as the element type. The expressions allowed are limited to a subset of XPath 2.0. Only the attributes of the current element and inheritable attributes from ancestor elements are accessible in the XPath expression. When you edit this property, the content completion list of proposals offers XPath expressions.	An XPath expression.
XPath Default Namespace	The default namespace used when the XPath expression is evaluated.	##defaultNamespace, ##targetNamespace, ##local.
ID	Specifies the component ID.	Any ID.
Component	Specifies the type of XML schema component.	Not editable property.
System ID	Points to the document location of the schema.	Not editable property.

xs:group

Figure 318: The *xs:group* Component

Defines a group of elements to be used in complex type definitions. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-group*.

When referenced, the graphical representation also contains the value for the *minOccurs* and *maxOccurs* properties (for 0..1 and 1..1 occurs the values are implied by the connector style) and the connectors to the group are drawn using dotted lines if the group is optional.

Table 18: *xs:group* Properties

Property Name	Description	Possible Values	Mentions
Name	The group name. Always required.	Any NCName for global groups, all declared groups for reference.	If missing, will be displayed as '[group]' in diagram.
Min Occurs	Minimum number of occurrences of the group.	A numeric positive value. Default value is 1.	Appears only for reference groups.

Property Name	Description	Possible Values	Mentions
Max Occurs	Maximum number of occurrences of the group.	A numeric positive value. Default value is 1.	Appears only for reference groups.
ID	The component ID.	Any ID	For all groups.
Component	The edited component name.	Not editable property.	For all groups.
Namespace	The component namespace.	Not editable property	For all groups.
System ID	The component system ID.	Not editable property.	For all groups.

xs:include

Figure 319: The *xs:include* Component

Adds multiple schemas with the same target namespace to a document. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-include*.

Table 19: *xs:include* properties

Property Name	Description	Possible Values
Schema Location	Included schema location.	Any URI
ID	Include ID.	Any ID
Component	The component name.	Not editable property.

xs:import

Figure 320: The *xs:import* Component

Adds multiple schemas with a different target namespace to a document. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-import*.

Table 20: *xs:import* Properties

Property Name	Description	Possible Values
Schema Location	Imported schema location	Any URI
Namespace	Imported schema namespace	Any URI
ID	Import ID	Any ID
Component	The component name	Not editable property.

xs:redefine

Figure 321: The *xs:redefine* Component

Redefines simple and complex types, groups, and attribute groups from an external schema. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-redefine*.

Table 21: *xs:redefine* Properties

Property Name	Description	Possible Values
Schema Location	Redefine schema location.	Any URI
ID	Redefine ID	Any ID
Component	The component name.	Not editable property.

xs:override

Figure 322: The *xs:override* Component

The override construct allows replacements of old components with new ones without any constraint. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-override*.

Table 22: *xs:override* Properties

Property Name	Description	Possible Values
Schema Location	Redefine schema location.	Any URI
ID	Redefine ID	Any ID

xs:notation

Figure 323: The *xs:notation* Component

Describes the format of non-XML data within an XML document. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-notation*.

Table 23: *xs:notation* Properties

Property Name	Description	Possible values	Mentions
Name	The notation name. Always required.	Any NCName.	If missing, will be displayed as '[notation]' in diagram.
System Identifier	The notation system identifier.	Any URI	Required if public identifier is absent (otherwise, optional).
Public Identifier	The notation public identifier.	A Public ID value	Required if system identifier is absent (otherwise, optional).
ID	The component ID.	Any ID	For all notations.
Component	The edited component name.	Not editable property.	For all notations.
Namespace	The component namespace.	Not editable property.	For all notations.

Property Name	Description	Possible values	Mentions
System ID	The component system ID.	Not editable property.	For all notations.

xs:sequence / xs:choice / xs:all

Figure 324: *xs:sequence*

xs:sequence specifies that the child elements must appear in a sequence. Each child element can occur from 0 to any number of times. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-sequence*.

Figure 325: *xs:choice*

xs:choice allows only one of the elements contained in the declaration to be present within the containing element. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-choice*.

Figure 326: *xs:all*

xs:all specifies that the child elements can appear in any order. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-all*.

The compositor graphical representation also contains the value for the *minOccurs* and *maxOccurs* properties (for 0..1 and 1..1 occurs the values are implied by the connector style) and the connectors to the compositor are drawn using dotted lines if the compositor is optional.

Table 24: *xs:sequence*, *xs:choice*, *xs:all* Properties

Property Name	Description	Possible Values	Mentions
Compositor	Compositor type.	sequence, choice, all.	'all' is only available as a child of a group or complex type.
Min Occurs	Minimum occurrences of compositor.	A numeric positive value. Default is 1.	The property is not present if compositor is 'all' and is child of a group.
Max Occurs	Maximum occurrences of compositor.	A numeric positive value. Default is 1.	The property is not present if compositor is 'all' and is child of a group.
ID	The component ID.	Any ID	For all compositors.
Component	The edited component name.	Not editable property.	For all compositors.
System ID	The component system ID.	Not editable property.	For all compositors.

xs:any

Figure 327: The *xs:any* Component

Enables the author to extend the XML document with elements not specified by the schema. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-any*.

The graphical representation also contains the value for the *minOccurs* and *maxOccurs* properties (for 0..1 and 1..1 occurs the values are implied by the connector style) and the connectors to the wildcard are drawn using dotted lines if the wildcard is optional.

Table 25: *xs:any* Properties

Property Name	Description	Possible Values
Namespace	The list of allowed namespaces. The namespace attribute expects a list of namespace URIs. In this list, two values have a specific meaning: '##targetNamespace' stands for the target namespace, and '##local' stands for local attributes (without namespaces).	##any, ##other, ##targetNamespace, ##local or anyURI
notNamespace	Specifies the namespace that extension elements or attributes cannot come from.	##local, ##targetNamespace
notQName	Specifies an element or attribute that is not allowed.	##defined
Process Contents	Type of validation required on the elements allowed for this wildcard.	skip, lax, strict
Min Occurs	Minimum occurrences of any	A numeric positive value. Default is 1.
Max Occurs	Maximum occurrences of any	A numeric positive value. Default is 1.
ID	The component ID.	Any ID.
Component	The name of the edited component.	Not editable property.
System ID	The component system ID.	Not editable property.

xs:anyAttribute

Figure 328: The *xs:anyAttribute* Component

Enables the author to extend the XML document with attributes not specified by the schema. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-anyAttribute*.

Table 26: *xs:anyAttribute* Properties

Property Name	Description	Possible Value
Namespace	The list of allowed namespaces. The namespace attribute expects a list of namespace URIs. In this list, two values have a specific meaning: '##targetNamespace' stands for the target namespace, and '##local' stands for local attributes (without namespaces).	##any, ##other, ##targetNamespace, ##local or anyURI
Process Contents	Type of validation required on the elements allowed for this wildcard.	skip, lax, strict
ID	The component ID.	Any ID.
Component	The name of the edited component.	Not editable property.
System ID	The component system ID.	Not editable property.

xs:unique

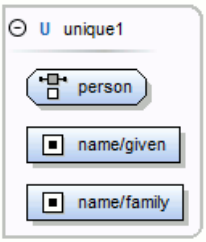


Figure 329: The *xs:unique* Component

Defines that an element or an attribute value must be unique within the scope. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-unique*.

Table 27: *xs:unique* Properties

Property Name	Description	Possible Values
Name	The unique name. Always required.	Any NCName.
ID	The component ID.	Any ID.
Component	The edited component name.	Not editable property.
Namespace	The component namespace.	Not editable property.
System ID	The component system ID.	Not editable property.

xs:key

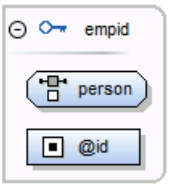


Figure 330: The *xs:key* Component

Specifies an attribute or element value as a key (unique, non-nullable and always present) within the containing element in an instance document. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-key*.

Table 28: *xs:key* Properties

Property Name	Description	Possible Value
Name	The key name. Always required.	Any NCName.
ID	The component ID.	Any ID.
Component	The edited component name.	Not editable property.
Namespace	The component namespace.	Not editable property.
System ID	The component system ID.	Not editable property.

xs:keyRef

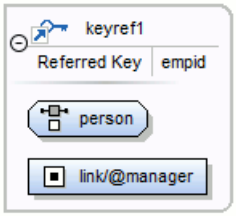


Figure 331: The *xs:keyRef* Component

Specifies that an attribute or element value corresponds to that of the specified key or unique element. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-keyref*.

A keyref by default displays the *Referenced Key* property when rendered.

Table 29: *xs:keyRef* Properties

Property Name	Description	Possible Values
Name	The keyref name. Always required.	Any NCName.
Referenced Key	The name of referenced key.	any declared element constraints.
ID	The component ID.	Any ID.
Component	The edited component name.	Not editable property.
Namespace	The component namespace.	Not editable property.
System ID	The component system ID.	Not editable property.

xs:selector

Figure 332: The *xs:selector* Component

Specifies an XPath expression that selects a set of elements for an identity constraint. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-selector*.

Table 30: *xs:selector* Properties

Property Name	Description	Possible Values
XPath	Relative XPath expression identifying the element that the constraint applies to.	An XPath expression.
ID	The component ID.	Any ID.
Component	The edited component name.	Not editable property.
System ID	The component system ID.	Not editable property.

xs:field

Figure 333: The *xs:field* Component

Specifies an XPath expression that specifies the value used to define an identity constraint. See more info at *http://www.w3.org/TR/xmlschema11-1/#element-field*.

Table 31: *xs:field* Properties

Property Name	Description	Possible Values
XPath	Relative XPath expression identifying the field(s) composing the key, key reference, or unique constraint.	An XPath expression.
ID	The component ID.	Any ID.
Component	The edited component name.	Not editable property.
System ID	The component system ID.	Not editable property.

xs:assert

Assertions provide a flexible way to control the occurrence and values of elements and attributes available in an XML Schema.

Note: *xs:assert* is available for XML Schema 1.1.

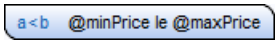


Figure 334: The *xs:assert* Component

Table 32: *xs:assert* Properties

Property Name	Description	Possible Values
Test	Specifies an XPath expression. If the XPath condition is valid, the specified type is selected as the element type. The expressions allowed are limited to a subset of XPath 2.0. Only the attributes of the current element and inheritable attributes from ancestor elements are accessible in the XPath expression. When you edit this property, the content completion list of proposals offers XPath expressions.	An XPath expression.
XPath Default Namespace	The default namespace used when the XPath expression is evaluated.	##defaultNamespace, ##targetNamespace, ##local.
ID	Specifies the component ID.	Any ID.
Component	The edited component name.	Not editable property.
System ID	The component system ID.	Not editable property.

xs:openContent

Figure 335: The *xs:openContent* Component

The *openContent* element enables instance documents to contain extension elements to be inserted amongst the elements declared by the schema. You can declare open content for your elements at one place (within the *complexType* definition) or at the schema level.

For further details about the *openContent* component, go to *http://www.w3.org/TR/xmlschema11-1/#element-openContent*.

Table 33: *xs:openContent* Properties

Property Name	Description	Possible Value
Mode	Specifies where the extension elements can be inserted.	The value can be: "interleave", "suffix" or "none". The default value is "interleave".
ID	The component ID.	Any ID.
Component	The edited component name.	Not editable property.
System ID	The component system ID.	Not editable property.

Note: This component is available for XML Schema 1.1 only. To change the version of the XML Schema, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XML Parser** > **XML Schema**.

Constructs Used to Group Schema Components

This section explains the components that can be used for grouping other schema components:

- *Attributes*

- *Constraints*
- *Substitutions*

Attributes

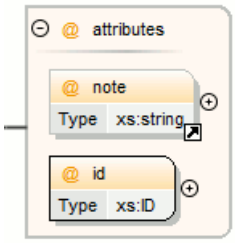


Figure 336: Attributes Construct

Groups all attributes and attribute groups belonging to a complex type.

Table 34: *attributes* Properties

Property Name	Description	Possible Values
Component	The element for which the attributes are displayed.	Not editable property.
System ID	The component system ID.	Not editable property.

Constraints

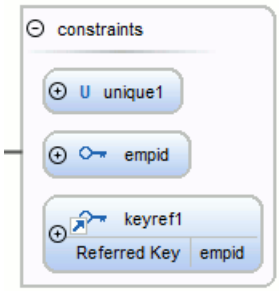


Figure 337: Constraints Construct

Groups all constraints (*xs:key*, *xs:keyRef* or *xs:unique*) belonging to an element.

Table 35: *constraints* Properties

Property Name	Description	Possible Values
Component	The element for which the constraints are displayed.	Not editable property.
System ID	The component system ID.	Not editable property.

Substitutions

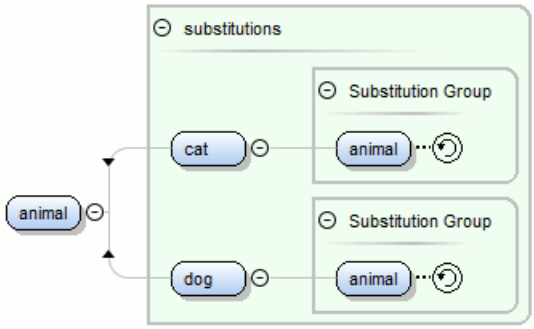


Figure 338: Substitutions Construct

Groups all elements that can substitute the current element.

Table 36: *substitutions* Properties

Property Name	Description	Possible Values
Component	The element for which the substitutions are displayed.	Not editable property.
System ID	The component system ID.	Not editable property.

Schema Validation

Validation for the **Design** mode is seamlessly integrated in the Oxygen XML Editor *XML documents validation* capability.

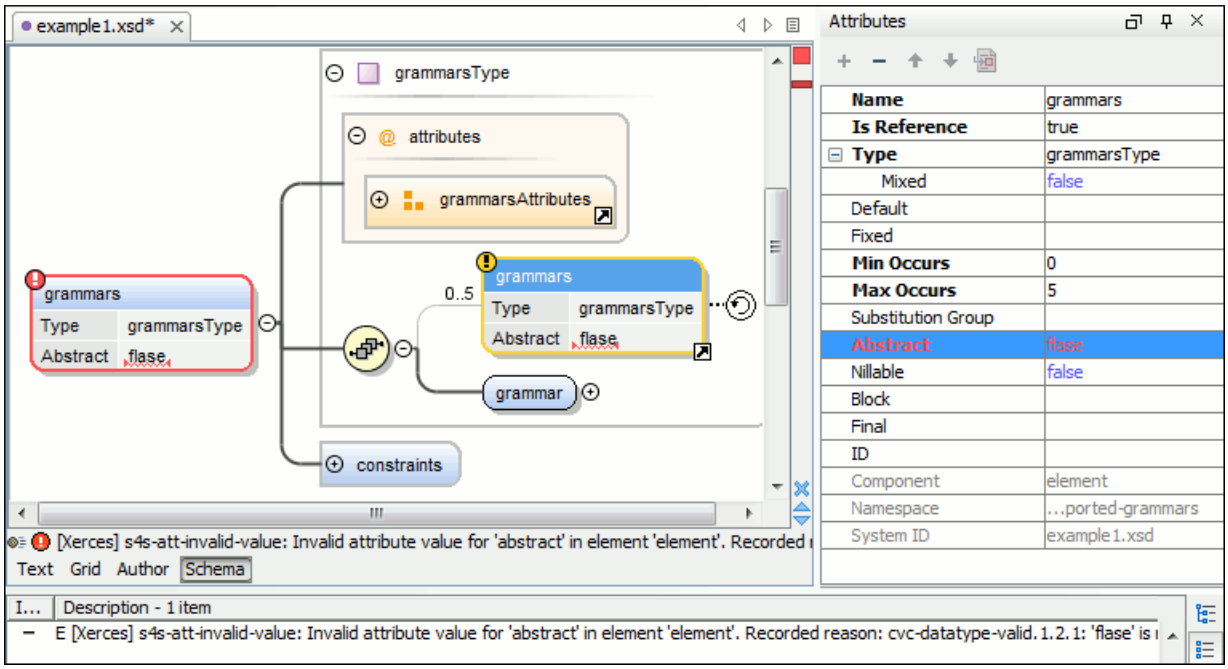


Figure 339: XML Schema Validation

A schema validation error is presented by highlighting the invalid component:

- In the *Attributes View*.
- In the diagram by surrounding the component that has the error with a red border.
- A marker on the errors stripe at the right of the diagram view.
- A status label with a red icon (!) below the diagram view.

Invalid facets for a component are highlighted in the *Facets View*.

Components with invalid properties are rendered with a red border. This is a default color, but you can customize it in the *Document checking user preferences*. When hovering an invalid component, the tooltip will present the validation errors associated with that component.

When editing a value that is supposed to be a qualified or unqualified XML name, the application provides automatic validation of the entered value. This proves to be very useful in avoiding setting invalid XML names for the given property.

If you validate the entire schema using the **Validate** action from the **Document** > **Validate** menu or from the **Validation** toolbar drop-down menu, all validation errors will be presented in the **Errors** tab. To resolve an error, just click it (or double-click for errors located in other schemas) and the corresponding schema component will be displayed as the diagram root so that you can easily correct the error.

Important: If the schema imports only the namespace of other schema without specifying the schema location and a *catalog is set-up* that maps the namespace to a certain location both the validation and the diagram will correctly identify the imported schema.

Tip: If the validation action finds that the schema contains unresolved references, the application will suggest the use of validation scenarios, but only if the current edited schema is an XML Schema module.

Edit Schema Namespaces

You can use the **XML Schema Namespaces** dialog box to easily set a target namespace and define namespace mappings for a newly created XML Schema. In the **Design** mode these namespaces can be modified anytime by choosing **Edit Schema Namespaces** from the contextual menu. You can also do this by double-clicking the schema root in the diagram.

The **XML Schema Namespaces** dialog box allows you to edit the following information:

- **Target namespace** - The target namespace of the schema.
- **Prefixes** - The dialog box displays a table with namespaces and the mapped prefixes. You can add a new prefix mapping or remove an already existing one.

Editing XML Schema in Text Editing Mode

The Oxygen XML Editor **Text** editing mode can be used for editing XML Schema in a source editing mode. It offers powerful content completion support, a synchronized Outline view, and multiple *refactoring actions*. The Outline view has two display modes: the *standard outline* mode and the *components* mode.

A diagram of the XML Schema can be presented side by side with the text. To activate the diagram presentation, select the *Show Full Model XML Schema diagram option* in the ***Diagram** preferences page*.

Editing XML Schema in the Master Files Context

Smaller interrelated modules that define a complex XML Schema cannot be correctly edited or validated individually, due to their interdependency with other modules. For example, an element defined in a main schema document is not visible when you edit an included module. Oxygen XML Editor provides the support for defining the main module (or modules), thus allowing you to edit any of the imported/included schema files in the context of the larger schema structure.

You can set a main XML document either using the *master files support from the **Project** view*, or using a validation scenario.

To set a *master file* using a validation scenario, add validation units that point to the main schemas. Oxygen XML Editor warns you if the current module is not part of the dependencies graph computed for the main schema. In this case, it considers the current module as the main schema.

The advantages of editing in the context of a *master file* include:

- Correct validation of a module in the context of a larger schema structure.
- *Content Completion Assistant* displays all the referable components valid in the current context. This include components defined in modules other than the currently edited one.
- The ***Outline** view* displays the components collected from the entire schema structure.

Validating XML Schema Documents

By default, XML Schema files are validated as you type. To change this, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Editor** > **Document Checking**, and deselect the ***Enable automatic validation** option*.

To validate an XML Schema document manually, select the **Validate** action from the **Validation** toolbar drop-down menu or the **Document** > **Validate** menu. When Oxygen XML Editor validates an XML Schema file, it expands all the included modules so the entire schema hierarchy is validated. The validation problems are highlighted directly in the editor, making it easy to locate and fix any issues.

Related Information:

Validating XML Documents Against a Schema on page 446
Validation Scenario on page 452
Associating a Schema to XML Documents on page 464
Presenting Validation Errors in Author Mode on page 236
Presenting Validation Errors in Text Mode on page 206

References to XML Schema Specification

The same as in editing XML documents, the message of an error obtained by validation of an XML Schema document includes a reference to the W3C specification for XML Schema. An error message contains an *Info* field that will open the browser on the "XML Schema Part 1:Structures" specification at exactly the point where the error is described, thus allowing you to understand the reason for that error.

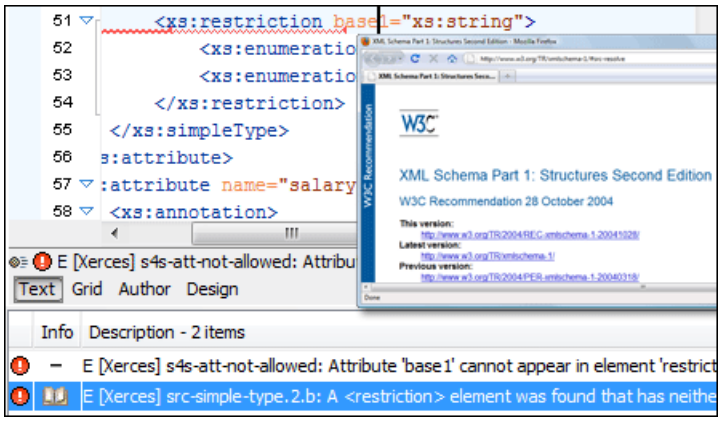


Figure 340: Link to Specification for XML Schema Errors

Validation of an XML Schema containing a type definition with a `minOccurs` or `maxOccurs` attribute having a value larger than 256 limits the value to 256 and issues a warning about this restriction in the Message panel at the bottom of the Oxygen XML Editor window. Otherwise, for large values of the `minOccurs` and `maxOccurs` attributes, the validator fails with an **OutOfMemory** error that might make Oxygen XML Editor unusable without restarting the entire application.

Important: If the schema imports only a namespace without specifying the schema location and a *catalog is set up* to map the namespace to a certain location, both validation and the schema components will correctly identify the imported schema.

Quick Fixes for DTD, XSD, and Relax NG Errors

Oxygen XML Editor offers *Quick Fixes* for common errors that appear in XML documents that are validated against DTD, XSD, or Relax NG schemas.

Note: For XML documents validated against XSD schemas, the *Quick Fixes* are only available if you use the default Xerces validation engine.

Quick Fixes are available in **Text** mode and **Author** mode.

Oxygen XML Editor provides *Quick Fixes* for numerous types of problems, including the following:

Problem Type	Available Quick Fixes
A specific element is required in the current context	Insert the required element
An element is invalid in the current context	Remove the invalid element
The content of the element should be empty	Remove the element content
An element is not allowed to have child elements	Remove all child elements
Text is not allowed in the current element	Remove the text content
A required attribute is missing	Insert the required attribute
An attribute is not allowed to be set for the current element	Remove the attribute
The attribute value is invalid	Propose the correct attribute values
ID value is already defined	Generate a unique ID value
References to an invalid ID	Change the reference to an already defined ID

Related Information:

Schematron Quick Fixes (SQF) on page 463

Content Completion in XML Schema

The intelligent *Content Completion Assistant* allows you to quickly identify and insert elements, attributes, and attribute values that are valid in the current editing context. All available proposals are listed in a pop-up menu displayed at the current cursor position.

The *Content Completion Assistant* is enabled by default. To disable it, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Editor** > **Content Completion**, and deselect the ***Enable content completion** option*.

When active, the *Content Completion Assistant* displays a list of context-sensitive proposals valid at the current cursor position. It can be manually activated with the **Ctrl + Space (Command + Space on OS X)** shortcut. You can navigate through the list of proposals by using the **Up** and **Down** keys on your keyboard. For each selected item in the list, the *Content Completion Assistant* displays a documentation window. You can customize the size of the documentation window by dragging its top, right, and bottom borders.

To insert the selected content in **Text** mode, do one of the following:

- Press **Enter** or **Tab** to insert both the start and end tags and position the cursor inside the start tag in a position suitable for inserting attributes.
- Press **Ctrl + Enter (Command + Enter on OS X)** to insert both the start and end tags and positions the cursor between the tags in a position where you can start typing content.

Depending on the *selected schema version*, Oxygen XML Editor populates the proposals list with information taken either from XML Schema 1.0 or 1.1.

Oxygen XML Editor helps you to easily reference a component by providing the list of proposals (complex types, simple types, elements, attributes, groups, attribute groups, or notations) valid in the current context. The components are collected from the current file or from the imported/included schemas.

When editing `xs:annotation/xs:appinfo` elements of an XML Schema, the *Content Completion Assistant* proposes elements and attributes from a custom schema (by default ISO Schematron). This feature can be configured from the *XSD Content Completion* preferences page.

Syntax Highlighting in XML Schema

Oxygen XML Editor supports syntax highlighting when editing XML Schema in **Text** mode to improve the readability of the content and reduce the time it takes to internalize the semantics of the structured content.

To customize the colors or styles used for the syntax highlighting colors for XML Schema files, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)***.
2. Go to ***Editor*** > ***Syntax Highlight***.
3. Select and expand the **XML** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the **XSD** tab in the **Preview** pane to see the effects of your changes.

Tip: Oxygen XML Editor also allows you to specify syntax highlighting colors for specific XML elements and attributes with specific namespace prefixes. This can be done in the ***Editor*** > ***Syntax Highlight*** > ***Elements/ Attributes by Prefix*** *preferences page*.

Related Information:

Customize Syntax Highlight colors on page 104

XML Schema Outline View

The **Outline** view for XML Schemas presents all the global components grouped by their location, namespace, or type. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

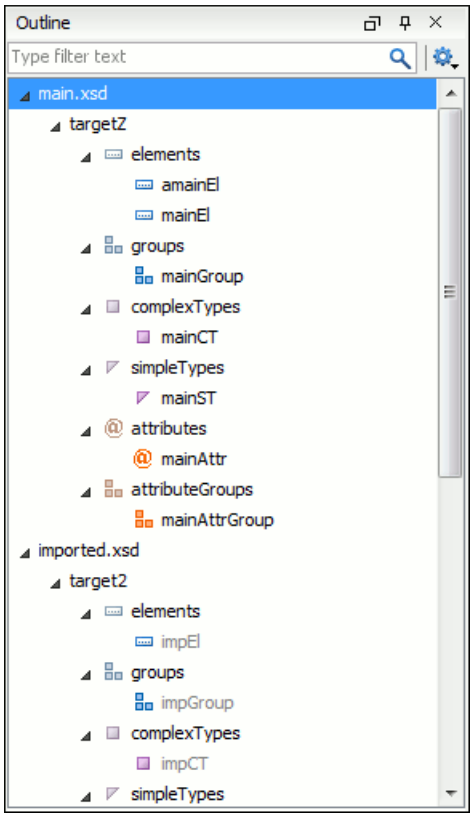


Figure 341: Outline View for XML Schema

The **Outline** view provides the following options in the **Settings** menu on the **Outline** view toolbar:

Filter returns exact matches

The text filter of the **Outline** view returns only exact matches;

Selection update on cursor move

Allows a synchronization between **Outline** view and schema diagram. The selected view from the diagram is also selected in the **Outline** view.

Sort

Allows you to sort alphabetically the schema components.

Show all components

Displays all components that were collected starting from the *master files*. Components that are not referable from the current file are marked with an orange underline. To reference them, add an import directive with the `componentNS` namespace.

Show referable components

Displays all components (collected starting from the *master files*) that can be referenced from the current file. This option is set by default.

Show only local components

Displays the components defined in the current file only.

Group by location/namespace/type

These three operations allow you to group the components by location, namespace, or type. When grouping by namespace, the main schema target namespace is the first presented in the **Outline** view.

The following contextual menu actions are available in the **Outline** view:

Remove (Delete)

Removes the selected item from the diagram.

Search References (Ctrl (Meta on Mac OS)+Shift+R)

Searches all references of the item found at current cursor position in the defined scope, if any.

Search References in

Searches all references of the item found at current cursor position in the specified scope.

Component Dependencies (Ctrl (Meta on Mac OS)+Shift+F4)

Opens the ***Component Dependencies*** *view* that allows you to see the dependencies for the current selected component.

Resource Hierarchy (F4)

Opens the ***Resource Hierarchy / Dependencies*** *view* that allows you to see the hierarchy for the current selected resource.

Resource Dependencies (Shift + F4)

Opens the ***Resource Hierarchy / Dependencies*** *view* that allows you to see the dependencies for the current selected resource.

Rename Component in

Renames the selected component.

Generate Sample XML Files

Generate XML files using the current opened schema. The selected component is the XML document root.

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (*, ?) and separate multiple patterns with commas.

Tip: The search filter is case insensitive. The following wildcards are accepted:

- ***** - any string
- **?** - any character
- **,** - patterns separator

If no wildcards are specified, the string to search will be searched as a partial match.

The content of the **Outline** view and the editing area are synchronized. When you select a component in the **Outline** view, its definition is highlighted in the editing area.

Related Information:

Searching and Refactoring Actions in XML Schemas

Component Dependencies View for XML Schema
XML Schema Resource Hierarchy / Dependencies View
Generating Sample XML Files
Editing Relax NG Schema in the Master Files Context on page 655

XML Schema Attributes View

The **Attributes** view for XML Schemas presents the properties for the selected component in the schema diagram. By default, it is displayed on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

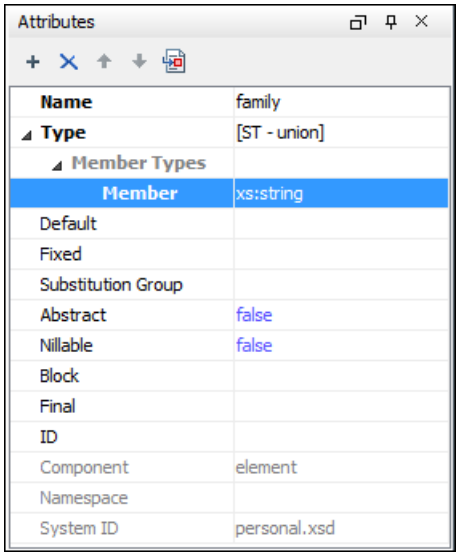


Figure 342: Attributes View

The default value of a property is presented in the **Attributes** view with blue foreground. The properties that can not be edited are rendered with gray foreground. A non-editable category that contains at least one child is rendered with bold. Bold properties are properties with values set explicitly to them.

Properties for components that do not belong to the current edited schema are read-only but if you double-click them you can choose to open the corresponding schema and edit them.

You can edit a property by double-clicking by pressing Enter. For most properties you can choose valid values from a list or you can specify another value. If a property has an invalid value or a warning, it will be highlighted in the table with the corresponding foreground color. By default, properties with errors are highlighted with red and the properties with warnings are highlighted with yellow. You can customize these colors from the *Document checking user preferences*.

For imports, includes and redefines, the properties are not edited directly in the **Attributes** view. A dialog box will open that allows you to specify properties for them.

The schema namespace mappings are not presented in **Attributes** view. You can view/edit these by choosing **Edit Schema Namespaces** from the contextual menu on the schema root. See more in the *Edit Schema Namespaces* section.

The **Attributes** view has five actions available on the toolbar and also on the contextual menu:

Add

Allows you to add a new member type to an union's member types category.

Remove

Allows you to remove the value of a property.

Move Up

Allows you to move up the current member to an union's member types category.

Move Down

Allows you to move down the current member to an union's member types category.

Copy

Copy the attribute value.

Show DefinitionCtrl (Meta on MAC OS) + Click

Shows the definition for the selected type.

Show Facets

Allows you to edit the facets for a simple type.

XML Schema Palette View

The **Palette** view is designed to offer quick access to XML Schema components and to improve the usability of the XML Schema diagram builder. You can use the **Palette** to drag and drop components in the **Design** mode. The components offered in the **Palette** view depend on the XML schema version set in the ***XML Schema*** *preferences page*. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

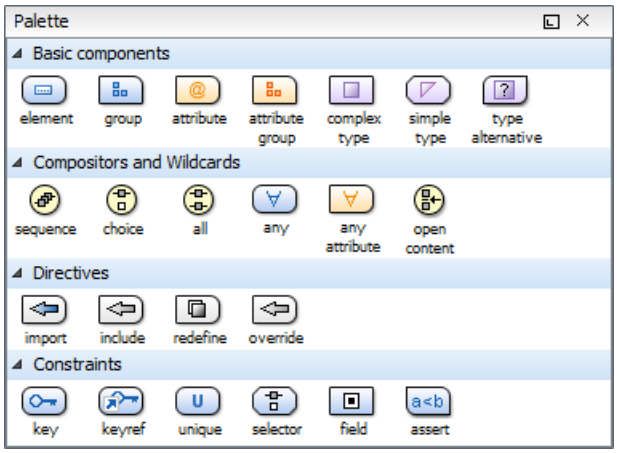


Figure 343: Palette View

Components are organized functionally into 4 collapsible categories:

- Basic components: *elements*, *group*, *attribute*, *attribute group*, *complex type*, *simple type*, *type alternative*.
- Compositors and Wildcards: *sequence*, *choice*, *all*, *any*, *any attribute*, *open content*.
- Directives: *import*, *include*, *redefine*, *override*.
- Identity constraints: *key*, *keyref*, *unique*, *selector*, *field*, *assert*.

Note: The *type alternative*, *open content*, *override*, and *assert* components are available for XML Schema 1.1.

To add a component to the edited schema:

- Click and hold a graphic symbol from the **Palette** view, then drag the component into the **Design** view.
- A line dynamically connects the component with the XML schema structure.
- Release the component into a valid position.

Note: You cannot drop a component into an invalid position. When you hover the component into an invalid position, the mouse cursor changes its shape into . Also, the connector line changes its color from the usual dark gray to the color defined in the ***Validation error highlight color*** *option* (default color is red).

To watch our video demonstration about the Schema palette and developing XML Schemas, go to *https://www.oxygenxml.com/demo/Schema_Palette.html* and *https://www.oxygenxml.com/demo/Developing_XML_Schemas.html* respectively.

XML Schema Facets View

The **Facets** view for XML Schemas presents the facets for the selected component, if available. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

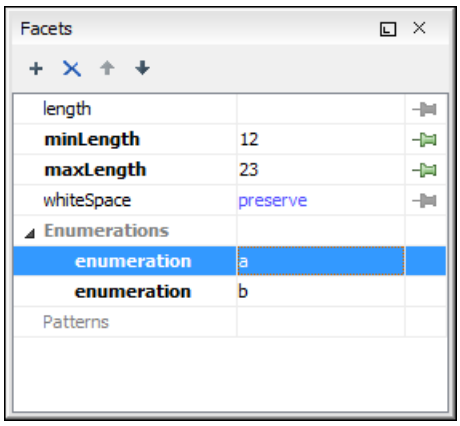


Figure 344: Facets View

The default value of a facet is rendered in the **Facets** view with a blue color. The facets that can not be edited are rendered with a gray color. The grouping categories (for example: **Enumerations** and **Patterns**) are not editable. If these categories contain at least one child they are rendered with bold. Bold facets are facets with values set explicitly to them.

Important: Usually inherited facets are presented as default in the **Facets** view but if patterns are inherited from a base type and also specified in the current simple type only the current specified patterns will be presented. You can see the effective pattern value obtained by combining the inherited and the specified patterns as a tooltip on the **Patterns** category.

Facets for components that do not belong to the current edited schema are read-only but if you double-click them you can choose to open the corresponding schema and edit them.

You can edit a facet by double-clicking it or by pressing Enter, when that facet is selected. For some facets you can choose valid values from a list or you can specify another value. If a facet has an invalid value or a warning, it will be highlighted in the table with the corresponding foreground color. By default, facets with errors are presented with red and the facets with warnings with yellow. You can customize the error colors from the *Document Checking user preferences*.

The **Facets** view provides the following actions in its toolbar and contextual menu:

Add

Allows you to add a new enumeration or a new pattern.

Remove

Allows you to remove the value of a facet.

Edit Annotations

Allows you to edit an annotation for the selected facet.

Move Up

Allows you to move up the current enumeration/pattern in **Enumerations/Patterns** category.

Move Down

Allows you to move down the current enumeration/pattern in **Enumerations/Patterns** category.

Copy

Copy the attribute value.

Open in Regular Expressions Builder

Rather than editing regular expressions manually, this action allows you to open the pattern in the *XML Schema Regular Expressions Builder* that guides you through the process of testing and constructing the pattern..

Facets can be fixed to prevent a derivation from modifying its value. To fix a facet value just press the **Pin** button.

XML Schema Resource Hierarchy / Dependencies View

The **Resource Hierarchy / Dependencies** view allows you to easily see the hierarchy / dependencies for an XML Schema, a *Relax NG schema* or an *XSLT stylesheet*. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

The **Resource Hierarchy / Dependencies** is useful when you want to start from an XML Schema (XSD) file and build and review the hierarchy of all the other XSD files that are imported, included or redefined in the given XSD file. The view is also able to build the tree structure, that is the structure of all other XSD files that import, include or redefine the given XSD file. The scope of the search is configurable (the current project, a set of local folders, etc.)

The build process for the hierarchy view is started with the **Resource Hierarchy** action available on the contextual menu of the editor panel.

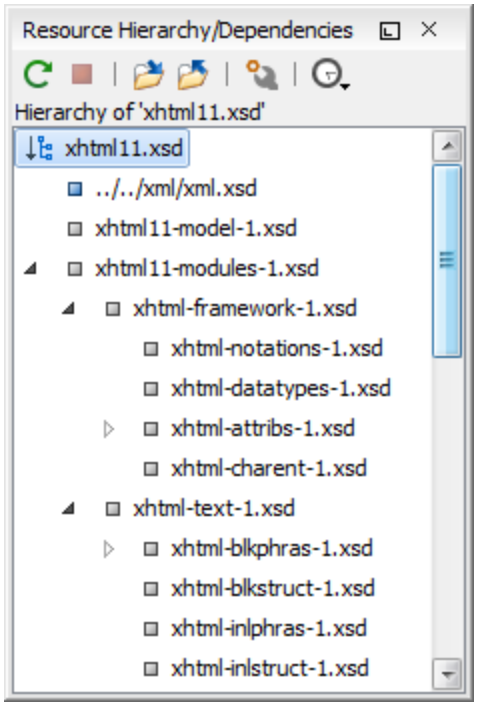


Figure 345: Resource Hierarchy/Dependencies View - Hierarchy for `xhtml11.xsd`

The build process for the dependencies view is started with the **Resource Dependencies** action available on the contextual menu.

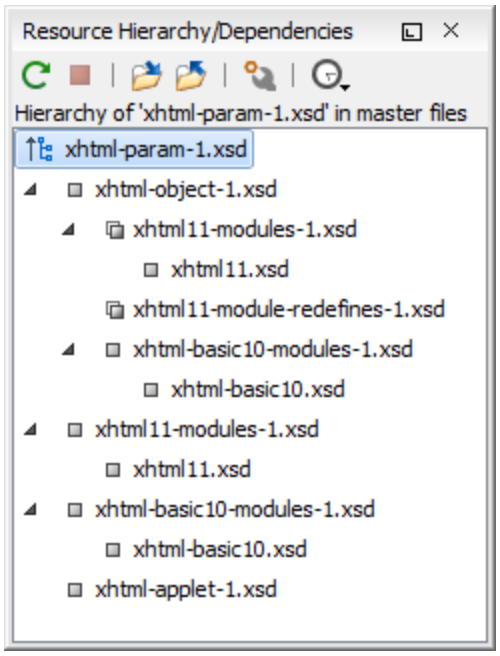


Figure 346: Resource Hierarchy/Dependencies View - Dependencies for `xhtml-param-1.xsd`

The following actions are available in the **Resource Hierarchy/Dependencies** view:

Refresh

Refreshes the Hierarchy/Dependencies structure.

Stop

Stops the hierarchy/dependencies computing.

Show Hierarchy

Allows you to choose a resource to compute the hierarchy structure.

Show Dependencies

Allows you to choose a resource to compute the dependencies structure.

Configure

Allows you to configure a scope to compute the dependencies structure. There is also an option for automatically using the defined scope for future operations.

History

Provides access to the list of previously computed dependencies. Use the **Clear history** button to remove all items from this list.

The contextual menu contains the following actions:

Open

Opens the resource. You can also double-click a resource in the Hierarchy/Dependencies structure to open it.

Copy location

Copies the location of the resource.

Move resource

Moves the selected resource.

Rename resource

Renames the selected resource.

Show Resource Hierarchy

Shows the hierarchy for the selected resource.

Show Resource Dependencies

Shows the dependencies for the selected resource.

Add to Master Files

Adds the currently selected resource in *the **Master Files** directory*.

Expand All

Expands all the children of the selected resource from the Hierarchy/Dependencies structure.

Collapse All

Collapses all children of the selected resource from the Hierarchy/Dependencies structure.

Tip: When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon .

Note: The **Move resource** or **Rename resource** actions give you the option to *update the references to the resource*.

Related Information:

Working with Modular XML Files in the Master Files Context on page 481

Search and Refactor Operations Scope on page 480

Moving/Renaming XML Schema Resources

You can move and rename a resource presented in the **Resource/Hierarchy Dependencies** view, using the **Rename resource** and **Move resource** refactoring actions from the contextual menu.

When you select the **Rename** action in the contextual menu of the **Resource/Hierarchy Dependencies** view, the **Rename resource** dialog box is displayed. The following fields are available:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references** - Select this option to update the references to the resource you are renaming.

When you select the **Move** action from the contextual menu of the **Resource/Hierarchy Dependencies** view, the **Move resource** dialog box is displayed. The following fields are available:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, in accordance with the new location and name.

If the **Update references of the moved resource(s)** option is selected, a **Preview** option (which opens the **Preview** dialog box) is available for both actions. The **Preview** dialog box presents a list with the resources that are updated.

Component Dependencies View for XML Schema

The **Component Dependencies** view allows you to spot the dependencies for the selected component of an XML Schema, a *Relax NG schema*, a *NVDL schema* or an *XSLT stylesheet*. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

If you want to see the dependencies of a schema component:

- Select the desired schema component in the editor.
- Choose the **Component Dependencies** action from the contextual menu.

The action is available for all named components (for example, elements or attributes).

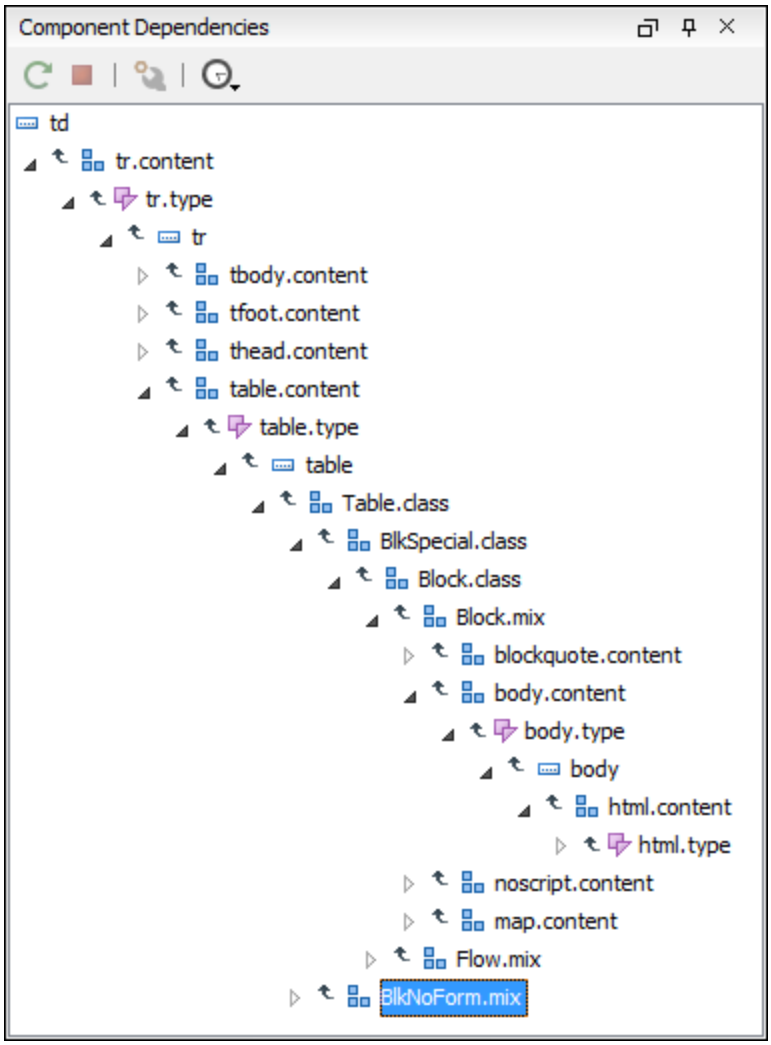


Figure 347: Component Dependencies View - Hierarchy for `xhtml11.xsd`

In the **Component Dependencies** view the following actions are available in the toolbar:

Refresh

Refreshes the dependencies structure.

Stop

Stop the dependencies computing.

Configure

Allows you to configure a search scope to compute the dependencies structure.

History

Contains a list of previously executed dependencies computations.

The contextual menu contains the following actions:

Go to First Reference

Selects the first reference of the referenced component from the current selected component in the dependencies tree.

Go to Component

Shows the definition of the currently selected component in the dependencies tree.

Tip: If a component contains multiple references to another components, a small table is displayed that contains all these references. When a recursive reference is encountered, it is marked with a special icon .

Related Information:

Search and Refactor Operations Scope on page 480

Highlight Component Occurrences

When a component (for example types, elements, attributes) is found at current cursor position, Oxygen XML Editor performs a search over the entire document to find the component declaration and all its references. When found, they are highlighted both in the document and in the stripe bar, at the right side of the document. Customizable colors are used: one for the component definition and another one for component references. Occurrences are displayed until another component is selected and a new search is performed. All occurrences are removed when you start to edit the document.

This feature is on by default. To configured it, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Mark Occurrences**. A search can also be triggered with the **Search** > **Search Occurrences in File ()** contextual menu action. All matches are displayed in a separate tab of the ***Results** view*.

Searching and Refactoring Actions in XML Schemas

Search Actions

The following search actions can be applied on `attribute`, `attributeGroup`, `element`, `group`, `key`, `unique`, `keyref`, `notation`, `simple`, or `complex` types and are available from the **Search** submenu in the contextual menu of the current editor or from the **Document** > **References** menu:

- **Search References** - Searches all references of the item found at current cursor position in the defined scope, if any. If a scope is defined, but the current edited resource is not part of the range of resources determined by this, a warning dialog box is displayed and you have the possibility to define another search scope.
- **Search References in** - Searches all references of the item found at current cursor position in the file or files that you specify when define a scope in the **Search References** dialog box.
- **Search Declarations** - Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined, but the current edited resource is not part of the range of resources determined by this, a warning dialog box will be displayed and you have the possibility to define another search scope.
- **Search Declarations in** - Searches all declarations of the item found at current cursor position in the file or files that you specify when you define a scope for the search operation.
- **Search Occurrences in File** - Searches all occurrences of the item at the cursor position in the currently edited file.

The following action is available from the contextual menu and the **Document** > **Schema** menu:

- **Show Definition** - Moves the cursor to the definition of the referenced XML Schema item.

  Note: You can also use the **Ctrl + Single-Click (Command + Single-Click on OS X)** shortcut on a reference to display its definition.

Refactoring Actions

The following refactoring actions can be applied on `attribute`, `attributeGroup`, `element`, `group`, `key`, `unique`, `keyref`, `notation`, `simple`, or `complex` types and are available from the **Refactoring** submenu in the contextual menu of the current editor or from the **Document** > **Refactoring** menu:

- **Rename Component** - Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the **Esc** or **Enter** key on your keyboard.
- **Rename Component in** - Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the **Preview** button, you can view the files to be affected by the action.

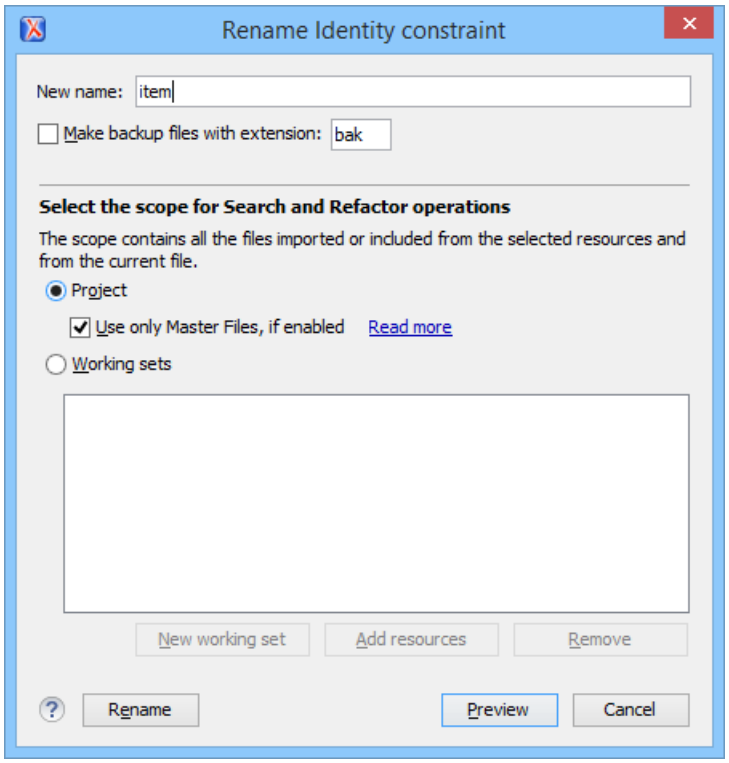


Figure 348: Rename Identity Constraint Dialog Box

Related Information:

Search and Refactor Operations Scope on page 480

XML Schema Quick Assist Support

The *Quick Assist support* improves the development work flow, offering fast access to the most commonly used actions when you edit schema documents.

The *Quick Assist feature* is activated automatically when the cursor is positioned over the name of a component. It is accessible via a yellow bulb icon ( ) placed at the current line in the stripe on the left side of the editor. Also, you can invoke the *quick assist* menu by using the **Alt + 1** (**Meta + Alt + 1** on Mac OS X) keyboard shortcuts.

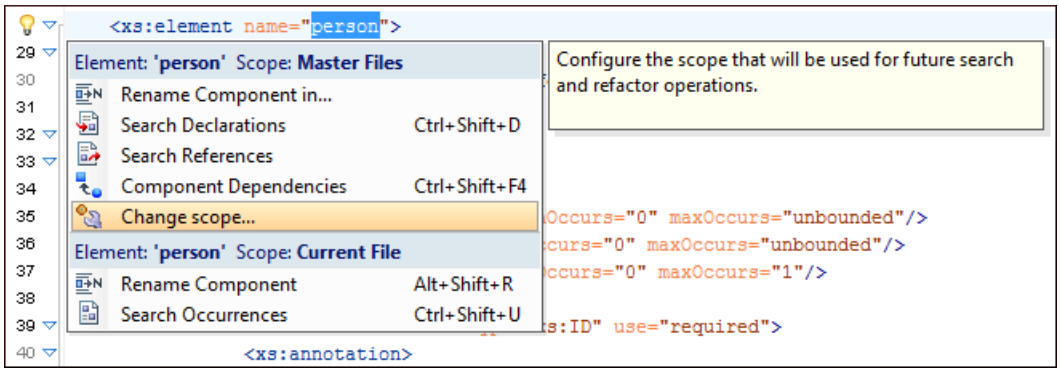


Figure 349: Quick Assist Support

The *Quick Assist* support offers direct access to the following actions:

Rename Component in

Renames the component and all its dependencies.

Search Declarations

Searches the declaration of the component in a predefined scope. It is available only when the context represents a component name reference.

Search References

Searches all references of the component in a predefined scope.

Component Dependencies

Searches the component dependencies in a predefined scope.

Change Scope

Configures the scope that will be used for future search or refactor operations.

Rename Component

Allows you to rename the current component in-place.

Search Occurrences

Searches all occurrences of the component within the current file.

To watch our video demonstration about improving schema development using the **Quick Assist** action set, go to *https://www.oxygenxml.com/demo/Quick_Assist.html*.

Related Information:

Resource Hierarchy / Dependencies View on page 591
Component Dependencies View on page 593
Searching and Refactoring Actions on page 595

Generating Sample XML Files

Oxygen XML Editor offers support to generate sample XML files both from XML schema 1.0 and XML schema 1.1, depending on the XML schema version set in ***XML Schema*** *preferences page*.

To generate sample XML files from an XML Schema, use the **Generate Sample XML Files** action from the **Tools** menu. This action is also available in the contextual menu of the schema *Design mode*. The action opens the **Generate Sample XML Files** dialog box that allows you to configure a variety of options for generating the files.

For more information about this tool, watch our video demonstration about generating sample XML files at *https://www.oxygenxml.com/demo/Generate_Sample_XML_Files.html*.

The **Generate Sample XML Files** dialog box contains three tabs with various configurable options. Default values for these options can be set in the ***Sample XML Files Generator*** *preferences page*. You can also run the tool from the command line using exported options.

Schema Tab (Generate Sample XML Files Tool)

The **Generate Sample XML Files** tool includes a dialog box that allows you to configure a variety of options for generating the XML files. The first set of options are found in the **Schema** tab.

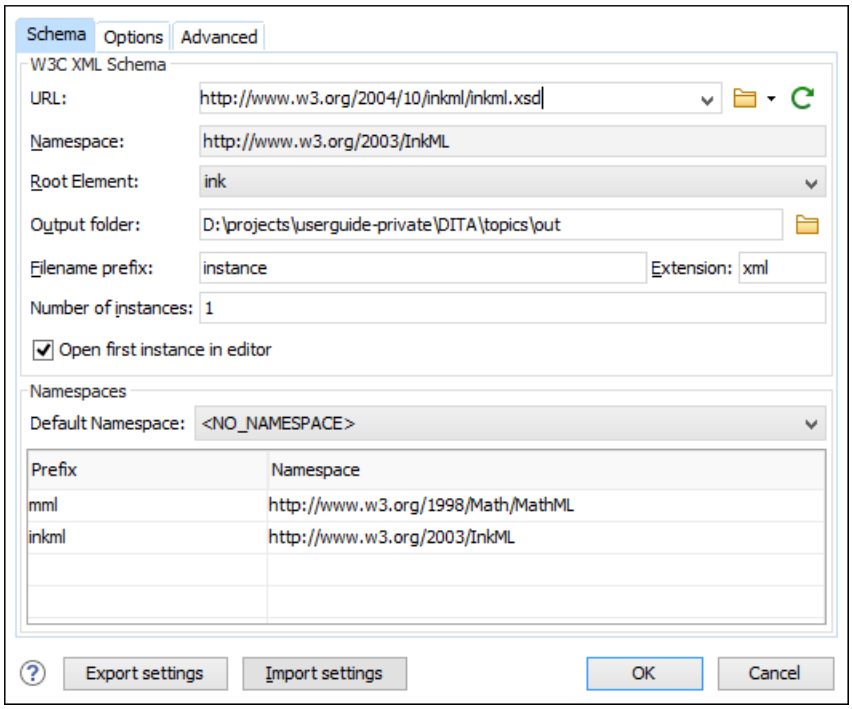


Figure 350: Generate Sample XML Files Dialog Box (Schema Tab)

This tab includes the following options:

URL

Specifies the URL of the Schema location. You can specify the path by using the text field, the history drop-down menu, or the browsing tools in the **Browse** drop-down list.

Namespace

Displays the namespace of the selected schema.

Root Element

After the schema is selected, this drop-down menu is populated with all root candidates gathered from the schema. Choose the root of the output XML documents.

Output folder

Path to the folder where the generated XML instances will be saved.

Filename prefix and Extension

You can specify the prefix and extension for the file name that will be generated. Generated file names have the following format: `prefixN.extension`, where N represents an incremental number from 0 up to the specified ***Number of instances***.

Number of instances

The number of XML files to be generated.

Open first instance in editor

When selected, the first generated XML file is opened in the editor.

Namespaces section

You can specify the **Default Namespace**, as well as the prefixes for the namespaces.

Export settings

Use this button to save the current settings for future use.

Import settings

Use this button to load previously exported settings.

You can click **OK** at any point to generate the sample XML files.

Options Tab (Generate Sample XML Files Tool)

The **Generate Sample XML Files** tool includes a dialog box that allows you to configure a variety of options for generating the XML files. The **Options** tab allows you to set specific options for namespaces and elements.

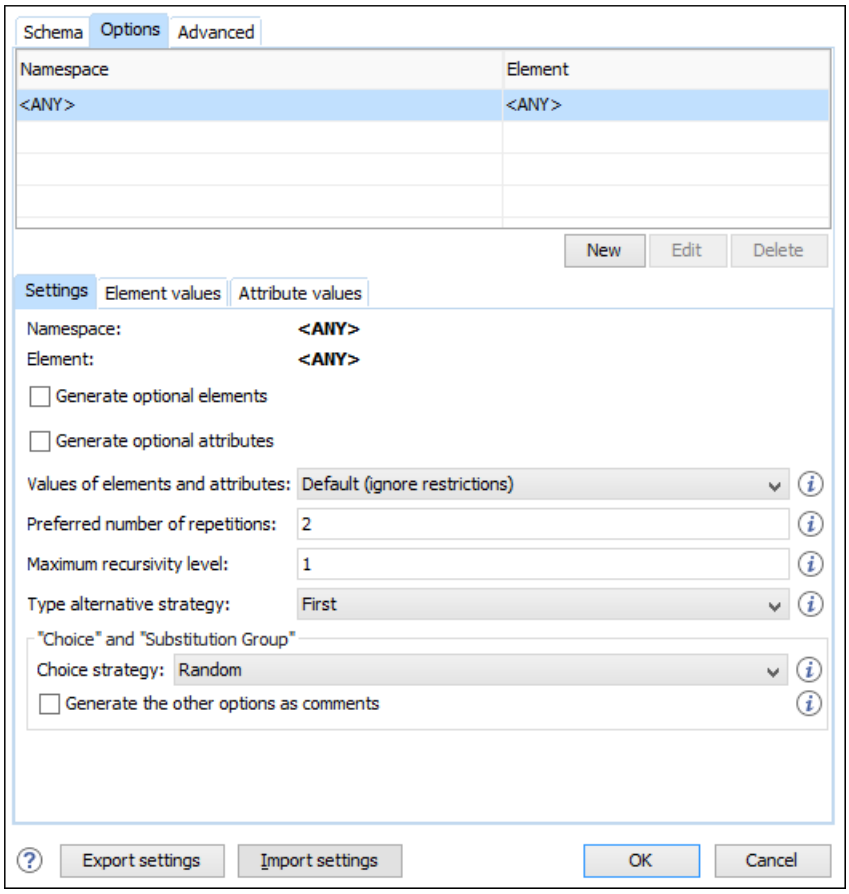


Figure 351: Generate Sample XML Files Dialog Box (Options Tab)

This tab includes the following options:

Namespace / Element table

Allows you to set a namespace for each element name that appears in an XML document instance. The following prefix-to-namespace associations are available:

- All elements from all namespaces (`<ANY>` - `<ANY>`). This is the default setting.
- All elements from a specific namespace.
- A specific element from a specific namespace.

Settings subtab

Namespace

Displays the namespace specified in the table at the top of the dialog box.

Element

Displays the element specified in the table at the top of the dialog box.

Generate optional elements

When selected, all elements are generated, including the optional ones (having the `minOccurs` attribute set to 0 in the schema).

Generate optional attributes

When selected, all attributes are generated, including the optional ones (having the `use` attribute set to `optional` in the schema).

Values of elements and attributes

Controls the content of generated attribute and element values. The following choices are available:

- **None** - No content is inserted.
- **Default** - Inserts a default value depending of data type descriptor of the particular element or attribute. The default value can be either the data type name or an incremental name of the attribute or element (according to the global option from the **Sample XML Files Generator** preferences page). Note that type restrictions are ignored when this option is selected. For example, if an element is of a type that restricts an **xs:string** with the **xs:maxLength** facet to allow strings with a maximum length of 3, the XML instance generator tool may generate string element values longer than 3 characters.
- **Random** - Inserts a random value depending of data type descriptor of the particular element or attribute.

  Important: If all of the following are true, the **Generate Sample XML Files** tool outputs invalid values:

  - At least one of the restrictions is a `regexp`.
  - The value generated after applying the regexp does not match the restrictions imposed by one of the facets.

Preferred number of repetitions

Allows you to set the preferred number of repeating elements related to `minOccurs` and `maxOccurs` facets defined in the XML Schema.

- If the value set here is between `minOccurs` and `maxOccurs`, then that value is used.
- If the value set here is less than `minOccurs`, then the `minOccurs` value is used.
- If the value set here is greater than `maxOccurs`, then `maxOccurs` is used.

Maximum recursion level

If a recursion is found, this option controls the maximum allowed depth of the same element.

Type alternative strategy

Used for the `xs:alternative` element from XML Schema 1.1. The possible strategies are:

- **First** - The first valid alternative type is always used.
- **Random** - A random alternative type is used.

Choice strategy

Used for `xs:choice` or `substitutionGroup` elements. The possible strategies are:

- **First** - The first branch of `xs:choice` or the head element of `substitutionGroup` is always used.
- **Random** - A random branch of `xs:choice` or a substitute element or the head element of a `substitutionGroup` is used.

Generate the other options as comments

If selected, generates the other possible choices or substitutions (for `xs:choice` and `substitutionGroup`). These alternatives are generated inside comments groups so you can uncomment and use them later. Use this option with care (for example, on a restricted namespace and element) as it may generate large result files.

Element values subtab

Allows you to add values that are used to generate the content of elements. If there are multiple values, then the values are used in a random order.

Attribute values subtab

Allows you to add values that are used to generate the content of attributes. If there are multiple values, then the values are used in a random order.

Export settings

Use this button to save the current settings for future use.

Import settings

Use this button to load previously exported settings.

You can click **OK** at any point to generate the sample XML files.

Advanced Tab (Generate Sample XML Files Tool)

The **Generate Sample XML Files** tool includes a dialog box that allows you to configure a variety of options for generating the XML files. The **Advanced** tab allows you to set some options in regards to output values and performance.

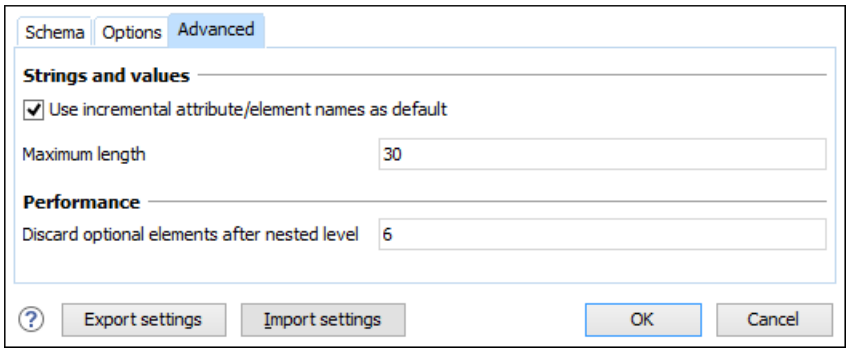


Figure 352: Generate Sample XML Files Dialog Box (Advanced Tab)

This tab includes the following options:

Use incremental attribute / element names as default

If selected, the value of an element or attribute starts with the name of that element or attribute. For example, for an a element the generated values are: a1, a2, a3, and so on. If not selected, the value is the name of the type of that element / attribute (for example: `string`, `decimal`, etc.)

Maximum length

The maximum length of string values generated for elements and attributes.

Discard optional elements after nested level

The optional elements that exceed the specified nested level are discarded. This option is useful for limiting deeply nested element definitions that can quickly result in very large XML documents.

Export settings

Use this button to save the current settings for future use.

Import settings

Use this button to load previously exported settings.

Running the Generate Sample XML Files Tool from the Command Line

The **Generate Sample XML Files** tool can be also used from command line by running the script called `xmlGenerator.bat` (on Windows) / `xmlGenerator.sh` (on Mac OS X / Unix / Linux) located in the Oxygen XML Editor installation folder. The parameters can be set in the dialog box, exported to an XML file on disk with the **Export settings** button, and then reused from command line. With the exported settings file, you can generate the same XML instances from the command line as from the dialog box. For example:

```
xmlGenerator.bat path_of_CFG_file
```

The script can be integrated in an external batch process launched from the command line. The command line parameter of the script is the relative path to the exported XML settings file. The files specified with relative paths in the exported XML settings will be made absolute relative to the folder where the script is run.

The following example shows such an XML configuration file:

```
<settings>
    <schemaSystemId>http://www.w3.org/2001/XMLSchema.xsd</schemaSystemId>
    <documentRoot>schema</documentRoot>
    <outputFolder>D:\projects\output</outputFolder>
    <filenamePrefix>instance</filenamePrefix>
    <filenameExtension>xml</filenameExtension>
    <noOfInstances>1</noOfInstances>
    <openFirstInstance>true</openFirstInstance>
    <defaultNamespace>&lt;NO_NAMESPACE></defaultNamespace>
    <element namespace="&lt;ANY>" name="&lt;ANY>">
        <generateOptionalElements>false</generateOptionalElements>
        <generateOptionalAttributes>false</generateOptionalAttributes>
        <valuesForContentType>DEFAULT</valuesForContentType>
        <preferredNumberOfRepetitions>2</preferredNumberOfRepetitions>
```

```
        <maximumRecursivityLevel>1</maximumRecursivityLevel>
        <choicesAndSubstitutions strategy="RANDOM"
                generateOthersAsComments="false"/>
        <attribute namespace="&lt;ANY>"
                name="&lt;ANY>">
            <attributeValue>attrValue1</attributeValue>
            <attributeValue>attrValue2</attributeValue>
        </attribute>
    </element>
    <element namespace="&lt;NO_NAMESPACE>"
            name="&lt;ANY>">
        <generateOptionalElements>true</generateOptionalElements>
        <generateOptionalAttributes>true</generateOptionalAttributes>
        <valuesForContentType>DEFAULT</valuesForContentType>
        <preferredNumberOfRepetitions>2</preferredNumberOfRepetitions>
        <maximumRecursivityLevel>1</maximumRecursivityLevel>
        <choicesAndSubstitutions strategy="RANDOM"
                generateOthersAsComments="true"/>
        <elementValue>value1</elementValue>
        <elementValue>value2</elementValue>
        <attribute namespace="&lt;ANY>"
                name="&lt;ANY>">
            <attributeValue>attrValue1</attributeValue>
            <attributeValue>attrValue2</attributeValue>
        </attribute>
    </element>
</settings>
```

Generating Documentation for an XML Schema

Oxygen XML Editor can generate detailed documentation for the components of an XML Schema in HTML, PDF, DocBook, or other custom formats. You can select the components and the level of detail. The components are hyperlinked in both HTML and DocBook documents.

Note: You can generate documentation for both XML Schema version 1.0 and 1.1.

To generate documentation for an XML Schema document, select **XML Schema Documentation** from the **Tools** > **Generate Documentation** menu or from the **Generate Documentation** submenu in the contextual menu of the ***Project*** *view*.

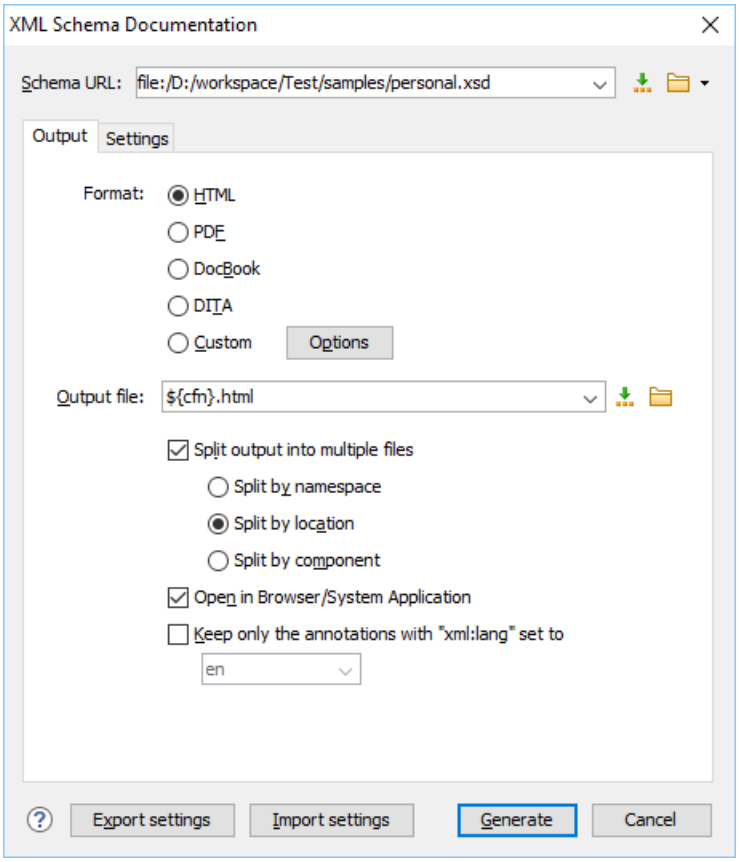


Figure 353: XML Schema Documentation Dialog Box

The **Schema URL** field of the dialog box must contain the full path to the XML Schema (XSD) file for which you want to generate documentation. The schema may be a local or a remote file. You can specify the path to the schema by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.

Output Tab

The following options are available in the **Output** tab:

- **Format** - Allows you to choose between the following formats:
  - **HTML** - The documentation is generated in *HTML output format*.
  - **PDF** - The documentation is generated in *PDF output format*.
  - **DocBook** - The documentation is generated in *DocBook output format*.
  - **DITA** - The documentation is generated in *DITA output format*.
  - **Custom** - The documentation is generated in a *custom output format*, allowing you to control the output. Click the **Options** button to open a **Custom format options** dialog box where you can specify a custom stylesheet for creating the output. There is also an option to **Copy additional resources to the output folder** and you can select the path to the additional **Resources** that you want to copy. You can also choose to keep the intermediate XML files created during the documentation process by deselecting the **Delete intermediate XML file** option.
- **Output file** - You can specify the path of the output file by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.
- **Split output into multiple files** - Instructs the application to split the output into multiple files. You can choose to split them by namespace, location, or component name.
- **Open in Browser/System Application** - Opens the result in the system application associated with the output file type. For DITA and DocBook documents, this option appears as **Open in Editor** and the result will be opened in Oxygen XML Editor (in the current editor).

  Note: To set the browser or system application that will be used, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.
- **Keep only the annotations with xml:lang set to** - The generated output will contain only the annotations with the `xml:lang` attribute set to the selected language. If you choose a primary language code (for example, **en** for English), this includes all its possible variations (**en-us**, **en-uk**, etc.).

Settings Tab

When you generate documentation for an XML schema you can choose what components to include in the output and the details to be included in the documentation.

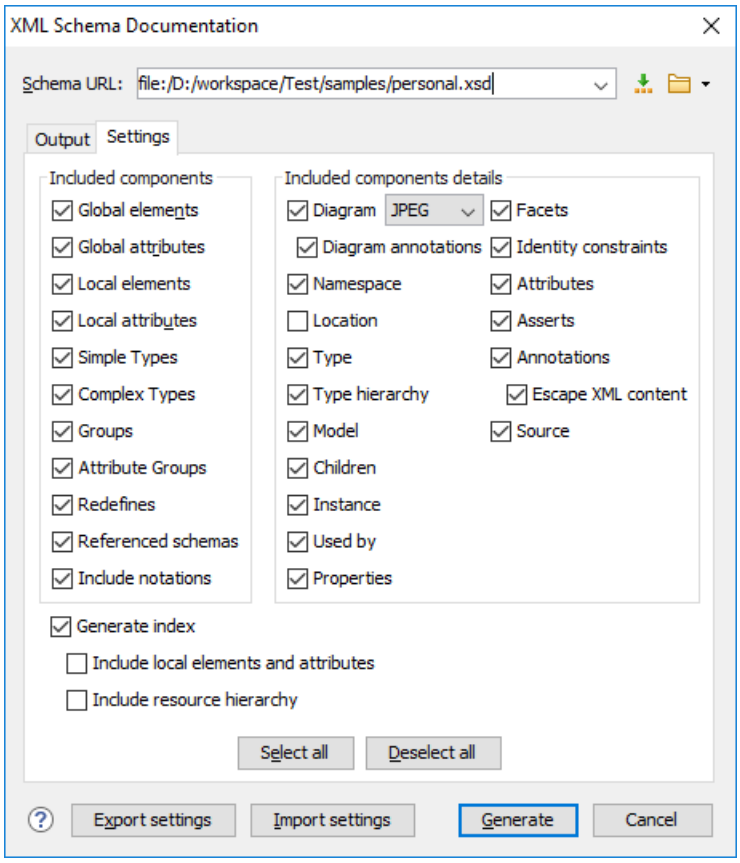


Figure 354: Settings Tab of the XML Schema Documentation Dialog Box

The **Settings** tab allows you to choose whether or not to include the following components: **Global elements**, **Global attributes**, **Local elements**, **Local attributes**, **Simple Types**, **Complex Types**, **Groups**, **Attribute Groups**, **Redefines**, **Referenced schemas**, **Include notations**.

You can choose whether or not to include the following other details:

- **Diagram** - Displays the diagram for each component. You can choose the image format (JPEG, PNG, GIF, SVG) to use for the diagram section. The generated diagrams are dependent on the options from the *Schema Design Properties* page.
  - **Diagram annotations** - This option controls whether or not the annotations of the components presented in the diagram sections are included.
- **Namespace** - Displays the namespace for each component.
- **Location** - Displays the schema location for each component.
- **Type** - Displays the component type if it is not an anonymous one.
- **Type hierarchy** - Displays the types hierarchy.
- **Model** - Displays the model (sequence, choice, all) presented in BNF form. The separator characters that are used depend upon the information item used:
  - `xs:all` - Its children will be separated by space characters.
  - `xs:sequence` - Its children will be separated by comma characters.
  - `xs:choice` - Its children will be separated by / characters.
- **Children** - Displays the list of component's children.
- **Instance** - Displays an XML instance generated based on each schema element.
- **Used by** - Displays the list of all the components that reference the current one. The list is sorted by component type and name.
- **Properties** - Displays some of the component's properties.
- **Facets** - Displays the facets for each simple type

- **Identity constraints** - Displays the identity constraints for each element. For each constraint there are presented the name, type (unique, key, keyref), reference attribute, selector and field(s).
- **Attributes** - Displays the attributes for the component. For each attribute there are presented the name, type, fixed or default value, usage and annotation.
- **Asserts** - Displays the **assert** elements defined in a complex type. The test, XPath default namespace, and annotation are presented for each assert.
- **Annotations** - Displays the annotations for the component. If you choose **Escape XML Content**, the XML tags are present in the annotations.
- **Source** - Displays the text schema source for each component.
- **Generate index** - Displays an index with the components included in the documentation.
  - **Include local elements and attributes** - If selected, local elements and attributes are included in the documentation index.
  - **Include resource hierarchy** - Specifies whether or not the resource hierarchy for an XML Schema documentation is generated. It is deselected by default.

Export settings - Save the current settings in a settings file for further use (for example, with the exported settings file you can generate the same *documentation from the command line interface*.)

Import settings - Reloads the settings from the exported file.

Generate - Use this button to generate the XML Schema documentation.

Related Information:

Customizing the PDF Output of Generated XML Schema Documentation on page 608

Output Formats for Generating XML Schema Documentation

XML Schema documentation can be generated in HTML, PDF, DocBook, or a custom format. You can choose the format from the *Schema Documentation* dialog box. For the PDF and DocBook formats, the option to split the output in multiple files is not available.

HTML Output Format

The XML Schema documentation generated in HTML format contains images corresponding to the same schema definitions as the ones displayed by *the schema diagram editor.* These images are divided in clickable areas that are linked to the definitions of the names of types or elements. The documentation of a definition includes a **Used By** section with links to the other definitions that reference it. If the **Escape XML Content** option is unchecked, the HTML or XHTML tags used inside the `xs:documentation` elements of the input XML Schema for formatting the documentation text (for example, `<b>`, `<i>`, `<u>`, `<ul>`, `<li>`, etc.) are rendered in the generated HTML documentation.

The generated images format is **PNG**. The image of an XML Schema component contains the graphical representation of that component as it is rendered in *the schema diagram panel of the Oxygen XML Editor XSD editor panel.*

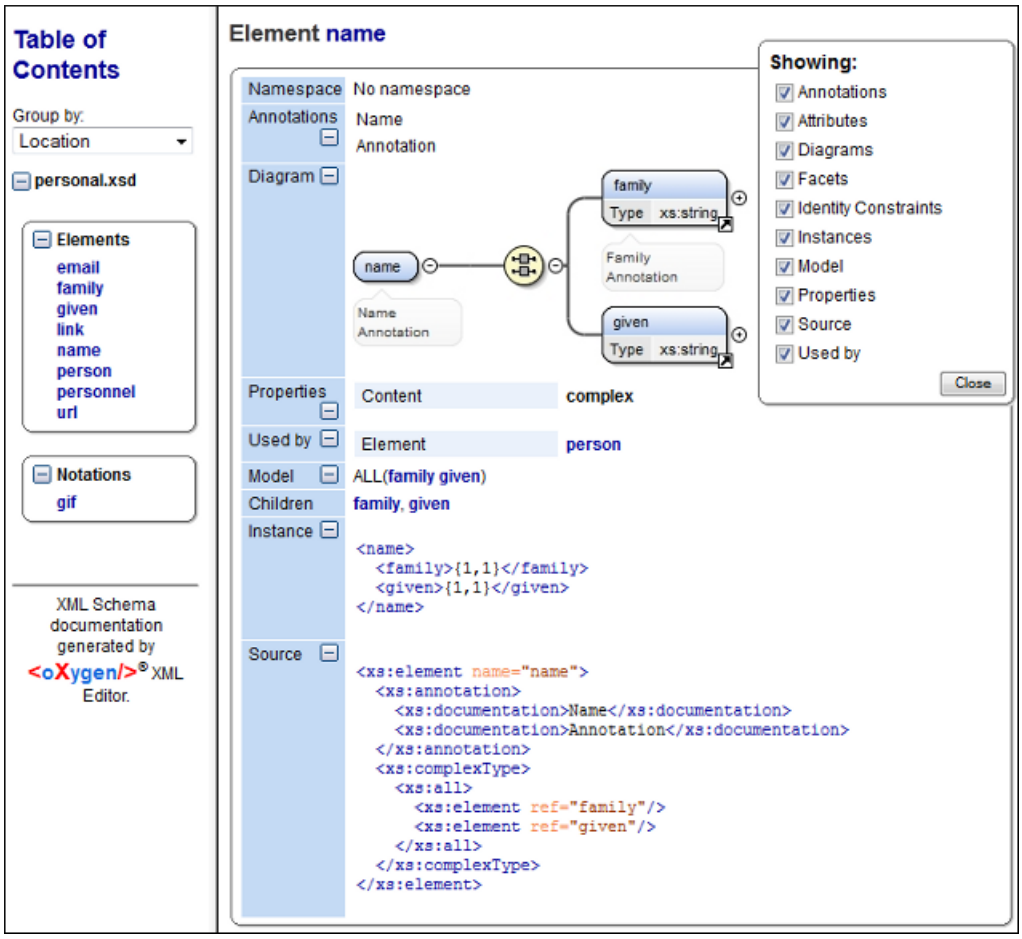


Figure 355: XML Schema Documentation Example

The generated documentation includes a table of contents. You can group the contents by namespace, location, or component type. After the table of contents there is some information about the main, imported, included, and redefined schemas. This information contains the schema target namespace, schema properties (attribute form default, element form default, version), and schema location.

Figure 356: Information About a Schema

If you choose to split the output into multiple files, the table of contents is displayed in the left frame. The contents are grouped in the same mode. If you split the output by location, each file contains a schema description and the components that you have chosen to include. If you split the output by namespace, each file contains information about schemas from that namespace and the list with all included components. If you choose to split the output by component, each file contains information about a schema component.

After the documentation is generated, you can collapse or expand details for some schema components by using the **Showing** options or the ⊟**Collapse** or ⊞**Expand** buttons.

Figure 357: Showing Options

For each component included in the documentation, the section presents the component type follow by the component name. For local elements and attributes, the name of the component is specified as *parent name/ component name*. You can easily go to the parent documentation by clicking the parent name.

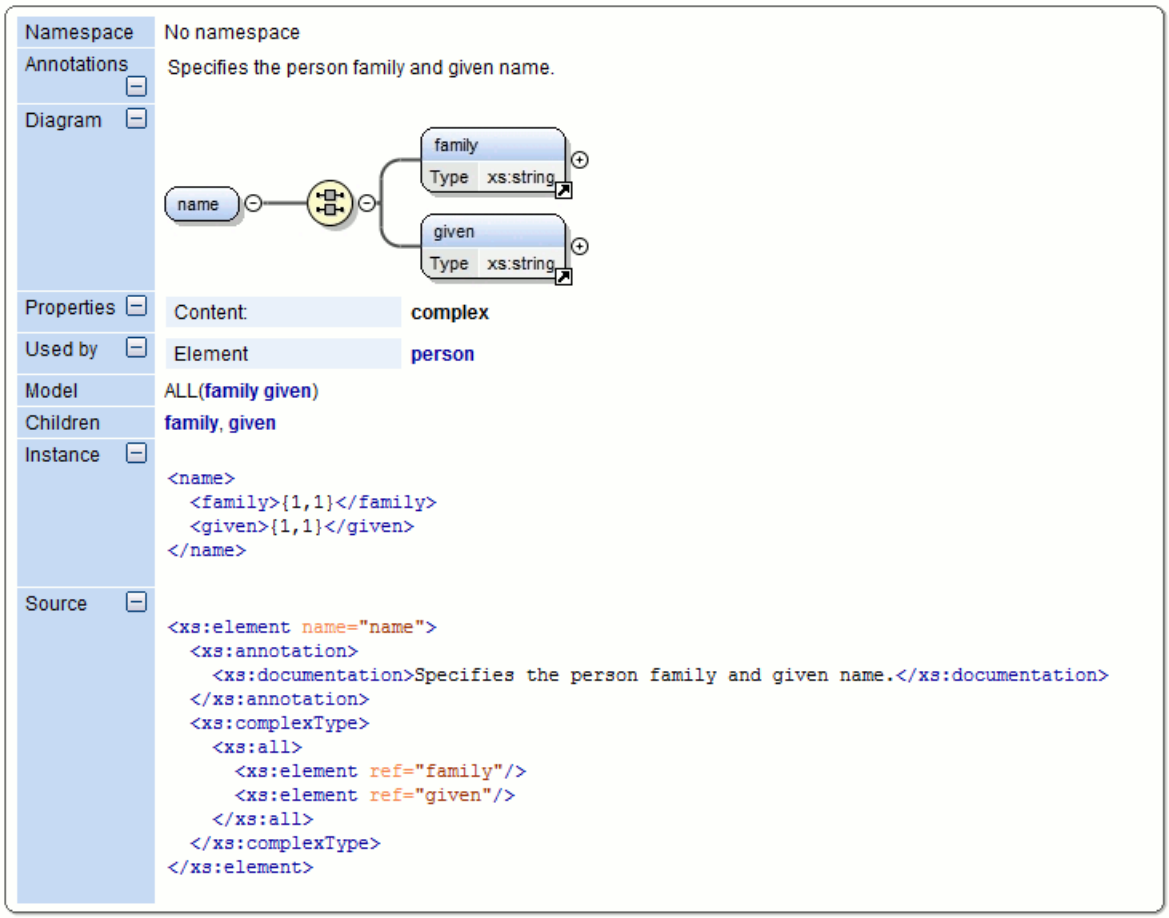


Figure 358: Documentation for a Schema Component

If the schema contains imported or included modules, their dependencies tree is generated in the documentation.

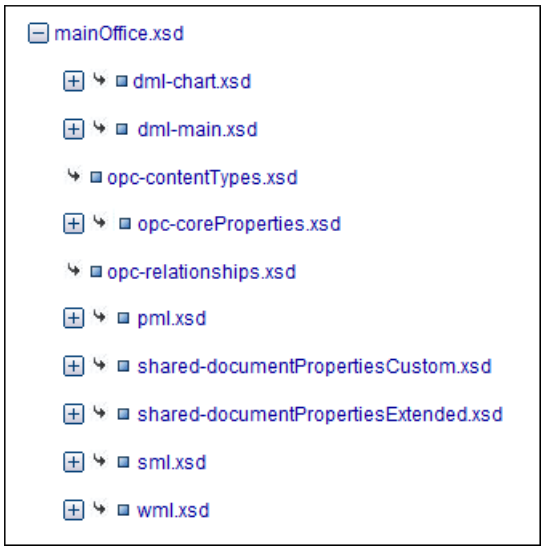


Figure 359: Example: Generated Documentation

PDF Output Format

For the PDF output format, the documentation is generated in DocBook format and a transformation using the FOP processor is applied to obtain the PDF file. To configure the FOP processor, see the *FO Processors* preferences page.

For information about customizing the PDF output, see *Customizing the PDF Output of Generated XML Schema Documentation* on page 608.

DocBook Output Format

If you generate the documentation in DocBook output format, the documentation is generated as a DocBook XML file. You can then apply a *predefined DocBook transformation scenario* (such as, *DocBook PDF* or *DocBook HTML*) on the output file, or *configure your own transformation scenario* to convert it into whatever format you desire.

DITA Output Format

If you generate the documentation in DITA output format, each element of the schema is converted to a DITA *Topic* and all the generated topics are referenced in a *DITA map* file. You can then *apply a predefined DITA transformation scenario* (such as, *DITA Map PDF* or *DITA Map XHTML*), or *configure your own DITA OT transformation scenario* to convert it into whatever format you desire.

Custom Output Format

For the custom format, you can specify a stylesheet to transform the intermediary XML file generated in the documentation process. You have to edit your stylesheet based on the schema `xsdDocSchema.xsd` from `[OXYGEN_INSTALL_DIR]/frameworks/schema_documentation`. You can create a custom format starting from one of the stylesheets used in the predefined HTML, PDF, DocBook, and DITA formats. These stylesheets are available in `[OXYGEN_INSTALL_DIR]/frameworks/schema_documentation/xsl`.

When using a custom format you can also copy additional resources into the output folder and choose to keep the intermediate XML files created during the documentation process.

Important: If you create a custom format for DITA, you must select the ***Split output into multiple files*** *option in the* ***Output*** *tab* and choose **Split by component.**

Customizing the PDF Output of Generated XML Schema Documentation

To customize the PDF output of the generated XML Schema documentation, use the following procedure:

1. Customize the `[OXYGEN_INSTALL_DIR]/frameworks/schema_documentation/xsl/xsdDocDocbook.xsl` stylesheet to include the content that you want to add in the PDF output. Add the

content in the XSLT template with the `match="schemaDoc"` attribute between the `info` and `xsl:apply-templates` elements, as commented in the following example:

```
<info>
    <pubdate><xsl:value-of select="format-date(current-date(),
        '[Mn] [D], [Y]', 'en', (), ())"/></pubdate>
</info>
   <!-- Add the XSLT template content with match="schemaDoc" attribute here -->
<xsl:apply-templates select="schemaHierarchy"/>
```

Note: The content that you insert here has to be wrapped in DocBook markup. For details about working with DocBook see the following video demonstration: *https://www.oxygenxml.com/demo/DocBook_Editing_Support.html*.

2. Create an intermediary file that holds the content of your XML Schema documentation.
   a. Go to **Tools** > **Generate Documentation** > **XML Schema Documentation**.
   b. Select **Custom** for the output format and click the **Options** button.
   c. In the **Custom format options** dialog box, do the following:
      a. Enter the customized stylesheet in the **Custom XSL** field (`[OXYGEN_INSTALL_DIR]/frameworks/schema_documentation/xsl/xsdDocDocbook.xsl`).
      b. Select the **Copy additional resources to the output folder** option and leave the default selection in the **Resources** field.
      c. Click **OK**.
   d. When you return to the **XML Schema Documentation** dialog box, just press the **Generate** button to generate a DocBook XML file with an intermediary form of the Schema documentation.
3. Create the final PDF document.
   a. Use the **Configure Transformation Scenario(s)** action from the toolbar or the **Document** > **Transformation** menu, click **New**, and select **XML transformation with XSLT**.
   b. In the **New Scenario** dialog box, go to the **XSL URL** field and choose the `[OXYGEN_INSTALL_DIR]/frameworks/docbook/oxygen/xsdDocDocbookCustomizationFO.xsl` file.
   c. Go to the **FO Processor** tab and select the **Perform FO Processing** and **XSLT result as input** options.
   d. Go to the **Output** tab and select the directory where you want to store the XML Schema documentation output and click **OK**.
   e. Click **Apply Associated**.

Tip: If you want your modifications to be permanent so that you can simply select the PDF output format in the **XML Schema Documentation** dialog box, rather than configuring a custom format each time you generate this documentation, follow this procedure:

1. Create a *JAR* or ZIP file that includes the modified stylesheet (customized in step 1 above), placed in the following directory structure: `builtin/documentation/schema_documentation/xsdDocDocbook.xsl`.
2. Create a new directory named `endorsed` inside the `[OXYGEN_INSTALL_DIR]/lib` directory and place the created *JAR* or ZIP file inside it.
3. Restart Oxygen XML Editor and the PDF output format will now use your customized stylesheet.

Generating XML Schema Documentation From the Command-Line Interface

You can export the settings of the ***XML Schema Documentation** dialog box* to an XML file by pressing the **Export settings** button. With the exported configuration file, you can generate the same documentation from the command-line interface by running the following script:

- `schemaDocumentation.bat` on Windows.
- `schemaDocumentation.sh` on OS X / Unix / Linux.

The script is located in the Oxygen XML Editor installation folder. The script can be integrated in an external batch process launched from the command-line interface. The script accepts a variety command line arguments that allow you to configure the documentation.

The accepted syntax and arguments are as follows:

`schemaDocumentation schemaFile [ [-cfg:configFile] | [ [-out:outputFile] [-format:<value>] [-xsl:<xslFile>] [-split:<value>] [-openInBrowser:<value>] ] | --help | -help | --h | -h ]`

schemaFile

The XML Schema file.

-cfg:configfile

The exported configuration file. It contains the output file, output format options, split method, and some advanced options regarding the included components and components details. If an external configuration file is specified, all other supplied arguments except for the XML Schema file will be ignored.

-out:outputFile

The file where the generated documentation will be saved. By default, it is the name of the schema file with an *html* extension.

-format:<value>

The output format type used when generating the documentation. Possible values are as follows:

- `html` - To generate documentation in HTML format.
- `pdf` - To generate documentation in PDF format.
- `docbook` - To generate documentation in DocBook format.
- `custom` - To generate documentation in a custom format.

-xsl:<xslFile>

The XSL file to be applied on the intermediate XML format. If there is no XSL file provided, the result will be in the HTML format.

-split:<value>

The split method used when generating the documentation. Splitting is recommended for large schemas. Possible values are as follows:

- `none` (default value) - To generate one single output file.
- `namespace` - To generate an output file for every namespace in the schema.
- `component` - To generate an output file for every component in the schema.
- `location` - To generate an output file for every schema location.

-openInBrowser:<value>

Opens the result of the transformation in a browser or system application. Possible values are `true` or `false` (default value).

--help | -help | --h | -h

Displays the available options.

Example:

Example of the script in a Windows command line:

```
schemaDocumentation example.xsd -out:schemaDocumentation.html
    -format:custom -xsl:example.xsl -split:namespace
```

Converting Schema to Another Schema Language

The **Generate/Convert Schema** tool allows you to convert a DTD or Relax NG (full or compact syntax) schema or a set of XML files to an equivalent XML Schema, DTD or Relax NG (full or compact syntax) schema. Where perfect equivalence is not possible due to limitations of the target language, Oxygen XML Editor generates an approximation of the source schema. Oxygen XML Editor uses *Trang multiple format converter* to perform the actual schema conversions.

To use this tool, select the **Generate/Convert Schema (Alt + Shift + C (Command + Alt + C on OS X))** action from the **Tools** menu or from the **Open with** submenu when invoking the contextual menu in the ***Project*** *view*. This action opens the **Generate/Convert Schema** dialog box that allows you to configure various options for conversion.

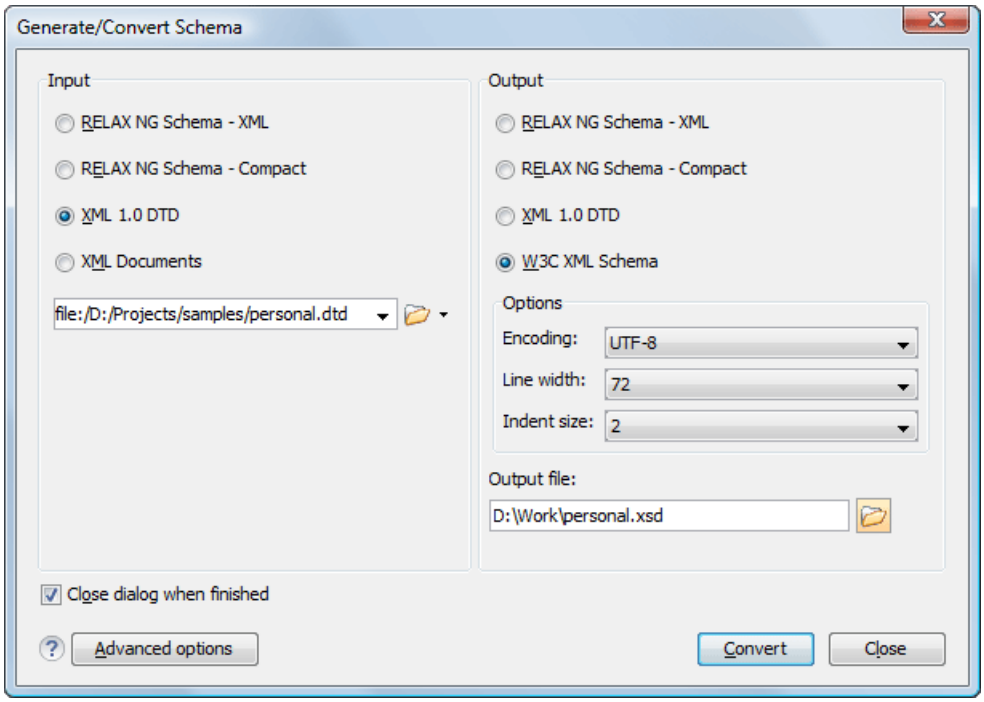


Figure 360: Generate/Convert Schema Dialog Box

The **Generate/Convert Schema** dialog box includes the following options:

Input section

Allows you to select the language of the source schema. If the conversion is based on a set of XML files, rather than just a single XML file, select the **XML Documents** option and use the file selector to add the XML files involved in the conversion.

Output section

Allows you to select the language of the target schema.

Options

You can choose the **Encoding**, the maximum **Line width**, and the **Indent size** (in number of spaces) for one level of indentation.

Output file

Specifies the path for the output file that will be generated.

Close dialog when finished

If you deselect this option, the dialog box will remain opened after the conversion so that you can easily continue to convert more files.

Advanced options

If you select **XML 1.0 DTD** for the input, you can click this button to access more advance options to further fine-tune the conversion. The following advanced options are available:

XML 1.0 DTD Input section

These options apply to the source DTD:

- **xmlns** - Specifies the default namespace, that is the namespace used for unqualified element names.
- **attlist-define** - Specifies how to construct the name of the definition representing an attribute list declaration from the name of the element. The specified value must contain exactly one percent character. This percent character is replaced by the name of element (after colon replacement) and the result is used as the name of the definition.
- **colon-replacement** - Replaces colons in element names with the specified chars when constructing the names of definitions used to represent the element declarations and attribute list declarations in the DTD.

- **any-name** - Specifies the name of the definition generated for the content of elements declared in the DTD as having a content model of ANY.
- **element-define** - Specifies how to construct the name of the definition representing an element declaration from the name of the element. The specified value must contain exactly one percent character. This percent character is replaced by the name of element (after colon replacement) and the result is used as the name of the definition.
- **annotation-prefix** - Default values are represented using an annotation attribute `prefix:defaultValue` where prefix is the specified value and is bound to `http://relaxng.org/ns/compatibility/annotations/1.0` as defined by the RELAX NG DTD Compatibility Committee Specification. By default, the conversion engine will use a for prefix unless that conflicts with a prefix used in the DTD.
- **inline-attlist** - Instructs the application not to generate definitions for attribute list declarations, but instead move attributes declared in attribute list declarations into the definitions generated for element declarations. This is the default behavior when the output language is XSD.
- **strict-any** - Preserves the exact semantics of *ANY* content models by using an explicit choice of references to all declared elements. By default, the conversion engine uses a wildcard that allows any element
- **generate-start** - Specifies whether or not the conversion engine should generate a start element. DTD's do not indicate what elements are allowed as document elements. The conversion engine assumes that all elements that are defined but never referenced are allowed as document elements.
- **xmlns mappings** table - Each row specifies the prefix used for a namespace in the input schema.

W3C XML Schema Output section

This section is available if you select **W3C XML Schema** for the output.

- **disable-abstract-elements** - Disables the use of abstract elements and substitution groups in the generated XML Schema. This can also be controlled using an annotation attribute.
- **any-process-contents** - One of the values: strict, lax, skip. Specifies the value for the `processContents` attribute of any elements. The default is skip (corresponding to RELAX NG semantics) unless the input format is DTD, in which case the default is strict (corresponding to DTD semantics).
- **any-attribute-process-contents** - Specifies the value for the `processContents` attribute of `anyAttribute` elements. The default is skip (corresponding to RELAX NG semantics).

Converting Database to XML Schema

Oxygen XML Editor includes a tool that allows you to create an XML Schema from the structure of a database.

To convert a database structure to an XML Schema, use the following procedure:

1. Select the **Convert DB Structure to XML Schema** action from the **Tools** menu.

   Result: The **Convert DB Structure to XML Schema** dialog box is opened and your current database connections are displayed in the **Connections** section.
2. If the database source is not listed, click the **Configure Database Sources** button to open the ***Data Sources*** *preferences page* where you can configure data sources and connections.
3. In the **Format for generated schema** section, select one of the following formats:
   - **Flat schema** - A flat structure that resembles a tree-like view of the database without references to elements.
   - **Hierarchical schema** - Display the table dependencies visually, in a type of tree view where dependent tables are shown as indented child elements in the content model. Select this option if you want to configure the database columns of the tables to be converted.
4. Click **Connect**.

   Result: The database structure is listed in the **Select database tables** section according to the format you chose.
5. Select the database tables that you want to be included in the XML Schema.
6. If you selected **Hierarchical schema** for the format, you can configure the database columns.

   a. Select the database column you want to configure.

b. In the **Criterion** section you can choose to convert the selected database column as an **Element**, **Attribute**, or to be **Skipped** in the resulting XML Schema.
c. You can also change the name of the selected database column by changing it in the **Name** text field.

7. Click **Generate XML Schema**.

Result: The database structure is converted to an XML Schema and it is opened for viewing and editing.

Flatten an XML Schema

You can organize an XML schema linked by `xs:include` and `xs:import` statements on several levels. In some cases, working on such a schema as if it were a single file is more convenient than working on multiple files separately. The **Flatten Schema** operation allows you to flatten an entire hierarchy of XML schemas. Starting with the main XML schema, Oxygen XML Editor calculates its hierarchy by processing the `xs:include` and `xs:import` statements.

The **Flatten Schema** action is available from the **Tools** menu or the contextual menu in **Text** mode. The action opens the **Flatten Schema** dialog box that allows you to configure the operation.

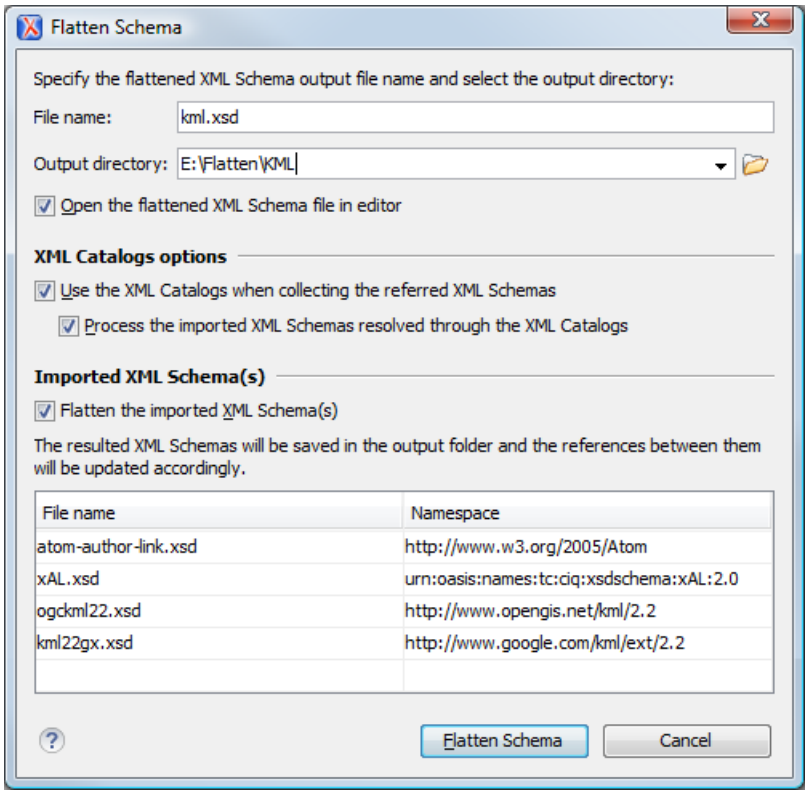


Figure 361: Flatten Schema Dialog Box

For the main schema file and for each imported schema, a new flattened schema is generated in the specified output folder. These schemas have the same name as the original ones.

Note: If necessary, the operation renames the resulted schemas to avoid duplicated file names.

A flattened XML schema is obtained by recursively adding the components of the included schemas into the main one. This means Oxygen XML Editor replaces the `xs:include`, `xs:redefine`, and `xs:override` elements with the ones coming from the included files.

Options in the Flatten Schema Dialog Box

The following options are available in the **Flatten Schema** dialog box:

File name

The name of the output file.

Output directory

The path of the output directory where the flattened schema file will be saved.

Open the flattened XML Schema file in editor

Opens the main flattened schema in the editing area after the operation completes.

Use the XML Catalogs when collecting the referenced XML Schemas

Enables the imported and included schemas to be resolved through the available *XML Catalogs*.

Note: Changing this option triggers the recalculation of the dependencies graph for the main schema.

Process the imported XML Schemas resolved through the XML Catalogs

Specifies whether or not the imported schemas that were resolved through an *XML Catalog* are also processed.

Flatten the imported XML Schema(s)

Specifies whether or not the imported schemas are flattened.

Note: For the schemas skipped by the flatten operation, no files are created in the output folder and the corresponding import statements remain unchanged.

Flatten Schema from the Command Line

The **Flatten Schema** tool can be also ran from command line by using the following command:

- `flattenSchema.bat` on Windows
- `sh flattenSchemaMac.sh` on OS X
- `sh flattenSchema.sh` on Unix/Linux

The command line accepts the following parameters:

- `-in:inputSchemaURL` - The input schema URL.
- `-outDir:outputDirectory` - The directory where the flattened schemas should be saved.
- `-flattenImports:<boolean_value>` - Controls whether or not the imported XML Schemas should be flattened. The default value `true`.
- `-useCatalogs:<boolean_value>` - Controls if the references to other XML Schemas should be resolved through the available *XML Catalogs*. The default value `false`.
- `-flattenCatalogResolvedImports:<boolean_value>` - Controls whether or not the imported schemas that were resolved through the *XML Catalogs* should be flattened. The default value is `true`.

  Note: This option is used only when `-useCatalogs` is set to `true`.
- `-verbose` - Provides information about the current flatten XML Schema operation.
- `--help | -help | --h | -h` - Prints the available parameters for the operation.

Example: Command Line Example for Windows

```
flattenSchema.bat -in:http://www.w3.org/MarkUp/SCHEMA/xhtml11.xsd
   -outDir:mySchemas/flattened/xhtml -flattenImports:true -useCatalogs:true
      -flattenCatalogResolvedImports:true -verbose
```

Example: Command Line Example for OS X

```
sh flattenSchemaMac.sh -in:http://www.w3.org/MarkUp/SCHEMA/xhtml11.xsd
    -outDir:mySchemas/flattened/xhtml -flattenImports:true -useCatalogs:true
       -flattenCatalogResolvedImports:true -verbose
```

Example: Command Line Example for Unix/Linux

```
sh flattenSchema.sh -in:http://www.w3.org/MarkUp/SCHEMA/xhtml11.xsd
     -outDir:mySchemas/flattened/xhtml -flattenImports:true -useCatalogs:true
        -flattenCatalogResolvedImports:true -verbose
```

XML Schema Regular Expressions Builder

The XML Schema regular expressions builder allows you to test regular expressions on a fragment of text as they are applied to an XML instance document. Start the tool by selecting **XML Schema Regular Expressions Builder** from the **Tools** menu.

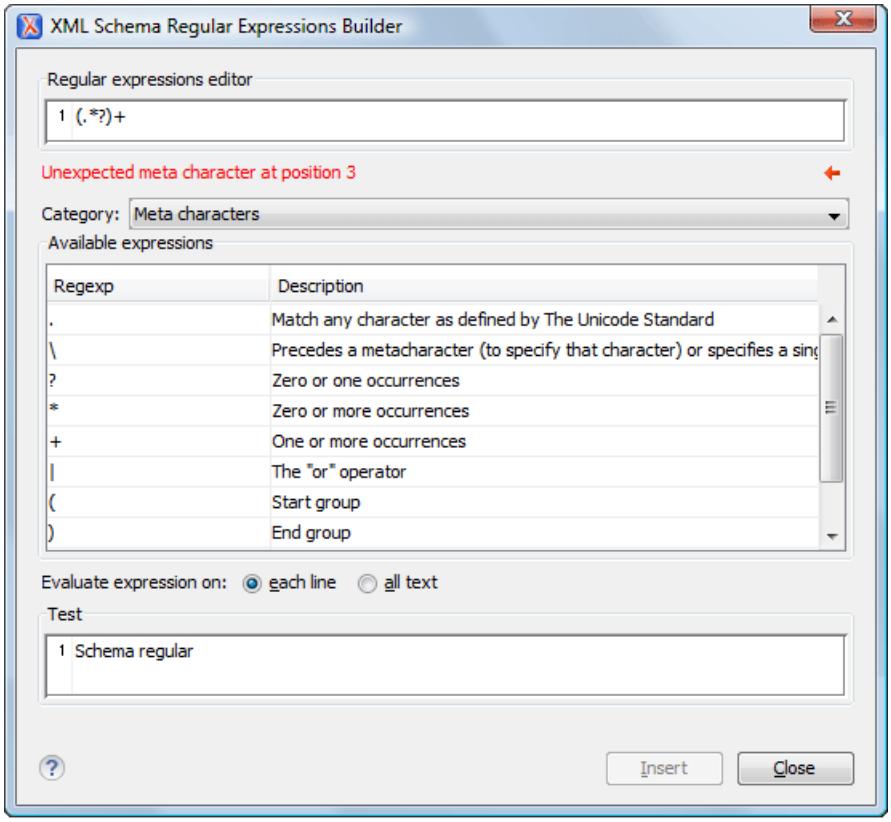


Figure 362: XML Schema Regular Expressions Builder Dialog Box

The dialog box contains the following:

Regular expressions editor

Allows you to edit the regular expression to be tested and used. Content completion is available and presents a list with all the predefined expressions. It is triggered by pressing **Ctrl + Space (Command + Space on OS X)**.

Error display area

If the edited regular expression is incorrect, an error message will be displayed here. The message contains the description and the exact location of the error. Also, clicking the quick navigation button (←) highlights the error inside the regular expression.

Category

You can choose from several categories of predefined expressions. The selected category influences the displayed expressions in the **Available expressions** table.

Available expressions

This table includes the available regular expressions and a short description for each of them. The set of expressions depends on the category selected in the previous **Category** combo box. You can add an expression in the **Regular expressions editor** by double-clicking the expression row in the table. You will notice that in the case of **Character categories** and **Block names**, the expressions are also listed in complementary format.

Evaluate expression on

You can choose between two options:

- **Evaluate expression on each line** - The edited expression will be applied on each line in the **Test** area.
- **Evaluate expression on all text** - The edited expression will be applied on the whole text.

Test

A text editor that allows you to enter a text sample for which the regular expression will be applied. All matches of the edited regular expression will be highlighted.

After editing and testing your regular expression you can insert it in the current editor. The **Insert** button will become active when an editor is opened in the background and there is an expression in the **Regular expressions editor**.

The regular expression builder cannot be used to insert regular expressions in the ***Grid*** *mode* or *schema* ***Design*** *mode*. Accordingly, the **Insert** button will be not available if the current document is edited in these modes.

Note: Some regular expressions may indefinitely block the Java Regular Expressions engine. If the execution of the regular expression does not end in about five seconds, the application displays a dialog box that allows you to interrupt the operation.

XML Schema 1.1

Oxygen XML Editor offers full support for XML Schema 1.1, including:

- XML Documents *Validation* and *Content Completion* based on XML Schema 1.1.
- XML Schema 1.1 *Validation* and *Content Completion*.
- Editing XML Schema 1.1 files in the *Schema* ***Design*** *mode*.
- The *Flatten Schema* action.
- *Resource Hierarchy/Dependencies* and *Refactoring Actions*.
- *Master files*.
- *Generating Documentation for XML Schema 1.1*.
- Support for generating XML instances based on XML Schema.
- Support for validating XML documents with an NVDL schema that contains an XML Schema 1.1 validation step.

  Note: To enable XML Schema 1.1 validation in NVDL, you need to pass the following option to the validation engine to specify the schema version: `http://www.thaiopensource.com/validate/xsd-version` (the possible values are `1.0` or `1.1`.

  Tip: To enable the full XPath expression in assertions and type alternatives, you need to set the `http://www.thaiopensource.com/validate/full-xpath` option.

XML Schema 1.1 is a superset of XML Schema 1.0, that offers lots of new powerful capabilities.

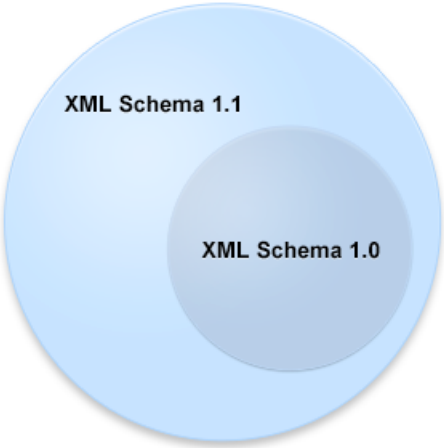


Figure 363: XML Schema 1.1

The significant new features in XSD 1.1 are:

- **Assertions** - Support to define assertions against the document content using XPath 2.0 expressions (an idea borrowed from Schematron).
- **Conditional type assignment** - The ability to select the type of schema an element is validated against, based on the values of the attribute of the element.

- **Open content** - Content models can use the `openContent` element to specify content models with *open content*. These content models allow elements not explicitly mentioned in the content model to appear in the document instance. It is as if wildcards were automatically inserted at appropriate points within the content model. A default may be set that causes all content models to be open unless specified otherwise.

To see the complete list with changes since version 1.0, go to *http://www.w3.org/TR/xmlschema11-1/#ch_specs*.

XML Schema 1.1 is intended to be mostly compatible with XML Schema 1.0 and to have approximately the same scope. It also addresses bug fixes and brings improvements that are consistent with the constraints on scope and compatibility.

Note: An XML document conforming to a 1.0 schema can be validated using a 1.1 validator, but an XML document conforming to a 1.1 schema may not validate using a 1.0 validator.

If you are constrained to use XML Schema 1.0 (for example, if you develop schemas for a server that uses an XML Schema 1.0 validator that cannot be updated), change the default XML Schema version to 1.0. If you keep the default XML Schema version set to 1.1, the *Content Completion Assistant* presents XML Schema 1.1 elements that you can insert accidentally in an 1.0 XML Schema. So even if you make a document invalid conforming with XML Schema 1.0, the validation process does not signal any issues.

To change the default XML Schema version, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XML Parser** > **XML Schema**.

To watch our video demonstration about the XML Schema 1.1 support, go to *https://www.oxygenxml.com/demo/XML_Schema_11.html*.

Related Information:

Setting the XML Schema Version on page 617

Setting the XML Schema Version

Oxygen XML Editor lets you set the version of the XML Schema you are editing either in the **XML Schema** preferences page, or through the versioning attributes. If you want to use the versioning attributes, set the *minVersion* and *maxVersion* attributes, from the *http://www.w3.org/2007/XMLSchema-versioning* namespace, on the *schema* root element.

Note: The versioning attributes take priority over the XML Schema version defined in the preferences page.

Table 37: Using the *minVersion* and *maxVersion* Attributes to Set the XML Schema Version

Versioning Attributes	XML Schema Version
minVersion = "1.0" *maxVersion* = "1.1"	1.0
minVersion = "1.1"	1.1
minVersion = "1.0" *maxVersion* = greater than "1.1"	The XML Schema version defined in the ***XML Schema** preferences page*
Not set in the XML Schema document	The XML Schema version defined in the ***XML Schema** preferences page*

To change the XML Schema version of the current document, use the **Change XML Schema version** action from the contextual menu. This is available both in the **Text** mode, and in the **Design** mode and opens the **Change XML Schema version** dialog box. The following options are available:

- **XML Schema 1.0** - Inserts the *minVersion* and *maxVersion* attributes on the *schema* element and gives them the values "1.0" and "1.1" respectively. Also, the namespace declaration (*xmlns:vc=http://www.w3.org/2007/XMLSchema-versioning*) is inserted automatically if it does not exist.
- **XML Schema 1.1** - Inserts the *minVersion* attribute on the *schema* element and gives it the value "1.1". Also, removes the *maxVersion* attribute if it exists and adds the versioning namespace (*xmlns:vc=http://www.w3.org/2007/XMLSchema-versioning*) if it is not declared.
- **Default XML Schema version** - Removes the *minVersion* and *maxVersion* attributes from the *schema* element. The default schema version, defined in the **XML Schema** preferences page, is used.

Note: The **Change XML Schema version** action is also available in the informative panel presented at the top of the edited XML Schema. If you close this panel, it will no longer appear until you restore Oxygen XML Editor to its default options.

Oxygen XML Editor automatically uses the version set through the versioning attributes, or the default version if the versioning attributes are not declared, when proposing content completion elements and validating an XML Schema. Also, the XML instance validation against an XML Schema is aware of the versioning attributes defined in the XML Schema.

When you generate sample XML files from an XML Schema, Oxygen XML Editor takes into account the *minVersion* and *maxVersion* attributes defined in the XML Schema.

Related Information:

XML Schema 1.1 on page 616

Editing XQuery Documents

XQuery is the query language for XML and is officially defined by *a W3C Recommendation document*. The many benefits of XQuery include:

- XQuery allows you to work in one common model no matter what type of data you are working with: relational, XML, or object data.
- XQuery is ideal for queries that must represent results as XML, to query XML stored inside or outside the database, and to span relational and XML sources.
- XQuery allows you to create many different types of XML representations of the same data.
- XQuery allows you to query both relational sources and XML sources, and create one XML result.

Related Information:

XQuery and Databases on page 1115

XQuery Validation

With Oxygen XML Editor, you can validate your documents before using them in your transformation scenarios. The validation uses the Saxon 9.7.0.15 PE, EE, or HE processor, or you can use some database engines (such as MarkLogic or eXist) if you installed them. Any other XQuery processor that offers an *XQJ API implementation* can also be used. This is in conformance with *the XQuery Working Draft*. The processor is used in two cases: validation of the expression and execution. Although the execution implies a validation, it is faster to check the expression syntactically, without executing it. The errors that occurred in the document are presented in the messages view at the bottom of editor window, with a full description message. As with all error messages, if you click an entry, the line where the error appeared is highlighted.

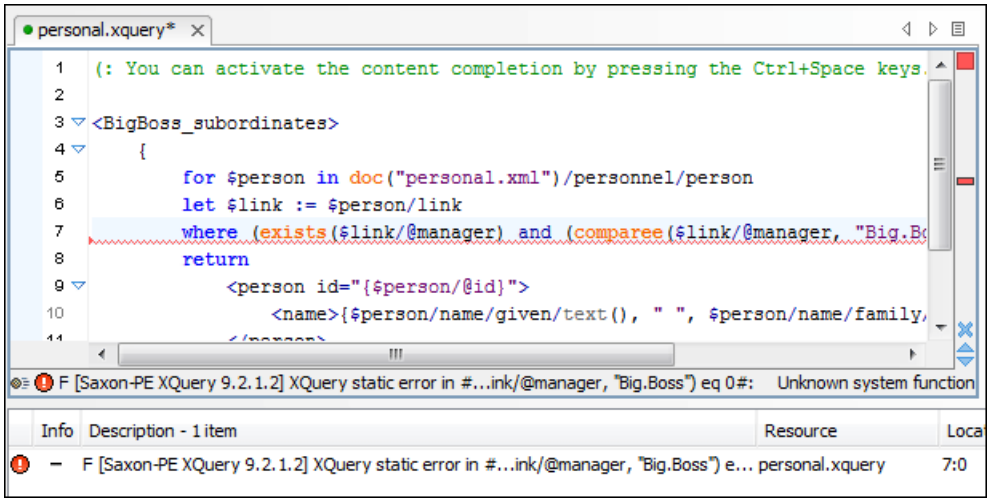


Figure 364: XQuery Validation

Note: If you choose a processor that does not support XQuery validation, Oxygen XML Editor displays a warning when trying to validate.

The **Validation options** button, available in the **Document** > **Validate** menu, allows quick access to the *XQuery options* in the Oxygen XML Editor preferences.

When you open an XQuery document from a connection that supports validation (for example, MarkLogic, or eXist), by default Oxygen XML Editor uses this connection for validation. If you open an XQuery file using a MarkLogic connection, the validation resolves imports better.

Content Completion in XQuery

Oxygen XML Editor provides content completion for keywords and all known XQuery functions and operators. The *Content Completion Assistant* can be manually activated with the **(Ctrl (Meta on Mac OS)+Space)** shortcut. The functions and operators are presented together with a description of the parameters and functionality, depending on the validation or transformation engine.

For some supported database engines such as MarkLogic, eXist, and Berkeley DB, the content completion list offers the specific XQuery functions implemented by that engine. This feature is available when the XQuery file has an associated transformation scenario that uses one of these database engines or the XQuery validation engine is set to one of them via a validation scenario or in the *XQuery Preferences* page. For more information about the support for working with XQuery with regards to databases, see *XQuery and Databases* on page 1115.

The extension functions included in the Saxon engine are available on content completion if one of the following conditions are true:

- The edited file has a transformation scenario associated that uses as transformation engine Saxon 9.7.0.15 PE or Saxon 9.7.0.15 EE.
- The edited file has a validation scenario associated that use as validation engine Saxon 9.7.0.15 PE or Saxon 9.7.0.15 EE.
- The validation engine specified in *Preferences* is Saxon 9.7.0.15 PE or Saxon 9.7.0.15 EE.

If the Saxon namespace (*http://saxon.sf.net*) is mapped to a prefix, the functions are presented using this prefix. Otherwise, the default prefix for the Saxon namespace (`saxon`) is used.

If you want to use a function from a namespace mapped to a prefix, just type that prefix and the content completion displays all the XQuery functions from that namespace. When the default namespace is mapped to a prefix, the XQuery functions from this namespace offered by content completion are also prefixed. Otherwise, only the function name being used.

The content completion pop-up window presents all the variables and functions from both the edited XQuery file and its imports.

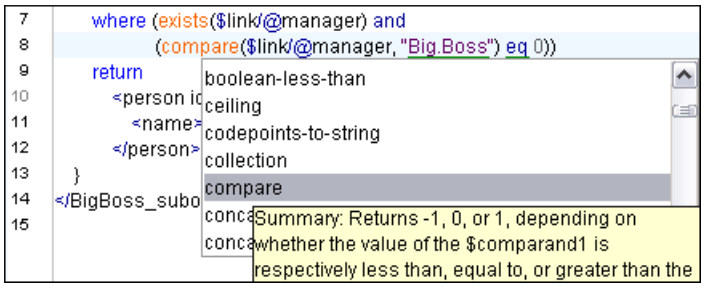


Figure 365: XQuery Content Completion

Syntax Highlighting in XQuery

Oxygen XML Editor supports syntax highlighting in XQuery documents to improve the readability of the content and reduce the time it takes to internalize the semantics of the structured content.

To customize the colors or styles used for the syntax highlighting colors for XQuery files, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)**.*

2. Go to ***Editor*** > ***Syntax Highlight***.
3. Select and expand the **XQuery/XPath** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

Related Information:

Customize Syntax Highlight colors on page 104

Formatting and Indenting XQuery Documents

Editing XQuery documents may lead to large chunks of content that are not easily readable by human audience. Also, each developer may have a particular way of writing XQuery code. Oxygen XML Editor assists you in maintaining a consistent code writing style with the **Format and Indent** action that is available in the **Document** > **Source** menu and also on the toolbar..

The **Format and Indent** action achieves this by performing the following steps:

- Manages whitespaces, by collapsing or inserting space characters where needed.
- Formats complex expressions on multiple, more readable lines by properly indenting each of them. The amount of whitespaces that form an indent unit is controlled through one of the **Indent with tabs** and **Indent size** options from the ***Format Preferences*** *page*.

Note: These operations can be performed only if your XQuery document conforms with W3C XQuery 1.0, XQuery Update Facility 1.0, and XQuery 3.0 specifications. If the *Format and Indent* operation fails, the document is left unaltered and an error message is presented in the ***Results*** *view*.

Folding in XQuery Documents

In a large XQuery document, the instructions enclosed in the '{' and '}' characters can be collapsed so that only the needed instructions remain in focus. The same *folding features available for XML documents* are also available in XQuery documents.

```
8   let $minRating := min($review/reviews/review[@movie-id = $movie-id]/rating)
9   return
10   <movie id="{$movie/@id}">
11   {$movie/title}
12   {$movie/year}
13   <avgRating>
14     {
15       if ($avgRating) then $avgRating else "not rated"
16     }
17   </avgRating>
18   <maxRating>
19     <value>
20       { [2 lines]
23     </value>
24     { [5 lines]
30   </maxRating>
31   <minRating>
32     <value>
33       { [2 lines]
36     </value>
37     { [5 lines]
43   </minRating>
44 </movie>
```

Figure 366: Folding in XQuery Documents

There is available the action **Go to Matching Bracket** **Ctrl + Shift + G** on contextual menu of XQuery editor for going to matching character when cursor is located at '{' character or '}' character. It helps for finding quickly matching character of current *folding element*.

XQuery Outline View

The XQuery document structure is presented in the **Outline** view. The outline tree presents the list of all the components (namespaces, imports, variables, and functions) from both the edited XQuery file and its imports and it allows quick access to components. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

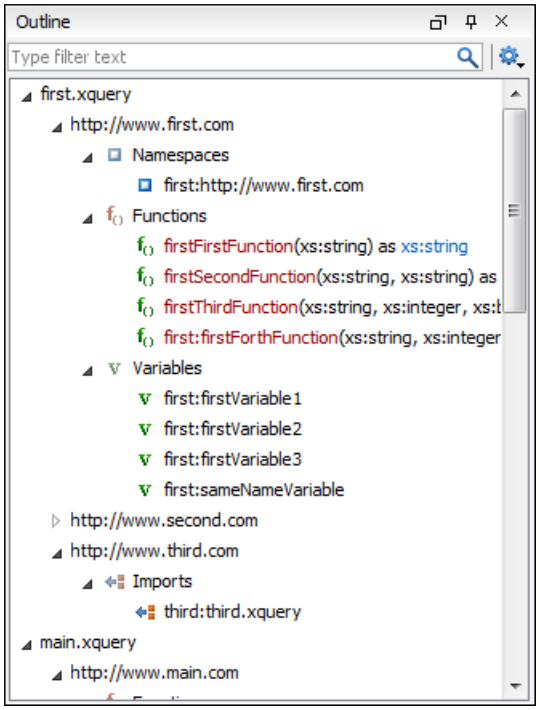


Figure 367: XQuery Outline View

The following actions are available in the **Settings** menu on the **Outline** view toolbar:

Selection update on cursor move

Controls the synchronization between **Outline** view and source document. The selection in the **Outline** view can be synchronized with the cursor moves or the changes performed in the XQuery editor. Selecting one of the components from the **Outline** view also selects the corresponding item in the source document.

Sort

Allows you to alphabetically sort the XQuery components.

Show all components

Displays all collected components starting from the current file. This option is set by default.

Show only local components

Displays the components defined in the current file only.

Group by location/namespace/type

Allows you to group the components by location, namespace, and type. When grouping by namespace, the main XQuery module namespace is presented first in the **Outline** view.

If you know the component name, you can search it in the **Outline** view by typing its name in the filter text field from the top of the view or directly on the tree structure. When you type the component name in the filter text field you can switch to the tree structure using the arrow keys of the keyboard, **(Enter)**, **(Tab)**, **(Shift-Tab)**. To switch from tree structure to the filter text field, you can use **(Tab)**, **(Shift-Tab)**.

Tip: The search filter is case insensitive. The following wildcards are accepted:

- ***** - any string
- **?** - any character
- **,** - patterns separator

If no wildcards are specified, the string to search is used as a partial match.

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (*, ?) and separate multiple patterns with commas.

XQuery Builder View

The **XPath/XQuery Builder** view allows you to compose complex XQuery expressions and execute them over the currently edited XML document. You can use the `doc( )` function to specify the source file for which the expressions are executed. When you connect to a database, the expressions are executed over that database. If you are using the **XPath/XQuery Builder** view and the current file is an XSLT document, Oxygen XML Editor executes the expressions over the XML document in the associated scenario.

If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

The upper part of the view contains the following actions:

XPath version chooser drop-down menu

A drop-down menu that allows you to select the type of the expression you want to execute. You can choose between:

- XPath 1.0 (Xerces-driven)
- XPath 2.0, XPath 2.0SA, XPath 3.0, XPath 3.0SA, XQuery 1.0, XQuery 3.0, Saxon-HE XQuery, Saxon-PE XQuery, or Saxon-EE XQuery (all of them are Saxon-driven)
- Custom connection to XML databases that can execute XQuery expressions

  Note: The results returned by XPath 2.0 SA and XPath 3.0 SA have a location limited to the line number of the start element (there are no column information and no end specified).

  Note: Oxygen XML Editor uses Saxon to execute XPath 3.0 expressions. Since Saxon implements a part of the 3.0 functions, when using a function that is not implemented, Oxygen XML Editor returns a compilation error.

Execute XPath button

Use this button to start the execution of the XPath or XQuery expression you are editing. The result of the execution is displayed in the ***Results*** *view*.

Favorites button

Allows you to save certain expressions that you can later reuse. To add an expression as a favorite, press the star button and enter a name for it. The star turns yellow to confirm that the expression was saved. Expand the drop-down menu next to the star button to see all your favorites. Oxygen XML Editor automatically groups favorites in folders named after the method of execution.

History drop-down menu

Keeps a list of the last 15 executed XPath or XQuery expressions. Use the **Clear history** action from the bottom of the list to remove them.

Settings drop-down menu

Contains the following three options:

- **Update on cursor move** - When selected and you navigate through a document, the XPath expression corresponding to the XML node at the current cursor position is displayed.
- **Evaluate as you type** - When you select this option, the XPath expression you are composing is evaluated in real time.

  Note: The **Evaluate as you type** option and the automatic validation are disabled when you edit *huge documents* or when the scope is other than **Current file**.
- **Options** - Opens the Preferences page of the currently selected processing engine.

XPath scope menu

Oxygen XML Editor allows you to define a scope for which the XPath operation will be executed. You can choose where the XPath expression will be executed:

- **Current file** - Current selected file only.
- **Project** - All the files in the project.
- **Selected project resources** - The files selected in the project.
- **All opened files** - All files opened in the application.
- **Current DITA Map hierarchy** - All resources referenced in the currently selected *DITA map*, opened in the ***DITA Maps Manager*** *view*.
- **Opened archive** - Files open in the ***Archive Browser*** *view*.
- **Working sets** - The selected *working sets*.

At the bottom of the scope menu the following scope configuration actions are available:

- **Configure XPath working sets** - Allows you to configure and manage collections of files and folders, encapsulated in logical containers called *working sets*.
- **XPath file filter** - You can filter the files from the selected scope for which the XPath expression will be executed. By default, the XPath expression will be executed only on XML files, but you can also define a set of patterns that will filter out files from the current scope. If you select the **Include archive** option, the XPath expression will be also executed on the files in any archive (including EPUB and DocX) found at the current scope.

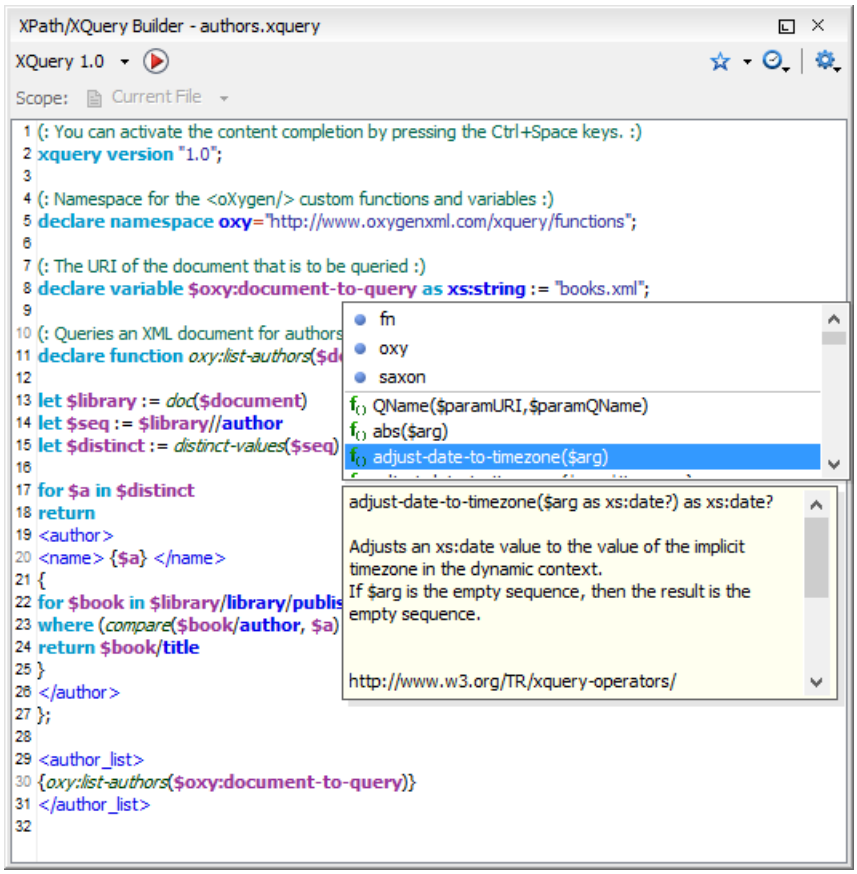


Figure 368: XPath/XQuery Builder View

While you edit an XPath or XQuery expression, Oxygen XML Editor assists you with the following features:

- *Content Completion Assistant* - It offers context-dependent proposals and takes into account the cursor position in the document you are editing. The set of functions proposed by the *Content Completion Assistant* also depends on the engine version. Select the engine version from the drop-down menu available in the toolbar.
- Syntax Highlighting - Allows you to identify the components of an expression. To customize the colors of the components of the expression, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to ***Editor** > **Syntax Highlight***.
- Automatic validation of the expression as you type.

  Note: When you type invalid syntax a red serrated line underlines the invalid fragments.
- Function signature and documentation balloon, when the cursor is located inside a function.

The usual edit actions (**Cut**, **Copy**, **Paste**, **Select All**, **Undo**, **Redo**) are available in the contextual menu of the top editable part of the view.

XSLT/XQuery Input View

The structure of the XML document associated to the edited XSLT stylesheet, or the structure of the source documents of the edited XQuery is displayed in a tree form in a view called the **XSLT/XQuery Input** view. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu. The tree nodes represent the elements of the documents.

XSLT

If you click a node in the **XSLT/XQuery Input** view, the corresponding template from the stylesheet is highlighted. A node can be dragged from this view and dropped in the editor area for quickly inserting `xsl:template`, `xsl:for-each`, or other XSLT elements that have the `match/select/test` attribute already completed. The value of the attribute is the correct XPath expression that refers to the dragged tree node. This value is based on the current editing context of the drop spot.

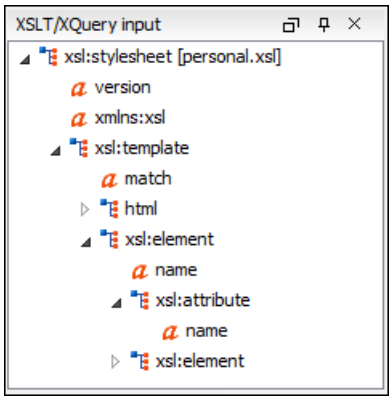


Figure 369: XSLT Input View

XSLT Example:

For the following XML document:

```
<personnel>
    <person id="Big.Boss">
        <name>
            <family>Boss</family>
            <given>Big</given>
        </name>
        <email>chief@oxygenxml.com</email>
        <link subordinates="one.worker"/>
    </person>
    <person id="one.worker">
        <name>
            <family>Worker</family>
            <given>One</given>
        </name>
        <email>one@oxygenxml.com</email>
        <link manager="Big.Boss"/>
    </person>
</personnel>
```

and the following XSLT stylesheet:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        version="2.0">
    <xsl:template match="personnel">
        <xsl:for-each select="*">

        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

if you drag the `given` element and drop it inside the `xsl:for-each` element, the following pop-up menu is displayed:

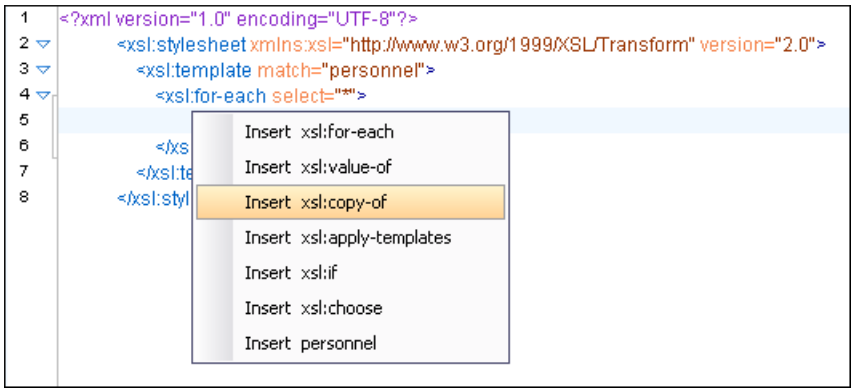


Figure 370: XSLT Input Drag and Drop Pop-up Menu

If you select **Insert xsl:copy-of** (for example), the resulting document will look like this:

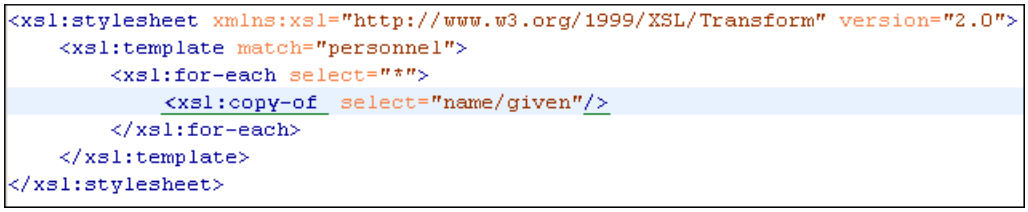


Figure 371: XSLT Input Drag and Drop Result

XQuery

You can also use the **XSLT/XQuery Input** view to drag and drop a node into the editing area to quickly insert XQuery expressions.

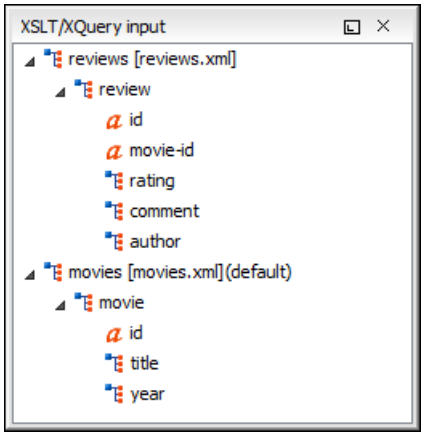


Figure 372: XQuery Input View

XQuery Example:

For the following XML documents:

```
<movies>
<movie id="1">
<title>The Green Mile</title>
<year>1999</year>
</movie>
<movie id="2">
<title>Taxi Driver</title>
<year>1976</year>
</movie>
</movies>
```

```
<reviews>
<review id="100" movie-id="1">
<rating>5</rating>
<comment>It is made after a great Stephen King book.
</comment>
<author>Paul</author>
</review>
<review id="101" movie-id="1">
<rating>3</rating>
<comment>Tom Hanks does a really nice acting.</comment>
<author>Beatrice</author>
</review>
<review id="104" movie-id="2">
<rating>4</rating>
<comment>Robert De Niro is my favorite actor.</comment>
<author>Maria</author>
</review>
</reviews>
```

and the following XQuery:

```
let $review := doc("reviews.xml")
for $movie in doc("movies.xml")/movies/movie
let $movie-id := $movie/@id
return
<movie id="{$movie/@id}">
{$movie/title}
{$movie/year}
<maxRating>
{

}
</maxRating>
</movie>
```

If you drag the **review** element and drop it between the braces, the following pop-up menu is displayed:

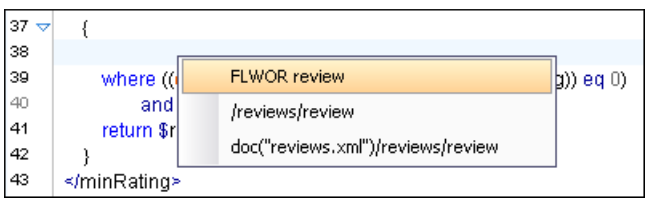


Figure 373: XQuery Input Drag and Drop Pop-up Menu

Select **FLWOR review**, the resulting document will look like this:

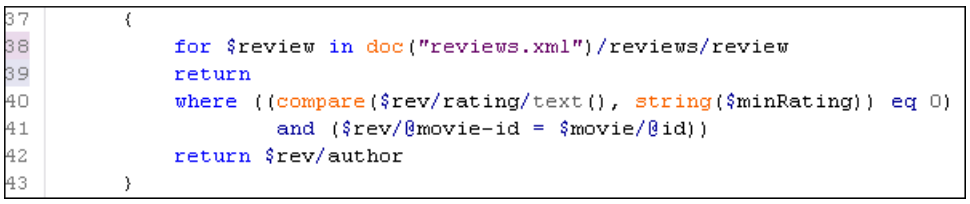


Figure 374: XQuery Input Drag and Drop Result

Generating HTML Documentation for an XQuery Document

To generate HTML documentation for an XQuery document, use the **XQuery Documentation** dialog box. It is opened with the **XQuery Documentation** action that is available from the **Tools** > **Generate Documentation** menu or from the **Generate Documentation** submenu in the contextual menu of the ***Project*** *view*.

The dialog box allows you to configure a set of parameters for the process of generating the HTML documentation.

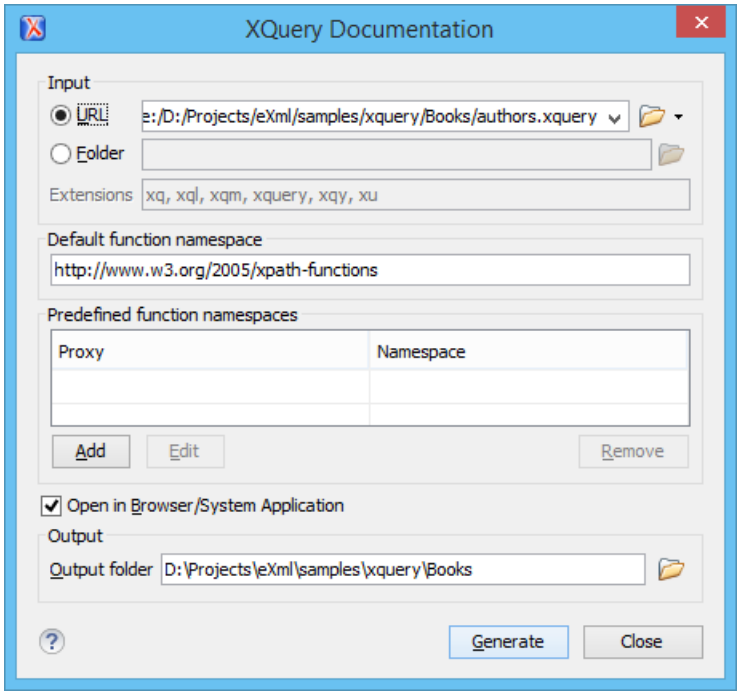


Figure 375: XQuery Documentation Dialog Box

The following options are available:

- **Input** - The full path to the XQuery file must be specified in one of the two fields in this section:
  - **URLFile** - The URL of the file in which you want to generate the documentation.
  - **Folder** - The directory that contains the files for which you want to generate the documentation. You can also specify the XQuery file extensions to be searched for in the specified directory.
- **Default function namespace** - Optional URI for the default namespace for the submitted XQuery.
- **Predefined function namespaces** - Optional, engine-dependent, predefined namespaces that the submitted XQuery refers to. They allow the conversion to generate annotation information to support the presentation component hypertext linking (only if the predefined modules have been loaded into the local xqDoc XML repository).
- **Open in Browser/System Application** - Select this option if you want the result to be opened in the system application associated with that file type.

  Note: To set the browser or system application that will be used, *open the* ***Preferences*** *dialog box* ***(Options > Preferences)***, go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.
- **Output** - Allows you to specify where the generated documentation is saved on disk.

Transforming XML Documents Using XQuery

XQuery is similar to XSL stylesheets, both being capable of transforming an XML input into another format. You specify the input URL when you *define the transformation scenario*. The result can be saved and opened in the associated application. You can even run a *FO processor* on the output of an XQuery. The transformation scenarios may be shared between many XQuery files, are *exported* together with the XSLT scenarios and can be managed in *the* ***Configure Transformation Scenario*** *dialog box* ,or in *the* ***Scenarios*** *view*. The transformation

can be performed on the XML document specified in the **XML URL** field, or, if this field is empty, the documents referenced from the query expression. The parameters of XQuery transforms must be set in *the **Parameters** dialog box*. Parameters that are in a namespace must be specified using the qualified name (for example, a `param` parameter in the *http://www.oxygenxml.com/ns* namespace must be set with the name `{http://www.oxygenxml.com/ns}param`).

The transformation uses one of the Saxon 9.7.0.15 HE, Saxon 9.7.0.15 PE, Saxon 9.7.0.15 EE processors, a database connection (details can be found in the *Working with Databases* chapter - in the *XQuery transformation* section) or any XQuery processor that provides an XQJ API implementation.

The Saxon 9.7.0.15 EE processor also supports XQuery 3.0 transformations.

Related Information:

XQuery and Databases on page 1115

Display XQuery Result in Sequence View

The result of an XQuery executed on a database can be very large and sometimes only a part of the full result is needed. To avoid the long time necessary for fetching the full result, select the ***Present as a sequence** option* in the **Output** tab of the **Edit scenario** dialog box. This option fetches only the first chunk of the result. Clicking the **More results available** label that is displayed at the bottom of the **Sequence** view fetches the next chunk of results.

The size of a chunk can be set with the ***Size limit of Sequence view** option*. The **XQuery options** button from the **More results available** label provides a quick access to this option by opening the ***XQuery** preferences page* where the option can be modified.

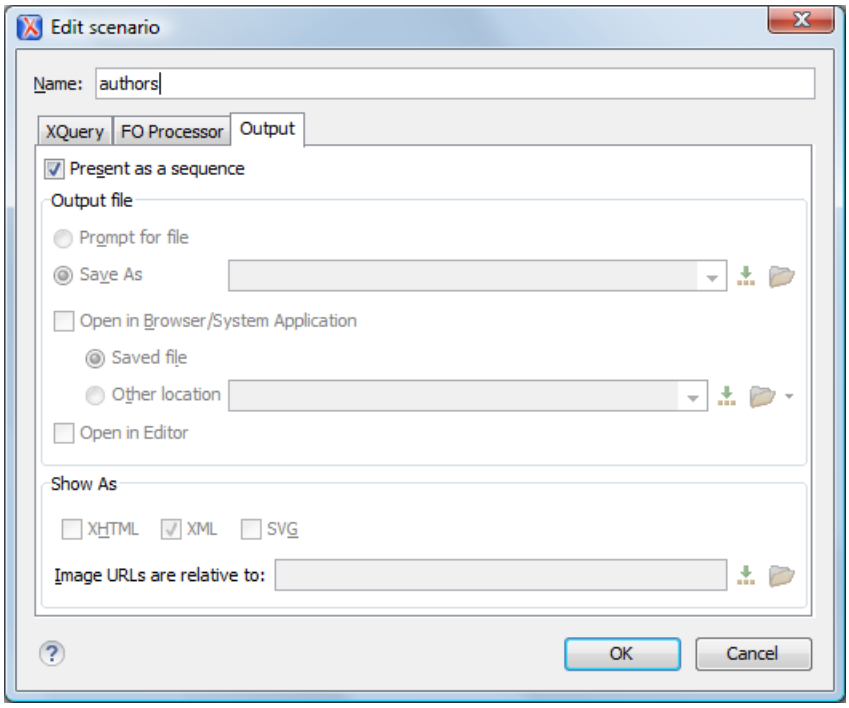


Figure 376: XQuery transformation result displayed in Sequence view

A chunk of the XQuery transformation result is displayed in the **Sequence** view.

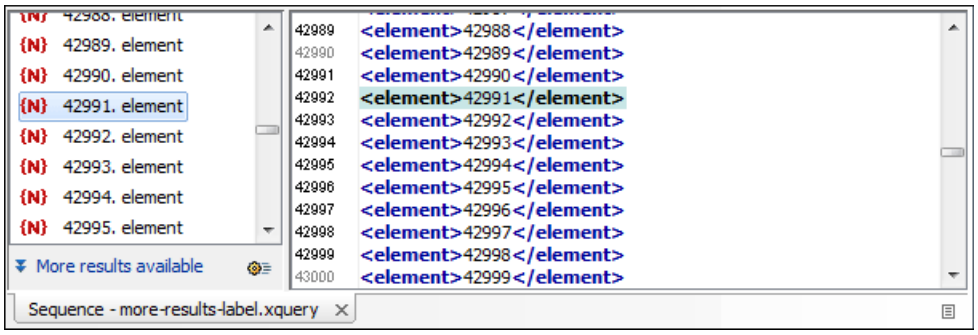


Figure 377: XQuery transformation result displayed in Sequence view

Advanced Saxon HE/PE/EE XQuery Transformation Options

The XQuery transformation scenario allows you to configure advanced options that are specific for the Saxon HE (Home Edition), PE (Professional Edition), and EE (Enterprise Edition) engines. They are the same options as *those in the **Saxon HE/PE/EE** preferences page* but they are configured as a specific set of transformation options for each transformation scenario, while the values set in the preferences page apply as global options. The advanced options configured in a transformation scenario override the *global options* defined in the preferences page.

The advanced options for Saxon 9.7.0.15 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE) are as follows:

Recoverable errors ("-warnings")

Allows you to choose how dynamic errors are handled. The following options can be selected:

- **Recover silently ("silent")** - Continues processing without reporting the error.
- **Recover with warnings ("recover")** - Issues a warning but continues processing.
- **Signal the error and do not attempt recovery ("fatal")** - Issues an error and stops processing.

Strip whitespaces ("-strip")

Allows you to choose how the *strip whitespaces* operation is handled. You can choose one of the following values:

- **All ("all")** - Strips *all* whitespace text nodes from source documents before any further processing, regardless of any `xml:space` attributes in the source document.
- **Ignore ("ignorable")** - Strips all *ignorable* whitespace text nodes from source documents before any further processing, regardless of any `xml:space` attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.
- **None ("none")** - Strips *no* whitespace before further processing.

Optimization level ("-opt")

Allows you to set the optimization level. It is the value is an integer in the range of 0 (no optimization) to 10 (full optimization). This option allows optimization to be suppressed when reducing the compiling time is important, optimization conflicts with debugging, or optimization causes extension functions with side-effects to behave unpredictably.

Use linked tree model ("-tree:linked")

This option activates the linked tree model.

Enable XQuery 3.0 support ("-qversion:(1.0|3.0)")

If selected (default value), Saxon runs the XQuery transformation with the XQuery 3.0 support.

Initializer class

Equivalent to the `-init` Saxon command-line argument. The value is the name of a user-supplied class that implements the `net.sf.saxon.lib.Initializer` interface. This initializer is called during the initialization process, and may be used to set any options required on the configuration programmatically. It is particularly useful for tasks such as registering extension functions, collations, or external object models, especially in Saxon-HE where the option cannot be set via a configuration file. Saxon only calls the initializer

when running from the command line, but the same code may be invoked to perform initialization when running user application code.

The following advanced options are specific for Saxon 9.7.0.15 Professional Edition (PE) and Enterprise Edition (EE) only:

Use a configuration file ("-config")

Sets a Saxon 9.7.0.15 configuration file that is used for XQuery transformation and validation scenarios.

Allow calls on extension functions ("-ext")

If selected, calls on external functions are allowed. Selecting this option is recommended in an environment where untrusted stylesheets may be executed. It also disables user-defined extension elements and the writing of multiple output files, both of which carry similar security risks.

The advanced options that are specific for Saxon 9.7.0.15 Enterprise Edition (EE) are as follows:

Validation of the source file ("-val")

Requests schema-based validation of the source file and of any files read using `document()` or similar functions. It can have the following values:

- **Schema validation ("strict")** - This mode requires an XML Schema and allows for parsing the source documents with strict schema-validation enabled.
- **Lax schema validation ("lax")** - If an XML Schema is provided, this mode allows for parsing the source documents with schema-validation enabled but the validation will not fail if, for example, element declarations are not found.
- **Disable schema validation** - This specifies that the source documents should be parsed with schema-validation disabled.

Validation errors in the result tree treated as warnings ("-outval")

Normally, if validation of result documents is requested, a validation error is fatal. Selecting this option causes such validation failures to be treated as warnings.

Write comments for non-fatal validation errors of the result document

The validation messages for non-fatal errors are written (wherever possible) as a comment in the result document itself.

Generate bytecode ("--generateByteCode:(on|off)")

If you select this option, Saxon-EE attempts to generate Java bytecode for evaluation of parts of a query or stylesheet that are amenable to such an action. For further details regarding this option, go to *http://www.saxonica.com/documentation9.5/index.html#!javadoc*.

Enable XQuery update ("-update:(on|off)")

This option controls whether or not XQuery update syntax is accepted. The default value is off.

Backup files updated by XQuery ("-backup:(on|off)")

If selected, backup versions for any XML files updated with an XQuery Update are generated. This option is available when the **Enable XQuery update** option is selected.

Updating XML Documents using XQuery

Using the bundled Saxon 9.7.0.15 EE XQuery processor Oxygen XML Editor offers support for XQuery Update 1.0. The XQuery Update Facility provides expressions that can be used to make persistent changes to instances of the XQuery 1.0 and XPath 2.0 Data Model. Thus, besides querying XML documents, you can modify them using the various insert/delete/modify/create methods available in the *XQuery Update 1.0* standard.

Choose Saxon 9.7.0.15 EE as a transformer in the scenario associated with the XQuery files containing update statements and Oxygen XML Editor will notify you if the update was successful.

Example: Using XQuery Update to modify a tag name in an XML file

```
rename node doc("books.xml")//publisher[1]//book[1] as "firstBook"
```

Editing WSDL Documents

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services).

Oxygen XML Editor provides a special type of editor dedicated to WSDL documents. The WSDL editor offers support for validation, a specialized *Content Completion Assistant*, a component oriented ***Outline*** *view*, searching and refactoring operations, and support to generate documentation.

Both WSDL version 1.1 and 2.0 are supported and SOAP versions 1.1 and 1.2. That means that in the location where a SOAP extension can be inserted the *Content Completion Assistant* offers elements from both SOAP 1.1 and SOAP 1.2. Validation of SOAP requests is executed first against a SOAP 1.1 schema and then against a SOAP 1.2 schema. In addition to validation against the XSD schemas, Oxygen XML Editor also checks if the WSDL file conforms with the WSDL specification (available only for WSDL 1.1 and SOAP 1.1).

In the following example you can see how the errors are reported.

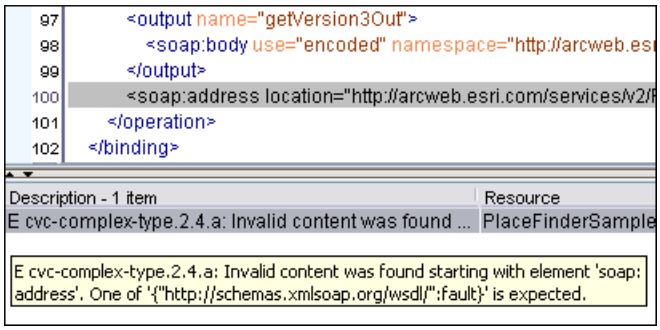


Figure 378: Validating a WSDL file

To watch our video demonstration about the WSDL editing support in Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/Create_New_WSDL.html*.

Related Information:

Editing XML Documents in Text Mode on page 290

Editing WSDL Documents in the Master Files Context

Smaller interrelated modules that define a complex WSDL structure cannot be correctly edited or validated individually, due to their interdependency with other modules. Oxygen XML Editor provides the support for defining the main module (or modules), allowing you to edit any of the imported/included files in the context of the larger WSDL structure.

You cat set a main WSDL document either using the *master files support from the* ***Project*** *view*, or using a validation scenario.

To set a *master file* using a validation scenario, add validation units that point to the main modules. Oxygen XML Editor warns you if the current module is not part of the dependencies graph computed for the main WSDL document. In this case, it considers the current module as the main WSDL document.

The advantages of editing in the context of a *master file* include:

- Correct validation of a module in the context of a larger WSDL structure.
- *Content Completion Assistant* displays all components valid in the current context.
- The ***Outline*** *view* displays the components collected from the entire WSDL structure.

Note: When you edit an XML schema document that has a WSDL document set as master, the validation operation is performed over the master WSDL document.

To watch our video demonstration about editing WSDL documents in the *master files* context, go to *https://www.oxygenxml.com/demo/WSDL_Working_Modules.html*.

Validating WSDL Documents

By default, WSDL files are validated as you type. To change this, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Editor** > **Document Checking**, and deselect the ***Enable automatic validation** option*.

To validate a WSDL document manually, select the **Validate** action from the **Validation** toolbar drop-down menu or the **Document** > **Validate** menu. The validation problems are highlighted directly in the editor, making it easy to locate and fix any issues.

Content Completion Assistance in WSDL Documents

The *Content Completion Assistant* is a powerful feature that enhances the editing of WSDL documents. It helps you define WSDL components by proposing context-sensitive element names. It can be manually activated with the **Ctrl + Space (Command + Space on OS X)** shortcut.

Another important capability of the *Content Completion Assistant* is to propose references to the defined components when you edit attribute values. For example, when you edit the `type` attribute of a `binding` element, the *Content Completion Assistant* proposes all the defined port types. Each proposal that the *Content Completion Assistant* offers is accompanied by a documentation hint.

Note: XML schema specific elements and attributes are offered when the current editing context is the internal XML schema of a WSDL document.

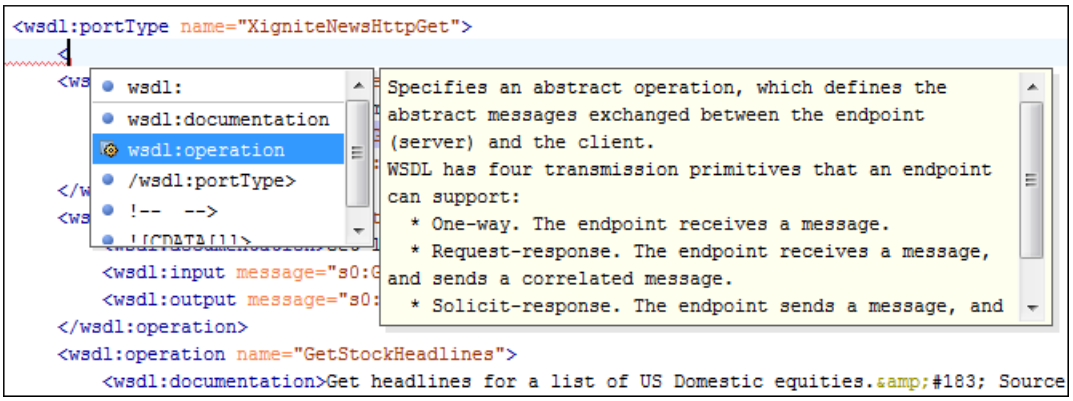


Figure 379: WSDL Content Completion Assistant

Note: If you are using the concept of *master files* to import/include modules, the *Content Completion Assistant* collects its components starting from the *master files*. The *master files* can be defined in the project or in the associated validation scenario. For more information about the *Master Files* support in Oxygen XML Editor, see *Defining Master Files at Project Level*.

Namespace prefixes in the scope of the current context are presented at the top of the content completion assistance window to speed up the insertion into the document of prefixed elements.

Figure 380: Namespace Prefixes in the Content Completion Assistant

For the common namespaces, such as XML Schema namespace (*http://www.w3.org/2001/XMLSchema*) or SOAP namespace (*http://schemas.xmlsoap.org/wsdl/soap/*), Oxygen XML Editor provides an easy mode to declare them by proposing a prefix for these namespaces.

WSDL Syntax Highlighting

Oxygen XML Editor supports syntax highlighting in WSDL documents to improve the readability of the content and reduce the time it takes to internalize the semantics of the structured content.

To customize the colors or styles used for the syntax highlighting colors for WSDL files, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)***.
2. Go to ***Editor*** > ***Syntax Highlight***.
3. Select and expand the **XML** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the **XML** tab in the **Preview** pane to see the effects of your changes.

Tip: Oxygen XML Editor also allows you to specify syntax highlighting colors for specific XML elements and attributes with specific namespace prefixes. This can be done in the ***Editor*** > ***Syntax Highlight*** > ***Elements/ Attributes by Prefix*** *preferences page*.

Related Information:

Customize Syntax Highlight colors on page 104

WSDL Outline View

The **Outline** view for WSDL documents displays the list of all the components (services, bindings, port types and so on) of the currently open WSDL document along with the components of its imports.

If you use the ***Master Files*** *support*, the **Outline** view collects the components of a WSDL document starting from the *master files* of the current document.

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

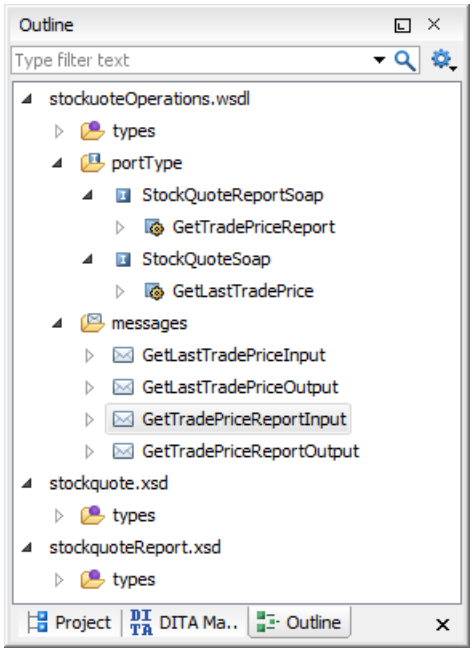


Figure 381: WSDL Outline View

The **Outline** view can display both the components of the current document and its XML structure, organized in a tree-like fashion. You can switch between the display modes by using the **Show XML structure** and **Show components** actions in the **Settings** menu on the **Outline** view toolbar. The following actions are available:

Filter returns exact matches

The text filter of the **Outline** view returns only exact matches.

Selection update on cursor move

Controls the synchronization between the **Outline** view and the current document. The selection in the **Outline** view can be synchronized with the cursor moves or the changes in the WSDL editor. Selecting one of the components from the **Outline** view also selects the corresponding item in the current document.

When the **Show components** option is selected, the following actions are available:

Show XML structure

Displays the XML structure of the current document in a tree-like manner.

Sort

Sorts the components in the **Outline** view alphabetically.

Show all components

Displays all the components that were collected starting from current document or from the main document, if it is defined.

Show referable components

Displays all the components that you can reference from the current document.

Show only local components

Displays the components defined in the current file only.

Group by location

Groups the WSDL components by their location.

Group by type

Groups the WSDL components by their type.

Group by namespace

Groups the WSDL components by their namespace.

Note: By default, all the three grouping criteria are active.

When the **Show XML structure** option is selected, the following actions are available:

Show components

Switches the **Outline** view to the components display mode.

Flat presentation mode of the filtered results

When active, the application flattens the filtered result elements to a single level.

Show comments and processing instructions

Show/hide comments and processing instructions in the **Outline** view.

Show element name

Show/hide element name.

Show text

Show/hide additional text content for the displayed elements.

Show attributes

Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from *the Outline preferences panel*.

Configure displayed attributes

Displays the ***XML Structured Outline*** *preferences page*.

The following contextual menu actions are available in the **Outline** view when the **Show components** option is selected in the **Settings** menu:

Edit Attributes

Opens a dialog box that allows you to edit the attributes of the currently selected component.

Cut

Cuts the currently selected component.

Copy

Copies the currently selected component.

Delete

Deletes the currently selected component.

Search references

Searches for the references of the currently selected component.

Search references in

Searches for the references of the currently selected component in the context of a scope that you define.

Component dependencies

Opens the ***Component Dependencies*** *view* that displays the dependencies of the currently selected component.

Resource Hierarchy

Opens the ***Resource Hierarchy/Dependencies*** *view* that displays the hierarchy for the currently selected resource.

Resource Dependencies

Opens the ***Resource Hierarchy/Dependencies*** *view* that displays the dependencies of the currently selected resource.

Rename Component in

Renames the currently selected component in the context of a scope that you define.

The following contextual menu actions are available in the **Outline** view when the **Show XML structure** option is selected in the **Settings** menu:

Append Child

Displays a list of elements that you can insert as children of the current element.

Insert Before

Displays a list of elements that you can insert as siblings of the current element, before the current element.

Insert After

Displays a list of elements that you can insert as siblings of the current element, after the current element.

Edit Attributes

Opens a dialog box that allows you to edit the attributes of the currently selected component.

Toggle Comment

Comments/uncomments the currently selected element.

Search references

Searches for the references of the currently selected component.

Search references in

Searches for the references of the currently selected component in the context of a scope that you define.

Component dependencies

Opens the ***Component Dependencies*** *view* that displays the dependencies of the currently selected component.

Rename Component in

Renames the currently selected component in the context of a scope that you define.

Cut

Cuts the currently selected component.

Copy

Copies the currently selected component.

Delete

Deletes the currently selected component.

Expand More

Expands the structure of a component in the **Outline** view.

Collapse All

Collapses the structure of all the component in the **Outline** view.

To switch from the tree structure to the text filter, use **Tab** and **Shift-Tab**.

Tip: The search filter is case insensitive. The following wildcards are accepted:

- ***** - any string
- **?** - any character
- **,** - patterns separator

If no wildcards are specified, the string to search is used as a partial match.

The content of the **Outline** view and the editing area are synchronized. When you select a component in the **Outline** view, its definition is highlighted in the editing area.

WSDL Resource Hierarchy/Dependencies View in WSDL Documents

The **Resource Hierarchy/Dependencies** view allows you to see the hierarchy/dependencies for a WSDL resource. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Note: The hierarchy of a WSDL resource includes the hierarchy of imported XML Schema resources. The dependencies of an XML Schema resource present the WSDL documents that import the schema.

To view the hierarchy of a WSDL document, select the document in the ***Project*** *view* and choose **Resource Hierarchy** from the contextual menu.

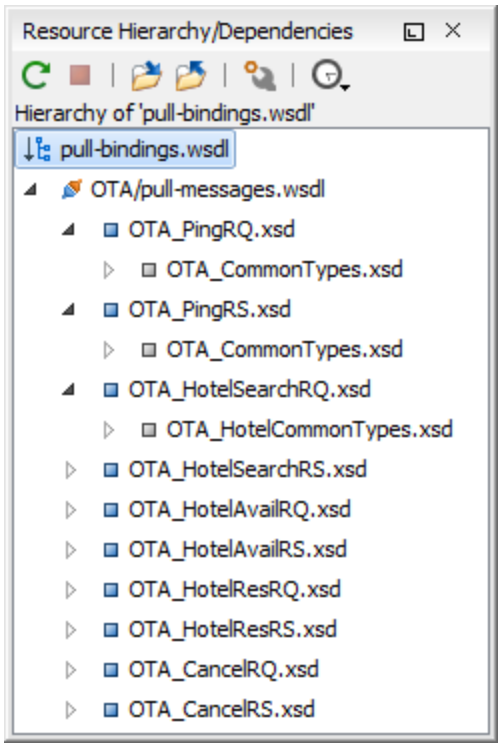


Figure 382: Resource Hierarchy/Dependencies View

If you want to see the dependencies of a WSDL document, select the document in the ***Project*** *view* and choose **Resource Dependencies** from the contextual menu.

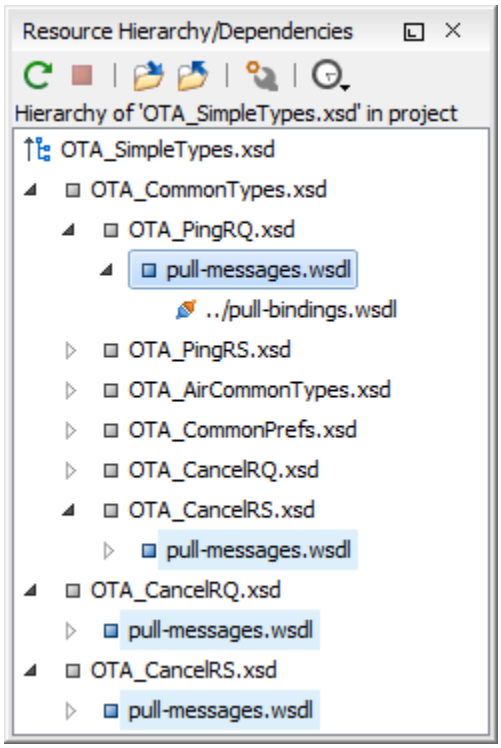


Figure 383: Resource Hierarchy/Dependencies View

The following actions are available in the **Resource Hierarchy/Dependencies** view:

Refresh

Refreshes the Hierarchy/Dependencies structure.

Stop

Stops the hierarchy/dependencies computing.

Show Hierarchy

Allows you to choose a resource to compute the hierarchy structure.

Show Dependencies

Allows you to choose a resource to compute the dependencies structure.

Configure

Allows you to configure a scope to compute the dependencies structure. There is also an option for automatically using the defined scope for future operations.

History

Provides access to the list of previously computed dependencies. Use the **Clear history** button to remove all items from this list.

The contextual menu contains the following actions:

Open

Opens the resource. You can also double-click a resource in the Hierarchy/Dependencies structure to open it.

Copy location

Copies the location of the resource.

Move resource

Moves the selected resource.

Rename resource

Renames the selected resource.

Show Resource Hierarchy

Shows the hierarchy for the selected resource.

Show Resource Dependencies

Shows the dependencies for the selected resource.

Add to Master Files

Adds the currently selected resource in *the **Master Files** directory*.

Expand All

Expands all the children of the selected resource from the Hierarchy/Dependencies structure.

Collapse All

Collapses all children of the selected resource from the Hierarchy/Dependencies structure.

Tip: When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon .

Note: The **Move resource** or **Rename resource** actions give you the option to *update the references to the resource*.

Related Information:

Working with Modular XML Files in the Master Files Context on page 481
Search and Refactor Operations Scope on page 480

Moving/Renaming WSDL Resources

You can move and rename a resource presented in the **Resource/Hierarchy Dependencies** view, using the **Rename resource** and **Move resource** refactoring actions from the contextual menu.

When you select the **Rename** action in the contextual menu of the **Resource/Hierarchy Dependencies** view, the **Rename resource** dialog box is displayed. The following fields are available:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references** - Select this option to update the references to the resource you are renaming.

When you select the **Move** action from the contextual menu of the **Resource/Hierarchy Dependencies** view, the **Move resource** dialog box is displayed. The following fields are available:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, in accordance with the new location and name.

If the **Update references of the moved resource(s)** option is selected, a **Preview** option (which opens the **Preview** dialog box) is available for both actions. The **Preview** dialog box presents a list with the resources that are updated.

Component Dependencies View in WSDL Documents

The **Component Dependencies** view allows you to view the dependencies for a selected WSDL component. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

To view the dependencies of an WSDL component, select the desired component in the editor and choose the **Component Dependencies** action from the contextual menu. This action is available for all WSDL components (`messages`, `port types`, `operations`, `bindings` and so on).

Note: If you search for dependencies of XML Schema elements, the **Component Dependencies** view presents the references from WSDL documents.

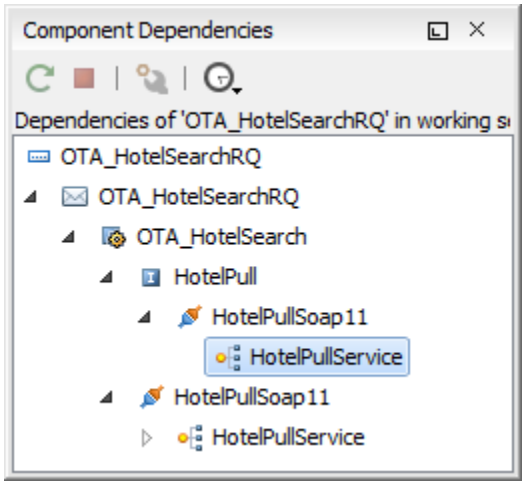


Figure 384: Component Dependencies View

The following action are available in the toolbar of the **Component Dependencies** view:

Refresh

Refreshes the dependencies structure.

Stop

Stops the dependencies computing.

Configure

Allows you to configure a *search scope* to compute the dependencies structure. You can decide to use the defined scope for future operations automatically, by selecting the corresponding checkbox.

History

Allows you to repeat a previous dependencies computation.

The following actions are available in the contextual menu of the **Component Dependencies** view:

Go to First Reference

Selects the first reference of the referenced component from the current selected component in the dependencies tree.

Go to Component

Displays the definition of the current selected component in the dependencies tree.

Tip: If a component contains multiple references to another, a small table is displayed that contains all references. When a recursive reference is encountered, it is marked with a special icon .

Highlight Component Occurrences in WSDL Documents

When you position your mouse cursor over a component in a WSDL document, Oxygen XML Editor searches for the component declaration and all its references and highlights them automatically.

Customizable colors are used: one for the component definition and another one for component references. Occurrences are displayed until another component is selected.

To change the default behavior of **Highlight Component Occurrences**, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Mark Occurrences**. You can also trigger a search using the **Search** > **Search Occurrences in File ()** action from contextual menu. Matches are displayed in separate tabs of the ***Results** view*.

Searching and Refactoring Operations in WSDL Documents

Search Actions

The following search actions are available from the **Search** submenu in the contextual menu of the current editor or from the **Document** > **References** menu:

- **Search References** - Searches all references of the item found at current cursor position in the defined scope, if any. If a scope is defined, but the current edited resource is not part of the range of resources determined by this, a warning dialog box is displayed and you have the possibility to define another search scope.
- **Search References in** - Searches all references of the item found at current cursor position in the file or files that you specify when define a scope in the **Search References** dialog box.
- **Search Declarations** - Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined, but the current edited resource is not part of the range of resources determined by this, a warning dialog box will be displayed and you have the possibility to define another search scope.
- **Search Declarations in** - Searches all declarations of the item found at current cursor position in the file or files that you specify when you define a scope for the search operation.
- **Search Occurrences in File** - Searches all occurrences of the item at the cursor position in the currently edited file.

The following action is available from the **Document** > **Schema** menu:

- **Show Definition** - Takes you to the location of the definition of the current item.

  Note: You can also use the **Ctrl + Single-Click (Command + Single-Click on OS X)** shortcut on a reference to display its definition.

Refactoring Actions

The following refactoring actions are available from the **Refactoring** submenu from the **Document** > **Refactoring** menu or in the contextual menu of the current editor:

- **Rename Component** - Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the **Esc** or **Enter** key on your keyboard.

- **Rename Component in** - Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the **Preview** button, you can view the files to be affected by the action.

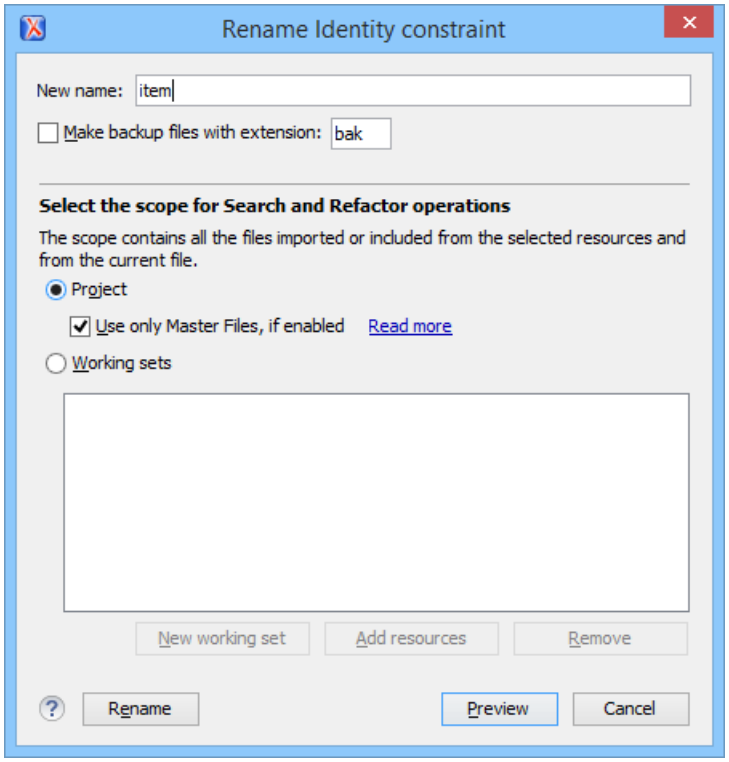


Figure 385: Rename Identity Constraint Dialog Box

Searching and Refactoring Operations Scope in WSDL Documents

The *scope* is a collection of documents that define the context of a search and refactor operation. To control it you can use the **Change scope** operation, available in the *Quick Assist* action set or on the **Resource Hierarchy/ Dependency View** toolbar. You can restrict the scope to the current project or to one or multiple *working sets*. The **Use only Master Files, if enabled** checkbox allows you to restrict the scope of the search and refactor operations to the resources from the **Master Files** directory. Click **read more** for details about the *Master Files support*.

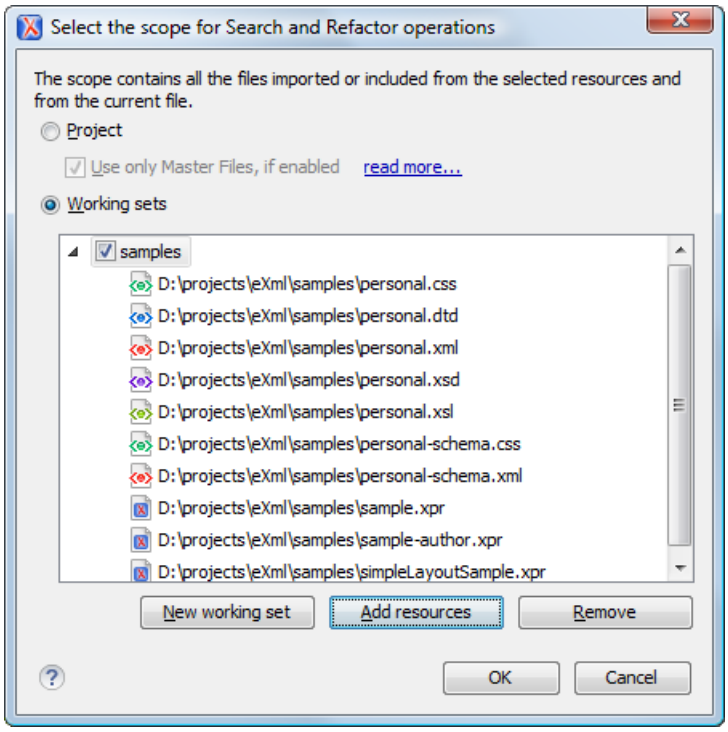


Figure 386: Change Scope Dialog Box

The scope you define is applied to all future search and refactor operations until you modify it. Contextual menu actions allow you to add or delete files, folders, and other resources to the *working set* structure.

Quick Assist Support in WSDL Documents

The *Quick Assist feature* is activated automatically when the cursor is positioned over the name of a component. It is accessible via a yellow bulb icon ( ) placed at the current line in the stripe on the left side of the editor. Also, you can invoke the *quick assist* menu by using the **Alt + 1** (**Meta + Alt + 1** on Mac OS X) keyboard shortcuts.

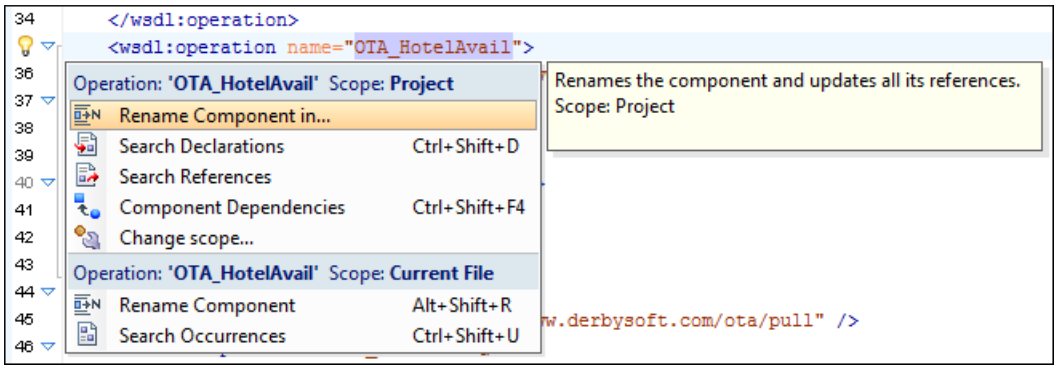


Figure 387: WSDL Quick Assist Support

The *Quick Assist* support offers direct access to the following actions:

Rename Component in

Renames the component and all its dependencies.

Search Declarations

Searches the declaration of the component in a predefined scope. It is available only when the context represents a component name reference.

Search References

Searches all references of the component in a predefined scope.

Component Dependencies

Searches the component dependencies in a predefined scope.

Change Scope

Configures the scope that will be used for future search or refactor operations.

Rename Component

Allows you to rename the current component in-place.

Search Occurrences

Searches all occurrences of the component within the current file.

Generating Documentation for WSDL Documents

You can use Oxygen XML Editor to generate detailed documentation for the components of a WSDL document in HTML format. You can select the WSDL components to include in your output and the level of details to present for each of them. Also, the components are hyperlinked. You can also generate the documentation in a *custom output format* by using a custom stylesheet.

Note: The WSDL documentation includes the XML Schema components that belong to the internal or imported XML schemas.

To generate documentation for a WSDL document, select **WSDL Documentation** from the **Tools** > **Generate Documentation** menu or from the **Generate Documentation** submenu in the contextual menu of the ***Project*** *view*.

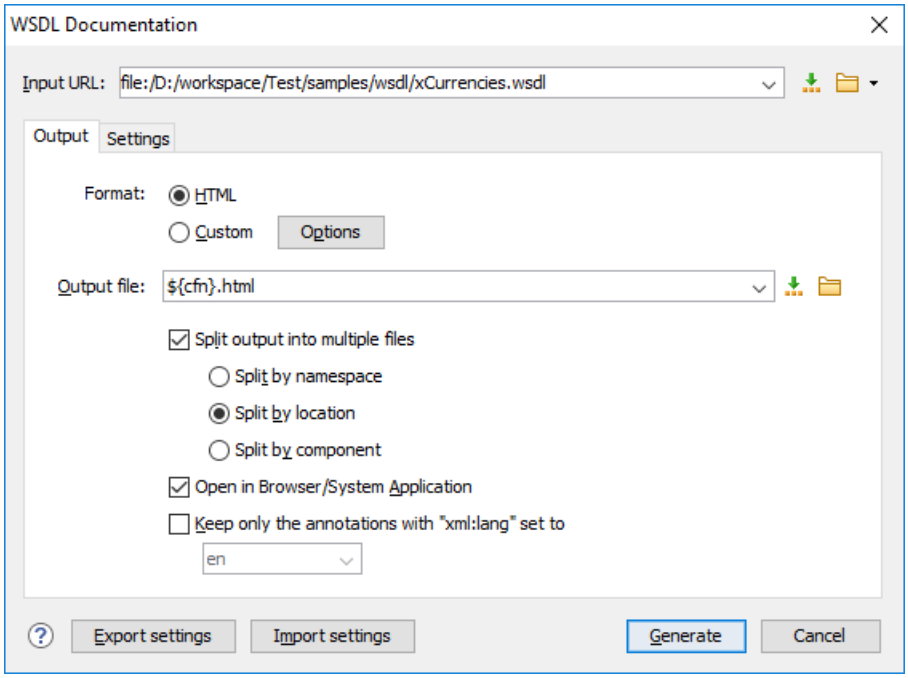


Figure 388: WSDL Documentation Dialog Box

The **Input URL** field of the dialog box must contain the full path to the WSDL document that you want to generate documentation for. The WSDL document may be a local or a remote file. You can specify the path to the WSDL file by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.

Output Tab

The following options are available in the **Output** tab:

- **Format** - Allows you to choose between the following formats:
  - **HTML** - The documentation is generated in *HTML output format*.

- **Custom** - The documentation is generated in a *custom output format*, allowing you to control the output. Click the **Options** button to open a **Custom format options** dialog box where you can specify a custom stylesheet for creating the output. There is also an option to **Copy additional resources to the output folder** and you can select the path to the additional **Resources** that you want to copy. You can also choose to keep the intermediate XML files created during the documentation process by deselecting the **Delete intermediate XML file** option.

- **Output file** - You can specify the path of the output file by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.
- **Split output into multiple files** - Instructs the application to split the output into multiple files. For large WSDL documents, choosing a different split criterion may generate smaller output files providing a faster documentation browsing. You can choose to split them by namespace, location, or component name.
- **Open in Browser/System Application** - Opens the result in the system application associated with the output file type.

  Note: To set the browser or system application that will be used, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.
- **Keep only the annotations with xml:lang set to** - The generated output will contain only the annotations with the `xml:lang` attribute set to the selected language. If you choose a primary language code (for example, **en** for English), this includes all its possible variations (**en-us**, **en-uk**, etc.).

Setting Tab

When you generate documentation for a WSDL document, you can choose what components to include in the output and the details to be included in the documentation.

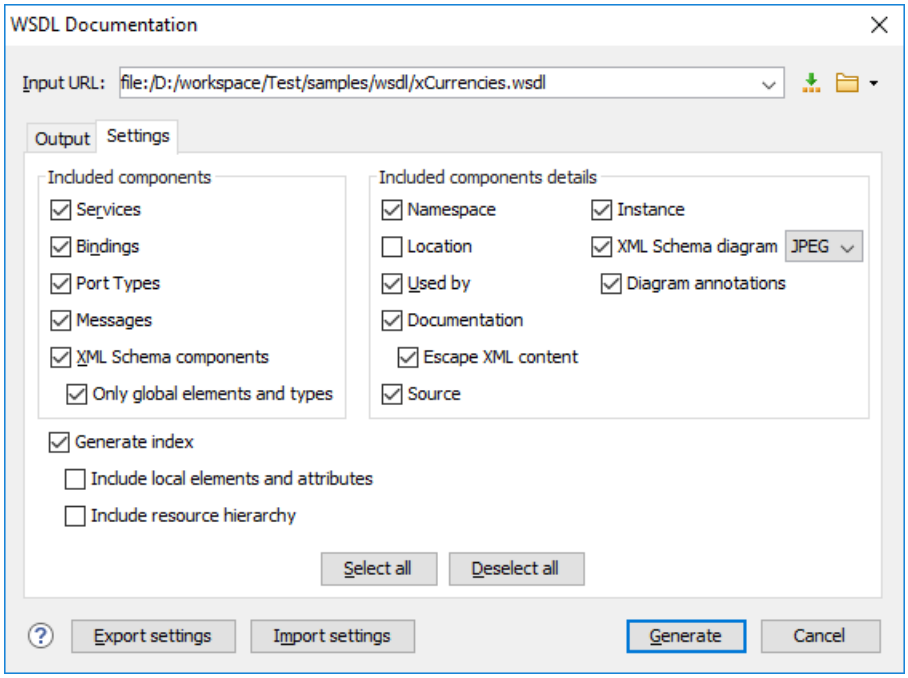


Figure 389: Settings Tab of the WSDL Documentation Dialog Box

The **Settings** tab allows you to choose whether or not to include the following:

- **Components**
  - **Services** - Specifies whether or not the generated documentation includes the WSDL services.
  - **Bindings** - Specifies whether or not the generated documentation includes the WSDL bindings.
  - **Port Types** - Specifies whether or not the generated documentation includes the WSDL port types.
  - **Messages** - Specifies whether or not the generated documentation includes the WSDL messages.

- **XML Schema Components** - Specifies whether or not the generated documentation includes the XML Schema components.
  - **Only global elements and types** - Specifies whether or not the generated documentation includes only global elements and types.
- **Component Details**
  - **Namespace** - Presents the namespace information for WSDL or XML Schema components.
  - **Location** - Presents the location information for each WSDL or XML Schema component.
  - **Used by** - Presents the list of components that reference the current one.
  - **Documentation** - Presents the component documentation. If you choose **Escape XML Content**, the XML tags are presented in the documentation.
  - **Source** - Presents the XML fragment that defines the current component.
  - **Instance** - Generates a sample XML instance for the current component.

    Note: This option applies to the XML Schema components only.
  - **XML Schema Diagram** - Displays the diagram for each XML Schema component. You can choose the image format (JPEG, PNG, GIF, SVG) to use for the diagram section.
    - **Diagram annotations** - Specifies whether or not the annotations of the components presented in the diagram sections are included.
- **Generate index** - Displays an index with the components included in the documentation.
  - **Include local elements and attributes** - If selected, local elements and attributes are included in the documentation index.
  - **Include resource hierarchy** - Specifies whether or not the resource hierarchy for an XML Schema documentation is generated. It is deselected by default.

Export settings - Save the current settings in a settings file for further use (for example, with the exported settings file you can generate the same *documentation from the command-line interface*.)

Import settings - Reloads the settings from the exported file.

Generate - Use this button to generate the WSDL documentation.

Generating WSDL Documentation in HTML Format

The WSDL documentation generated in HTML format is presented in a visual diagram style with various sections, hyperlinks, and options.

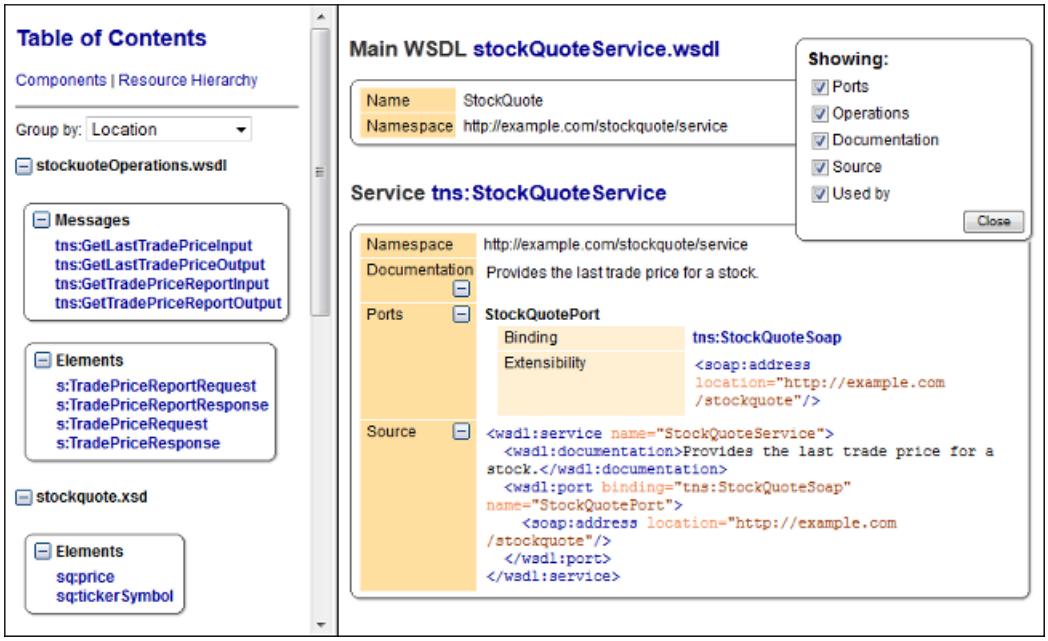


Figure 390: WSDL Documentation in HTML Format

The documentation of each component is presented in a separate section. The title of the section is composed of the component type and the component name. The component information (namespace, documentation, etc.) is presented in a tabular form.

If you choose to split the output into multiple files, the table of contents is displayed in the left frame and is divided in two tabs: **Components** and **Resource Hierarchy**.

The **Components** tab allows you to group the contents by namespace, location, or component type. The WSDL components from each group are sorted alphabetically. The **Resource Hierarchy** tab displays the dependencies between WSDL and XML Schema modules in a tree-like fashion. The root of the tree is the WSDL document that you generate documentation for.

After the documentation is generated, you can collapse or expand details for some WSDL components by using the **Showing** options or the **Collapse** or **Expand** buttons.

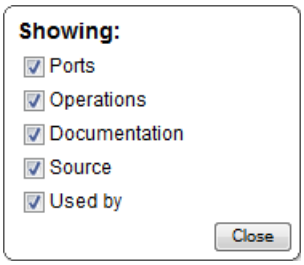


Figure 391: Showing Options

Generating WSDL Documentation in a Custom Format

To obtain the default HTML documentation output from a WSDL document, Oxygen XML Editor uses an intermediary XML document to which it applies an XSLT stylesheet. To create a custom output from your WSDL document, edit the `wsdlDocHtml.xsl` XSLT stylesheet or create your own.

Note: The `wsdlDocHtml.xsl` stylesheet that is used to obtain the HTML documentation is located in the *`[OXYGEN_INSTALL_DIR]`*`/frameworks/wsdl_documentation/xsl` folder.

Note: The intermediary XML document complies with the `wsdlDocSchema.xsd` XML Schema. This schema is located in the *`[OXYGEN_INSTALL_DIR]`*`/frameworks/wsdl_documentation` folder.

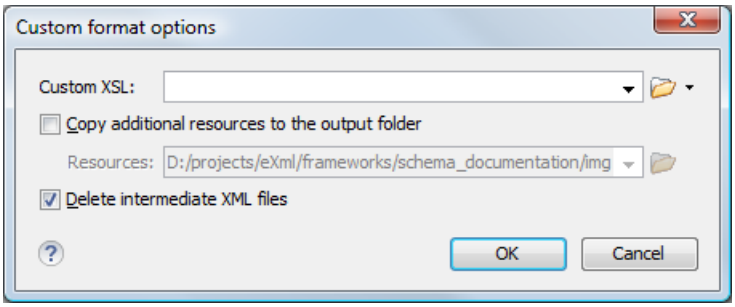


Figure 392: Custom Format Options Dialog Box

When using a custom format, you can also copy additional resources into the output folder or choose to keep the intermediate XML files created during the documentation process.

Generating WSDL Documentation from the Command-Line Interface

To generate documentation for a WSDL document from the command line, open the **WSDL Documentation** dialog box and click **Export settings**. Using the exported settings file you can generate the same documentation from the command line by running the following scripts:

- `wsdlDocumentation.bat` on Windows.
- `wsdlDocumentation.sh` on Unix / Linux.
- `wsdlDocumentationMac.sh` on Mac OS X.

The scripts are located in the installation folder of Oxygen XML Editor. You can integrate the scripts in an external batch process launched from the command-line interface.

WSDL SOAP Analyzer

WSDL SOAP Analyzer is a tool that helps you test if the messages defined in a Web Service Descriptor (WSDL) are accepted by a Web Services server.

After you edit and validate your Web service descriptor against a mix of the XML Schemas for WSDL and SOAP, it is easy to check if the defined SOAP messages are accepted by the remote Web Services server by using the integrated **WSDL SOAP Analyzer** tool (available from the toolbar or **Tools** menu).

Oxygen XML Editor provides two ways of testing, one for the currently edited WSDL document and another for the remote WSDL documents that are published on a web server. To open the **WSDL SOAP Analyzer** tool for the currently edited WSDL document do one of the following:

- Click the **WSDL SOAP Analyzer** toolbar button.
- Use the **WSDL SOAP Analyzer** action from the **Tools** menu.
- Go to **Open with** > **WSDL SOAP Analyzer** in the contextual menu of the ***Project*** view.

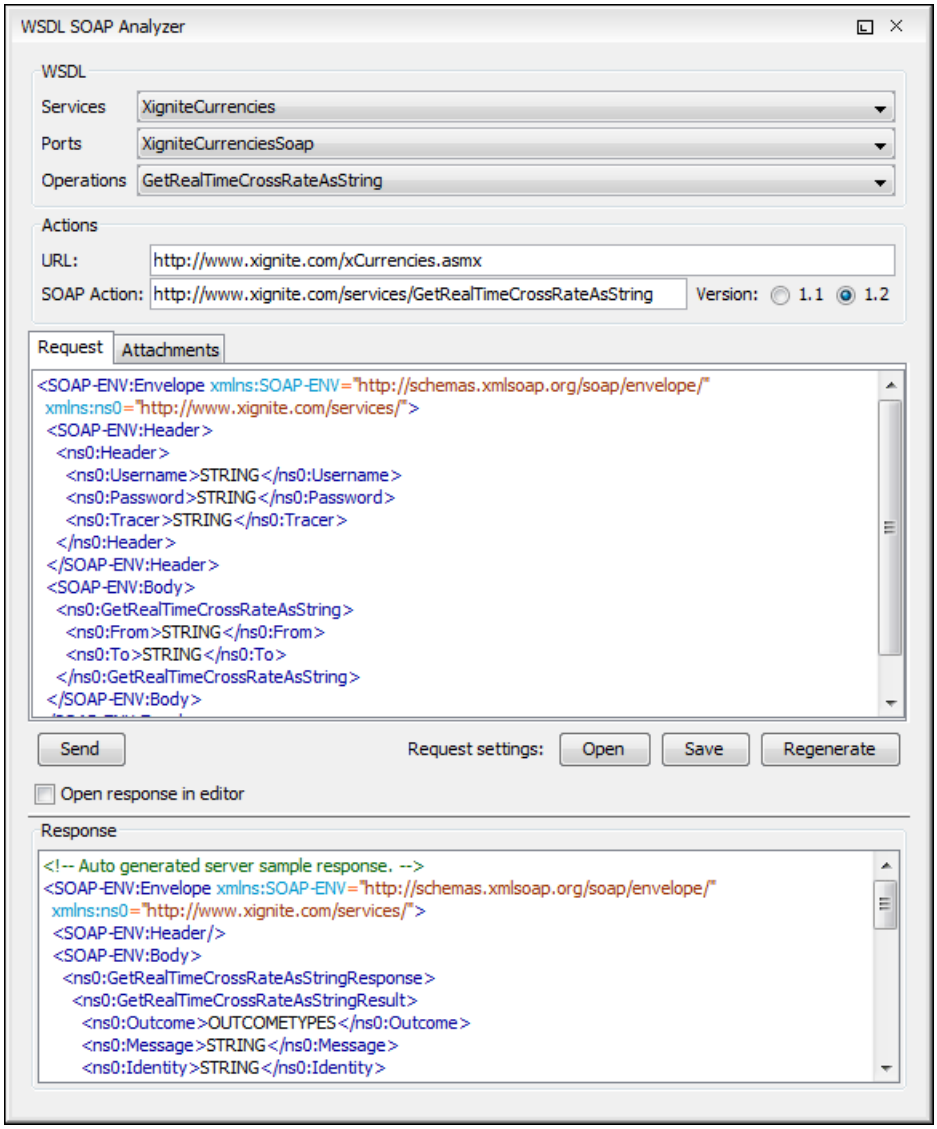


Figure 393: WSDL SOAP Analyzer Dialog Box

This tool contains a SOAP analyzer and sender for Web Services Description Language file types. The analyzer fields are as follows:

- **Services** - The list of services defined by the WSDL file.
- **Ports** - The ports for the selected service.
- **Operations** - The list of available operations for the selected service.
- **Action URL** - The script that serves the operation.
- **SOAP Action** - Identifies the action performed by the script.
- **Version** - Choose between 1.1 and 1.2. The SOAP version is selected automatically depending on the selected port.
- **Request Editor** - It allows you to compose the web service request. When an action is selected, Oxygen XML Editor tries to generate as much content as possible for the SOAP request. The envelope of the SOAP request has the correct namespace for the selected SOAP version, that is *http://schemas.xmlsoap.org/soap/envelope/* for SOAP 1.1 or *http://www.w3.org/2003/05/soap-envelope* for SOAP 1.2. Usually you just have to change a few values for the request to be valid. The *Content Completion Assistant* is available for this editor and is driven by the schema that defines the type of the current message. While selecting various operations, Oxygen XML Editor remembers the modified request for each one. You can press the **Regenerate** button to overwrite your modifications for the current request with the initial generated content.
- **Attachments List** - You can define a list of file URLs to be attached to the request.
- **Response Area** - Initially it displays an auto generated server sample response so you can have an idea about how the response looks like. After pressing the **Send** button, it presents the message received from the server in response to the Web Service request. It may show also error messages. If the response message contains attachments, Oxygen XML Editor prompts you to save them, then tries to open them with the associated system application.
- **Errors List** - There may be situations where the WSDL file is respecting the WSDL XML Schema, but it fails to be valid (for example, in the case of a message that is defined by means of an element that is not found in the types section of the WSDL). In such a case, the errors are listed here. This list is presented only when there are errors.
- **Send Button** - Executes the request. A status dialog box is displayed when Oxygen XML Editor is connecting to the server.

The testing of a WSDL file is straight-forward: click the WSDL analysis button, then select the service, the port, and the operation. The editor generates the skeleton for the SOAP request. You can edit the request, eventually attach files to it and send it to the server. Watch the server response in the response area. You can find more details in the *Testing Remote WSDL Files* section.

Note: SOAP requests and responses are automatically validated in the **WSDL SOAP Analyzer** using the XML Schemas specified in the WSDL file.

Once defined, a request derived from a Web Service descriptor can be saved with the **Save** button to a Web Service SOAP Call (WSSC) file for later reuse. In this way, you save time in configuring the URLs and parameters.

You can open the result of a Web Service call in an editor panel using the **Open** button.

Testing Remote WSDL Files

To open and test a remote WSDL file the steps are the following:

1. Go to **Tools** > **WSDL SOAP Analyzer** .
2. On the **WSDL File** tab enter the URL of the remote WSDL file.
   You enter the URL:
   - by typing
   - by browsing the local file system
   - by browsing a remote file system
   - by browsing *a UDDI Registry*
3. Press the **OK** button.
   This will open the ***WSDL SOAP Analyzer** tool*. In the **Saved SOAP Request** tab you can open directly a previously saved Web Service SOAP Call (WSSC) file, thus skipping the analysis phase.

UDDI Registry Browser

Pressing the button in the **WSDL File Opener** dialog box (menu **Tools** > **WSDL SOAP Analyzer**) opens the **UDDI Registry Browser** dialog box.

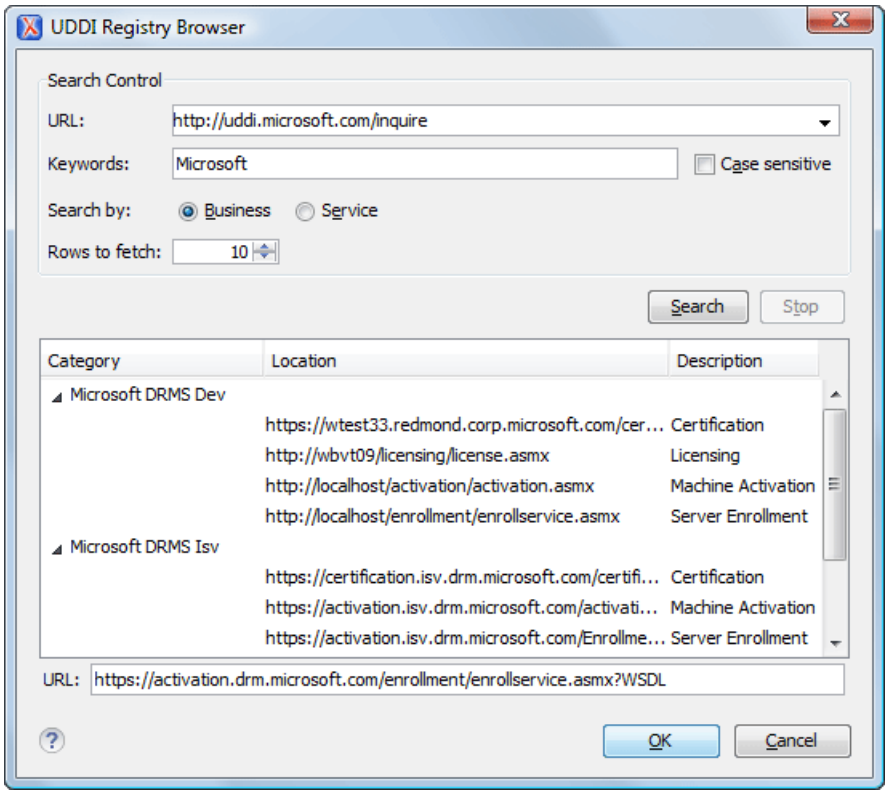


Figure 394: UDDI Registry Browser Dialog Box

The fields of the dialog box are as follows:

- **URL** - Type the URL of an UDDI registry or choose one from the default list.
- **Keywords** - Enter the string you want to be used when searching the selected UDDI registry for available Web services.
- **Rows to fetch** - The maximum number of rows to be displayed in the result list.
- **Search by** - You can choose to search either by company or by provided service.
- **Case sensitive** - When selected, the search takes into account the keyword case.
- **Search** - The WSDL files that matched the search criteria are added in the result list.

When you select a WSDL from the list and click the **OK** button, the **UDDI Registry Browser** dialog box is closed and you are returned to the **WSDL File Opener** dialog box.

Editing CSS Stylesheets

Oxygen XML Editor includes a built-in editor for CSS stylesheets. This section presents the features of the CSS editor and how these features should be used. The features of the CSS editor include:

- **Create new CSS files and templates** - You can use the built-in new file wizards to *create new CSS documents or templates*.
- **Open and Edit CSS files** - CSS files can be opened and edited in a source editing mode.
- **Validation** - Presents validation errors in CSS files.
- **Content completion** - Offers proposals for properties and the values that are available for each property.
- **Syntax highlighting** - The syntax highlighting in Oxygen XML Editor makes CSS files more readable.

- **Shortcut to open resources** - You can use **Ctrl + Single-Click (Command + Single-Click on OS X)** to open imported stylesheets or other resources (such as images) in the default system application for the particular type of resource.

Validating CSS Stylesheets

Oxygen XML Editor includes a built-in *CSS Validator*, integrated with general validation support. This makes the *usual validation features* for presenting errors also available for CSS stylesheets.

When you edit a CSS document, you can access the *CSS validator options* by selecting **Validation options** from the **Document** > **Validate** menu.

The CSS properties accepted by the validator are those included in the current CSS profile that is selected in *the CSS validation preferences*. The **CSS 3 with Oxygen extensions** profile includes all the CSS 3 standard properties plus the *CSS extensions specific for Oxygen* that can be used in *Author mode*. That means all Oxygen specific extensions are accepted in a CSS stylesheet by *the built-in CSS validator* when this profile is selected.

Specify Custom CSS Properties

To specify custom CSS properties, follow these steps:

1. Create a file named `CustomProperties.xml` that has the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<css_keywords
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.oxygenxml.com/ns/css
 http://www.oxygenxml.com/ns/css/CssProperties.xsd"
      xmlns="http://www.oxygenxml.com/ns/css">
      <property name="custom">
      <summary>Description for custom property.</summary>
        <value name="customValue"/>
        <value name="anotherCustomValue"/>
      </property>
</css_keywords>
```

2. Go to your desktop and create the `builtin/css-validator/` folder structure.
3. Press and hold **Shift** and right-click anywhere on your desktop. From the contextual menu, select **Open Command Window Here**.
4. In the command line, run the `jar cvf custom_props.jar builtin/` command.
   The `custom_props.jar` file is created.
5. Go to *`[OXYGEN_INSTALL_DIR]`*`/lib` and create the `endorsed` folder. Copy the `custom_props.jar` file to *`[OXYGEN_INSTALL_DIR]`*`/lib/endorsed`.

Content Completion in CSS Stylesheets

A *Content Completion Assistant*, similar to *the one available for XML documents* offers the CSS properties and the values available for each property. It can be manually activated with the **Ctrl + Space (Command + Space on OS X)** shortcut and is context-sensitive when invoked for the value of a property. The *Content Completion Assistant* also includes *code templates that can be used to quickly insert code fragments* into CSS stylesheets. The code templates that are proposed include form controls, actions, and **Author** mode operations.

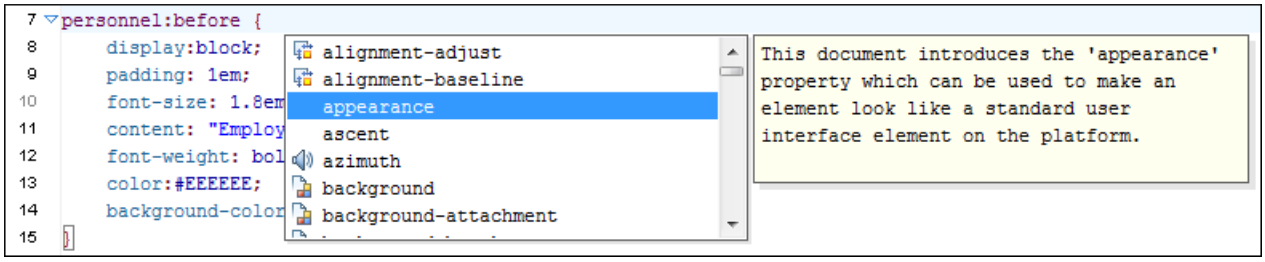


Figure 395: Content Completion in CSS Stylesheets

The properties and values available are dependent on the CSS Profile selected in the ***CSS** preferences*. The CSS 2.1 set of properties and property values is used for most of the profiles. However, with CSS 1 and CSS 3 specific proposal sets are used.

The profile *CSS 3 with Oxygen extensions* includes all the CSS 3 standard properties plus the *CSS extensions specific for Oxygen XML Editor* that can be used in ***Author** mode*.

Proposals for CSS Selectors - After inserting a *CSS selector*, the content completion assistance will propose a list of pseudo-elements and pseudo-classes that are available for the selected CSS profile.

Proposals for `@media` and `@import` Rules - After inserting `@media` or `@import <url>` rules, the content completion assistance will propose a list of supported media types.

Related Information:

Specify Custom CSS Properties on page 650

Syntax Highlighting in CSS Files

Oxygen XML Editor supports syntax highlighting in CSS files to improve the readability of the content and reduce the time it takes to internalize the semantics of the structured content.

To customize the colors or styles used for the syntax highlighting colors for CSS files, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)***.
2. Go to ***Editor*** > ***Syntax Highlight***.
3. Select and expand the **CSS** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

Related Information:

Syntax Highlight Preferences on page 104

CSS Outline View

The **Outline** view for CSS stylesheets presents the import declarations for other CSS stylesheet files and all the selectors defined in the current CSS document. The selector entries can be presented as follows:

- In the order they appear in the document.
- Sorted by the element name used in the selector.
- Sorted by the entire selector string representation.

You can synchronize the selection in the **Outline** view with the cursor moves or changes you make in the stylesheet document. When you select an entry from the **Outline** view, Oxygen XML Editor highlights the corresponding import or selector in the CSS editor.

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

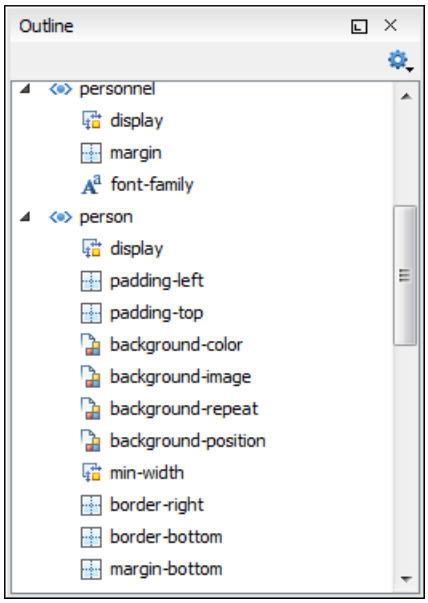


Figure 396: CSS Outline View

The selectors presented in this view can be found quickly using the key search field. When you press a sequence of character keys while the focus is in the view, the first selector that starts with that sequence is selected automatically.

Folding in CSS Stylesheets

In a large CSS stylesheet document, some styles can be collapsed so that only the styles that are needed remain in focus. The same *folding features available for XML documents* are also available in CSS stylesheets.

Note: To enhance your editing experience, you can select entire blocks (parts of text delimited by brackets) by double-clicking somewhere inside the brackets.

Formatting and Indenting CSS Stylesheets (Pretty Print)

If the edited CSS stylesheet becomes unreadable because of the bad alignment of the text lines, the *format and indent operation available for XML documents* is also available for CSS stylesheets. It works in the same way as for XML documents and is available as the same menu and toolbar action.

Minifying CSS Stylesheets

Minification (or *compression*) of a CSS document is the practice of removing unnecessary code without affecting the functionality of the stylesheet.

To minify a CSS, invoke the contextual menu anywhere in the edited document and choose the **Minify CSS** action. Oxygen XML Editor opens a dialog box that allows you to:

- Set the location of the resulting CSS.
- Place each style rule on a new line.

After pressing **OK**, Oxygen XML Editor performs the following actions:

- All spaces are normalized (all leading and trailing spaces are removed, while sequences of white spaces are replaced with single space characters).
- All comments are removed.

Note: The CSS minifier relies heavily upon the W3C CSS specification. If the content of the CSS file you are trying to minify does not conform with the specifications, an error dialog box will be displayed, listing all errors encountered during the processing.

The resulting CSS stylesheet gains a lot in terms of execution performance, but loses in terms of readability. The source CSS document is left unaffected.

Note: To restore the readability of a minified CSS, invoke the **Format and Indent** action from the **Document** > **Source** menu, the **Source** submenu from the contextual menu, or **Source** toolbar. However, this action will not recover any of the deleted comments.

Editing LESS CSS Stylesheets

Oxygen XML Editor provides support for stylesheets coded with the LESS dynamic stylesheet language. LESS extends the CSS language by adding features that allow mechanisms such as *variables*, *nesting*, *mixins*, *operators*, and *functions*. Oxygen XML Editor offers additional LESS features that include:

- **Create new LESS files and templates** - You can use the built-in new file wizards to *create new LESS documents or templates*.
- **Open and Edit LESS files** - LESS files can be opened and edited in a source editing mode.
- **Validation** - Presents validation errors in LESS files.
- **Content completion** - Offers proposals for properties and the values that are available for each property.
- **Compile to CSS** - Options are available to compile LESS files to CSS.
- **Syntax highlighting** - Oxygen XML Editor supports syntax highlighting in LESS files, although there may be some limitations in supporting all the LESS constructs.
- **Shortcut to open resources** - While editing LESS files, you can use **Ctrl + Single-Click (Command + Single-Click on OS X)** to open imported stylesheets or other resources (such as images) in the default system application for the particular type of resource.

For more information about LESS go to *http://lesscss.org/*.

Validating LESS Stylesheets

Oxygen XML Editor includes a built-in *LESS CSS Validator*, integrated with general validation support. The *usual validation features* for presenting errors also available for LESS stylesheets.

Oxygen XML Editor provides three validation methods:

- Automatic validation as you type - marks validation errors in the document as you are editing.
- Validation upon request, by pressing the **Validate** button from the **Validation** toolbar drop-down menu. An error list is presented in the message panel at the bottom of the editor.
- Validation scenarios, by selecting **Configure Validation Scenario(s)** from the **Validation** toolbar drop-down menu. Errors are presented in the message panel at the bottom of the editor. This is useful when you need to validate the current file as part of a larger LESS import hierarchy (for instance, you may change the URL of the file to validate to the root of the hierarchy).

Content Completion in LESS Stylesheets

A *Content Completion Assistant* offers the LESS properties and the values available for each property. It can be manually activated with the **Ctrl + Space (Command + Space on OS X)** shortcut and is context-sensitive when invoked for the value of a property in a LESS file. The *Content Completion Assistant* also includes *code templates that can be used to quickly insert code fragments* into LESS stylesheets. The code templates that are proposed include form controls, actions, and **Author** mode operations.

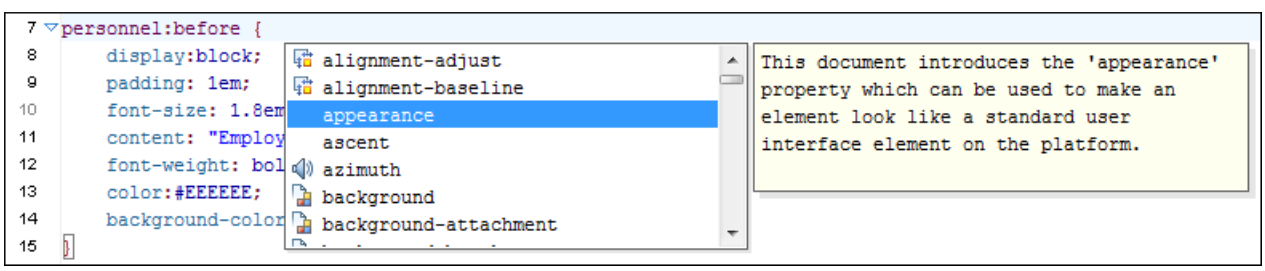


Figure 397: Content Completion in LESS Stylesheets

The properties and values available are dependent on the CSS Profile selected in the *CSS preferences* .

Syntax Highlighting in LESS Files

Oxygen XML Editor supports syntax highlighting in LESS files to improve the readability of the content and reduce the time it takes to internalize the semantics of the structured content.

To customize the colors or styles used for the syntax highlighting colors for LESS files, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)***.
2. Go to ***Editor*** > ***Syntax Highlight***.
3. Select and expand the **LESS** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

Related Information:

Syntax Highlight Preferences on page 104

Compiling LESS Stylesheets to CSS

When editing LESS files, you can compile the files into CSS. Oxygen XML Editor provides both manual and automatic options to compile LESS stylesheets into CSS.

Important: The LESS processor works well only with files having the *UTF-8* encoding. Thus, it is highly recommended that you always use the `utf-8` encoding when working with LESS files or the files they import (other LESS or CSS files). You can use the following directive at the beginning of your files:

```
@charset "utf-8";
```

You have two options for compiling LESS files to CSS:

1. Use the contextual menu in a LESS file and select **Compile to CSS (Ctrl + Shift + C (Command + Shift + C on OS X))**.
2. Select the ***Automatically compile LESS to CSS when saving** option* in the settings (*open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Open/Save** > **Save Hooks**). If selected, when you save a LESS file it will automatically be compiled to CSS (this option is deselected by default).

   Important: If this option is selected, when you save a LESS file, the CSS file that has the same name as the LESS file is overwritten without warning. Make sure all your changes are made in the LESS file. Do not edit the CSS file directly, as your changes might be lost.

Editing Relax NG Schemas

An XML Schema describes the structure of an XML document and is used to validate XML document instances against it, to check that the XML instances conform to the specified requirements. If an XML instance conforms to the schema then it is said to be valid. Otherwise, it is invalid.

Oxygen XML Editor offers support for editing Relax NG schema files in the following editing modes:

- ***Text editing mode*** - Allows you to edit Relax NG schema files in a source editing mode, along with a schema design pane with two tabs that offer a ***Full Model View*** and ***Logical Model View***.
- ***Grid editing mode*** - Displays Relax NG schema files in a structured spreadsheet-like grid.
- ***Author editing mode*** - The visual **Author** mode is also available for Relax NG schema files, presenting the schema similar to the Relax NG compact syntax. It links to imported schemas and external references. Embedded Schematron is also supported in Relax NG schemas with XML syntax.

For information about applying and detecting schemas, see *Associating a Schema to XML Documents* on page 464.

Related Information:

Associating a Schema to XML Documents on page 464

Editing Relax NG Schema in the Master Files Context

Smaller interrelated modules that define a complex Relax NG Schema cannot be correctly edited or validated individually, due to their interdependency with other modules. For example, an element defined in a main schema document is not visible when you edit an included module. Oxygen XML Editor provides the support for defining the main module (or modules), thus allowing you to edit any of the imported/included schema files in the context of the larger schema structure.

You cat set a main Relax NG document either using the *master files support from the **Project** view*, or using a validation scenario.

To set a *master file* using a validation scenario, add validation units that point to the main schemas. Oxygen XML Editor warns you if the current module is not part of the dependencies graph computed for the main schema. In this case, it considers the current module as the main schema.

The advantages of editing in the context of *master file* include:

- Correct validation of a module in the context of a larger schema structure.
- *Content Completion Assistant* displays all the referable components valid in the current context. This include components defined in modules other than the currently edited one.
- The ***Outline** view* displays the components collected from the entire schema structure.

Related Information:

Creating a New Validation Scenario on page 452
XML Schema Outline View on page 242

Relax NG Schema Diagram Editor

This section explains how to use the graphical diagram editor for Relax NG schemas.

Introduction to Relax NG Schema Diagram Editor

Oxygen XML Editor provides a simple, expressive, and easy-to-read schema diagram editor for Relax NG schemas.

With this new feature you can easily develop complex schemas, print them on multiple pages or save them as JPEG, PNG, or BMP images. It helps both schema authors in developing the schema and content authors who are using the schema to understand it.

Oxygen XML Editor is the only XML editor to provide a side by side source and diagram presentation and have them real-time synchronized:

- The changes you make in the Editor are immediately visible in the Diagram (no background parsing).
- Changing the selected element in the diagram selects the underlying code in the source editor.

Full Model View

When you create a new schema document or open an existing one, the editor panel is divided in two sections: one containing the schema diagram and the second the source code. The schema diagram editor has two tabs that offer a **Full Model View** and ***Logical Model View***.

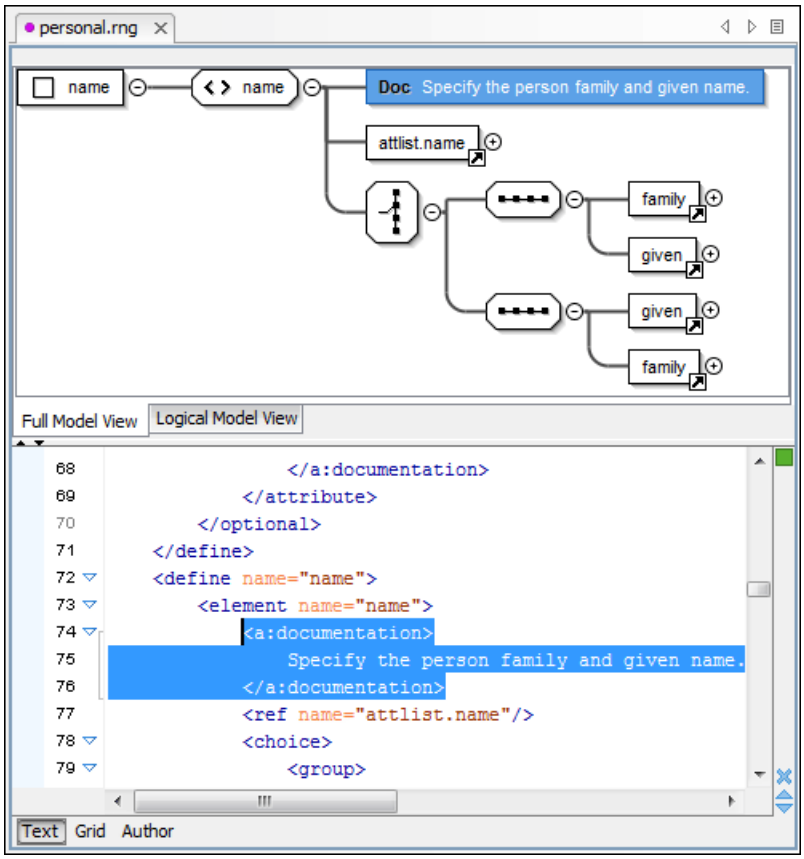


Figure 398: Relax NG Schema Editor - Full Model View

The following references can be expanded in place: patterns, includes, and external references. This expansion mechanism, coupled with the synchronization support, makes the schema navigation easy.

All the element and attribute names are editable by double-clicking the names.

Logical Model View

The **Logical Model View** presents the compiled schema in the form of a single pattern. The patterns that form the element content are defined as top level patterns with generated names. These names are generated depending of the elements name class.

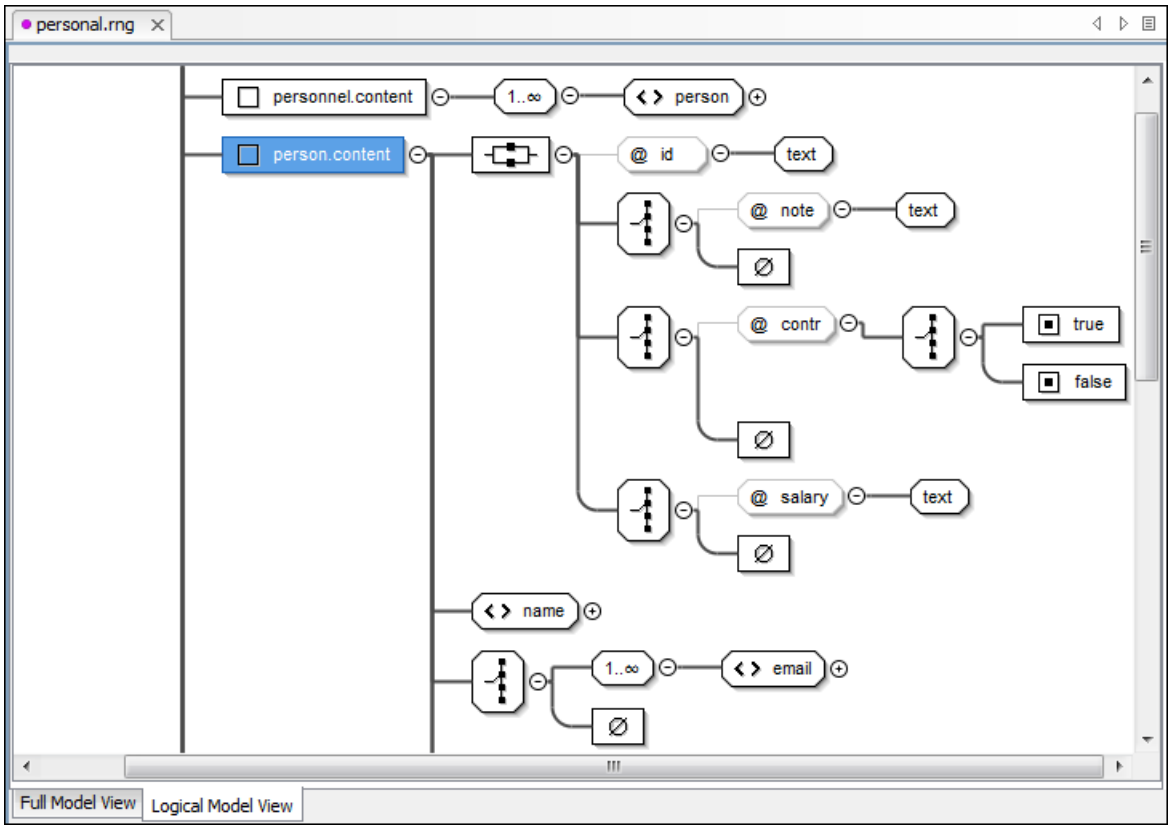
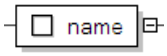
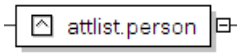


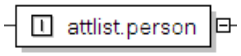
Figure 399: Logical Model View for a Relax NG Schema

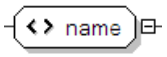
Symbols Used in the Schema Diagram

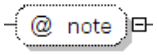
The views in the schema diagram editor renders all the Relax NG schema patterns with the following intuitive symbols:

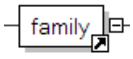
- 

- `define` pattern with the `name` attribute set to the value shown inside the rectangle (in this example `name`).
- 

- `define` pattern with the `combine` attribute set to `interleave` and the `name` attribute set to the value shown inside the rectangle (in this example `attlist.person`).
- 

- `define` pattern with the `combine` attribute set to `choice` and the `name` attribute set to the value shown inside the rectangle (in this example `attlist.person`).
- 

- `element` pattern with the `name` attribute set to the value shown inside the rectangle (in this example `name`).
- 

- `attribute` pattern with the `name` attribute set to the value shown inside de rectangle (in this case `note`).
- 

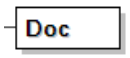
- `ref` pattern with the `name` attribute set to the value shown inside the rectangle (in this case `family`).
-

- `oneOrMore` pattern.
-

- `zeroOrMore` pattern.

-

- `optional` pattern.
-
- `choice` pattern.
-

- `value` pattern (for example, used inside a `choice` pattern).
-
- `group` pattern.
- 

- A pattern from the Relax NG Annotations namespace (http://relaxng.org/ns/compatibility/annotations/1.0) that is treated as a documentation element in a Relax NG schema.
-

- `text` pattern.
-
- `empty` pattern.

Actions Available in the Schema Diagram Editor

When editing Relax NG schemas in ***Full Model View***, the contextual menu offers the following actions:

Append child

Appends a child to the selected component.

Insert Before

Inserts a component before the selected component.

Insert After

Inserts a component after the selected component.

Edit attributes

Edits the attributes of the selected component.

Remove

Removes the selected component.

Show only the selected component

Depending on its state (selected/not selected), either the selected component or all the diagram components are shown.

Show Annotations

Depending on its state (selected/not selected), the documentation nodes are shown or hidden.

Auto expand to references

This option controls how the schema diagram is automatically expanded. If you select it and then edit a top-level element or you make a refresh, the diagram is expanded until it reaches referenced components. If this option is left unchecked, only the first level of the diagram is expanded, showing the top-level elements. For large schemas, the editor disables this option automatically.

Collapse Children

Collapses the children of the selected view.

Expand Children

Expands the children of the selected view.

Print Selection

Prints the selected view.

Save as Image

Saves the current selection as JPEG, BMP, SVG or PNG image.

Refresh

Refreshes the schema diagram according to the changes in your code. They represent changes in your imported documents or changes that are not reflected automatically in the compiled schema).

If the schema is not valid, you see only an error message in the ***Logical Model View*** instead of the diagram.

Validating Relax NG Schema Documents

By default, Relax NG schema files are validated as you type. To change this, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Editor** > **Document Checking**, and deselect the ***Enable automatic validation** option*.

To validate a Relax NG schema document manually, select the **Validate** action from the **Validation** toolbar drop-down menu or the **Document** > **Validate** menu. When Oxygen XML Editor validates a Relax NG schema file, it expands all the included modules so the entire schema hierarchy is validated. The validation problems are highlighted directly in the editor, making it easy to locate and fix any issues.

Related Information:

Validating XML Documents Against a Schema on page 446
Validation Scenario on page 452
Associating a Schema to XML Documents on page 464
Presenting Validation Errors in Author Mode on page 236
Presenting Validation Errors in Text Mode on page 206

Content Completion in Relax NG Schemas

The intelligent *Content Completion Assistant* allows you to quickly identify and insert elements, attributes, and attribute values that are valid in the current editing context. All available proposals are listed in a pop-up menu displayed at the current cursor position.

The *Content Completion Assistant* is enabled by default. To disable it, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Editor** > **Content Completion**, and deselect the ***Enable content completion** option*.

When active, the *Content Completion Assistant* displays a list of context-sensitive proposals valid at the current cursor position. It can be manually activated with the **Ctrl + Space (Command + Space on OS X)** shortcut. You can navigate through the list of proposals by using the **Up** and **Down** keys on your keyboard. For each selected item in the list, the *Content Completion Assistant* displays a documentation window. You can customize the size of the documentation window by dragging its top, right, and bottom borders.

To insert the selected content in **Text** mode, do one of the following:

- Press **Enter** or **Tab** to insert both the start and end tags and position the cursor inside the start tag in a position suitable for inserting attributes.
- Press **Ctrl + Enter (Command + Enter on OS X)** to insert both the start and end tags and positions the cursor between the tags in a position where you can start typing content.

If you are using the concept of *master files* to import/include modules, the *Content Completion Assistant* collects its components starting from the *master files*. The *master files* can be defined in the project or in the associated validation scenario. For more information about the *Master Files* support in Oxygen XML Editor, see *Defining Master Files at Project Level*.

The *Content Completion Assistant* also offers additional information for the element and attribute proposals in the form of schema annotations that is displayed in a tooltip.

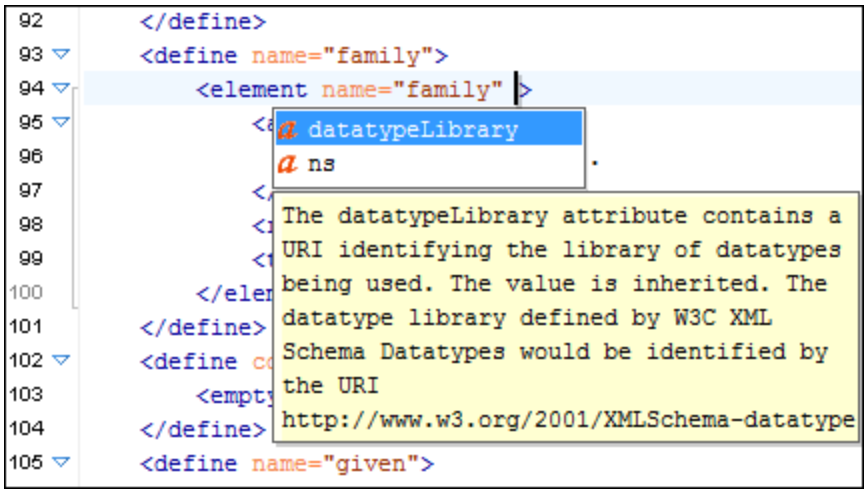


Figure 400: Relax NG Content Completion Assistant

Syntax Highlighting in Relax NG Schemas

Oxygen XML Editor supports syntax highlighting in Relax NG schemas to improve the readability of the content and reduce the time it takes to internalize the semantics of the structured content.

To customize the colors or styles used for the syntax highlighting colors for Relax NG schemas, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)***.
2. Go to ***Editor*** > ***Syntax Highlight***.
3. Select and expand the **XML** section in the top pane (for RELAX NG Compact Syntax schemas, select and expand the **RNC** section).
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the **XML** tab in the **Preview** pane to see the effects of your changes (for RELAX NG Compact Syntax schemas, the tab is **RNC**).

Tip: Oxygen XML Editor also allows you to specify syntax highlighting colors for specific XML elements and attributes with specific namespace prefixes. This can be done in the ***Editor*** > ***Syntax Highlight*** > ***Elements/ Attributes by Prefix*** *preferences page*.

Related Information:

Syntax Highlight Preferences on page 104

Quick Fixes for DTD, XSD, and Relax NG Errors

Oxygen XML Editor offers *Quick Fixes* for common errors that appear in XML documents that are validated against DTD, XSD, or Relax NG schemas.

Note: For XML documents validated against XSD schemas, the *Quick Fixes* are only available if you use the default Xerces validation engine.

Quick Fixes are available in **Text** mode and **Author** mode.

Oxygen XML Editor provides *Quick Fixes* for numerous types of problems, including the following:

Problem Type	Available Quick Fixes
A specific element is required in the current context	Insert the required element
An element is invalid in the current context	Remove the invalid element
The content of the element should be empty	Remove the element content

Problem Type	Available Quick Fixes
An element is not allowed to have child elements	Remove all child elements
Text is not allowed in the current element	Remove the text content
A required attribute is missing	Insert the required attribute
An attribute is not allowed to be set for the current element	Remove the attribute
The attribute value is invalid	Propose the correct attribute values
ID value is already defined	Generate a unique ID value
References to an invalid ID	Change the reference to an already defined ID

Related Information:

Schematron Quick Fixes (SQF) on page 463

Relax NG Outline View

The **Outline** view for Relax NG schemas presents a list with the patterns that appear in the diagram in both the ***Full Model View*** and ***Logical Model View*** cases and it allows for quick access to a component by name. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Figure 401: Relax NG Outline View

This view has two modes, with the tree showing either the XML structure or the defined pattern (components) collected from the current document. By default, the **Outline** view presents the components.

When the **Show components** option is selected in the **Settings** menu on the **Outline** view toolbar, the following option is available:

Show XML structure

Shows the XML structure of the current document in a tree-like manner.

The following actions are available in the **Settings** menu on the **Outline** view toolbar when the **Show XML structure** option is selected:

Filter returns exact matches

The text filter of the **Outline** view returns only exact matches.

Selection update on cursor move

Allows a synchronization between **Outline** view and schema diagram. The selected view from the diagram will be also selected in the **Outline** view.

Show components

Shows the defined pattern collected from the current document.

Flat presentation mode of the filtered results

When active, the application flattens the filtered result elements to a single level.

Show comments and processing instructions

Show/hide comments and processing instructions in the **Outline** view.

Show element name

Show/hide element name.

Show text

Show/hide additional text content for the displayed elements.

Show attributes

Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from *the Outline preferences panel*.

Configure displayed attributes

Displays the ***XML Structured Outline*** *preferences page*.

The following contextual menu actions are also available in the **Outline** view when the **Show XML structure** option is selected in the **Settings** menu:

Append Child

Displays a list of elements that you can insert as children of the current element.

Insert Before

Displays a list of elements that you can insert as siblings of the current element, before the current element.

Insert After

Displays a list of elements that you can insert as siblings of the current element, after the current element.

Edit Attributes

Opens a dialog box that allows you to edit the attributes of the currently selected component.

Toggle Comment

Comments/uncomments the currently selected element.

Search references

Searches for the references of the currently selected component.

Search references in

Searches for the references of the currently selected component in the context of a scope that you define.

Component dependencies

Opens the ***Component Dependencies** view* that displays the dependencies of the currently selected component.

Rename Component in

Renames the currently selected component in the context of a scope that you define.

Cut

Cuts the currently selected component.

Copy

Copies the currently selected component.

Delete

Deletes the currently selected component.

Expand More

Expands the structure of a component in the **Outline** view.

Collapse All

Collapses the structure of all the component in the **Outline** view.

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (*, ?) and separate multiple patterns with commas.

RNG Resource Hierarchy/Dependencies View

The **Resource Hierarchy/Dependencies** view allows you to see the hierarchy/dependencies for a schema. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

If you want to see the hierarchy of a schema, select the desired schema in the ***Project** view* and choose **Resource Hierarchy** from the contextual menu.

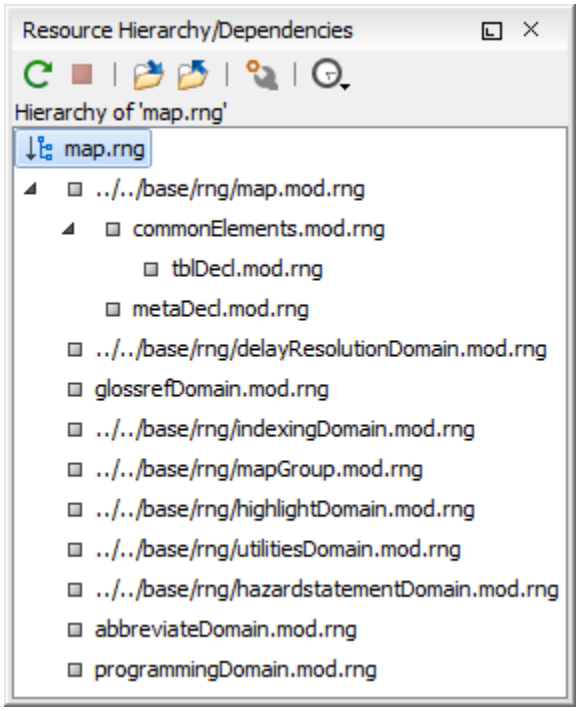


Figure 402: Resource Hierarchy/Dependencies View - hierarchy for `map.rng`

If you want to see the dependencies of a schema, select the desired schema in the **Project** view and choose **Resource Dependencies** from the contextual menu.

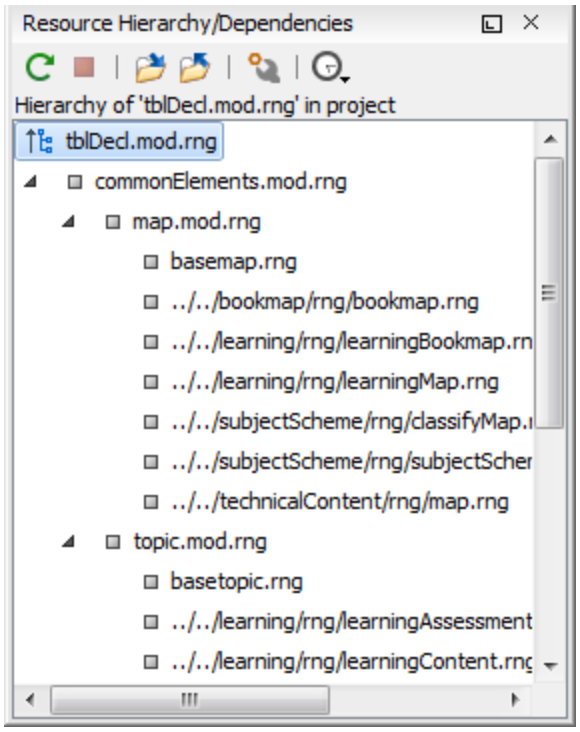


Figure 403: Resource Hierarchy/Dependencies View - Dependencies for `tblDecl.mod.rng`

The following actions are available in the **Resource Hierarchy/Dependencies** view:

Refresh

Refreshes the Hierarchy/Dependencies structure.

Stop

Stops the hierarchy/dependencies computing.

Show Hierarchy

Allows you to choose a resource to compute the hierarchy structure.

Show Dependencies

Allows you to choose a resource to compute the dependencies structure.

Configure

Allows you to configure a scope to compute the dependencies structure. There is also an option for automatically using the defined scope for future operations.

History

Provides access to the list of previously computed dependencies. Use the **Clear history** button to remove all items from this list.

The contextual menu contains the following actions:

Open

Opens the resource. You can also double-click a resource in the Hierarchy/Dependencies structure to open it.

Copy location

Copies the location of the resource.

Move resource

Moves the selected resource.

Rename resource

Renames the selected resource.

Show Resource Hierarchy

Shows the hierarchy for the selected resource.

Show Resource Dependencies

Shows the dependencies for the selected resource.

Add to Master Files

Adds the currently selected resource in *the **Master Files** directory*.

Expand All

Expands all the children of the selected resource from the Hierarchy/Dependencies structure.

Collapse All

Collapses all children of the selected resource from the Hierarchy/Dependencies structure.

Tip: When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon .

Note: The **Move resource** or **Rename resource** actions give you the option to *update the references to the resource*.

Related Information:

Working with Modular XML Files in the Master Files Context on page 481

Search and Refactor Operations Scope on page 480

Moving/Renaming RNG Resources

You can move and rename a resource presented in the **Resource/Hierarchy Dependencies** view, using the **Rename resource** and **Move resource** refactoring actions from the contextual menu.

When you select the **Rename** action in the contextual menu of the **Resource/Hierarchy Dependencies** view, the **Rename resource** dialog box is displayed. The following fields are available:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references** - Select this option to update the references to the resource you are renaming.

When you select the **Move** action from the contextual menu of the **Resource/Hierarchy Dependencies** view, the **Move resource** dialog box is displayed. The following fields are available:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, in accordance with the new location and name.

If the **Update references of the moved resource(s)** option is selected, a **Preview** option (which opens the **Preview** dialog box) is available for both actions. The **Preview** dialog box presents a list with the resources that are updated.

Note: Updating the references of a resource that is resolved through a catalog is not supported. Also, the update references operation is not supported if the path to the renamed or moved resource contains entities.

Component Dependencies View for RelaxNG Schemas

The **Component Dependencies** view allows you to see the dependencies for a selected Relax NG component. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

If you want to see the dependencies of a RelaxNG component, select the desired component in the editor and choose the **Component Dependencies** action from the contextual menu. The action is available for all named defines.

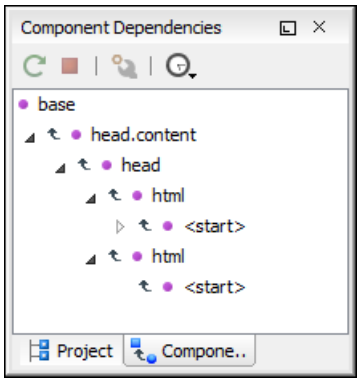


Figure 404: Component Dependencies View - Hierarchy for `base.rng`

In the Component Dependencies view you have several actions in the toolbar:

Refresh

Refreshes the dependencies structure.

Stop

Stops the dependencies computing.

Configure

Allows you to configure a search scope to compute the dependencies structure. You can decide to use automatically the defined scope for future operations by selecting the corresponding checkbox.

History

Allows you to repeat a previous dependencies computation.

The following actions are available on the contextual menu:

Go to First Reference

Selects the first reference of the referenced component from the current selected component in the dependencies tree.

Go to Component

Shows the definition of the current selected component in the dependencies tree.

Tip: If a component contains multiple references to another components, a small table is displayed that contains all references. When a recursive reference is encountered, it is marked with a special icon .

Related Information:

Search and Refactor Operations Scope on page 480

Searching and Refactoring Actions in RNG Schemas

Search Actions

The following search actions can be applied on named *defines* and are available from the **Search** submenu in the contextual menu of the current editor or from the **Document** > **References** menu:

- **Search References** - Searches all references of the item found at current cursor position in the defined scope, if any. If a scope is defined, but the current edited resource is not part of the range of resources determined by this, a warning dialog box is displayed and you have the possibility to define another search scope.
- **Search References in** - Searches all references of the item found at current cursor position in the file or files that you specify when define a scope in the **Search References** dialog box.
- **Search Declarations** - Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined, but the current edited resource is not part of the range of resources

determined by this, a warning dialog box will be displayed and you have the possibility to define another search scope.

- **Search Declarations in** - Searches all declarations of the item found at current cursor position in the file or files that you specify when you define a scope for the search operation.
- **Search Occurrences in File** - Searches all occurrences of the item at the cursor position in the currently edited file.

The following action is available from the contextual menu and the **Document** > **Schema** menu:

- **Show Definition** - Moves the cursor to the definition of the current element in the Relax NG (full syntax) schema.

  Note: You can also use the **Ctrl + Single-Click (Command + Single-Click on OS X)** shortcut on a reference to display its definition.

Refactoring Actions

The following refactoring actions can be applied on named *defines* and are available from the **Refactoring** submenu in the contextual menu of the current editor or from the **Document** > **Refactoring** menu:

- **Rename Component** - Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the **Esc** or **Enter** key on your keyboard.
- **Rename Component in** - Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the **Preview** button, you can view the files to be affected by the action.

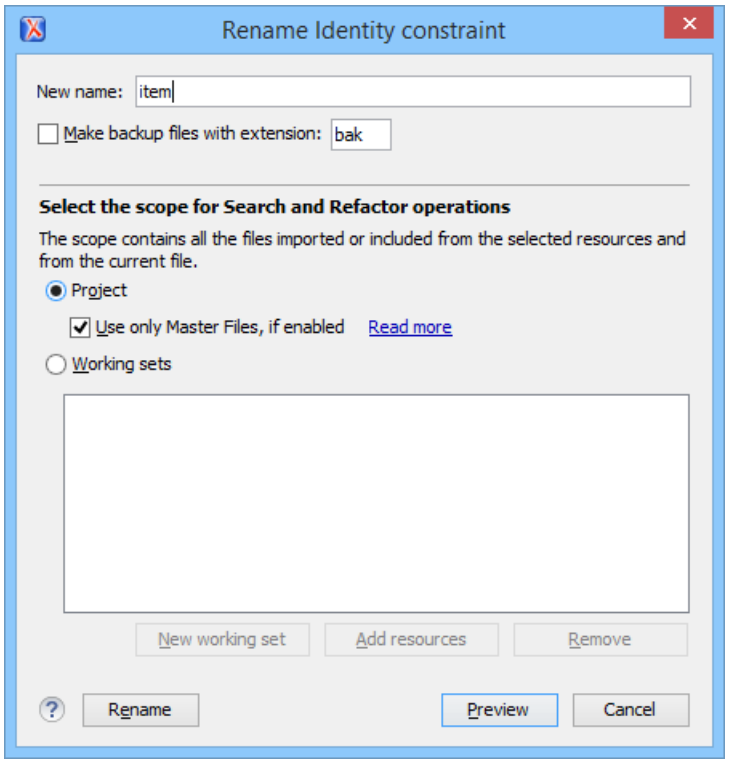


Figure 405: Rename Identity Constraint Dialog Box

RNG Quick Assist Support

The *Quick Assist support* improves the development work flow, offering fast access to the most commonly used actions when you edit schema documents.

The *Quick Assist feature* is activated automatically when the cursor is positioned over the name of a component. It is accessible via a yellow bulb icon ( ) placed at the current line in the stripe on the left side of the editor. Also, you can invoke the *quick assist* menu by using the **Alt + 1** (**Meta + Alt + 1** on Mac OS X) keyboard shortcuts.

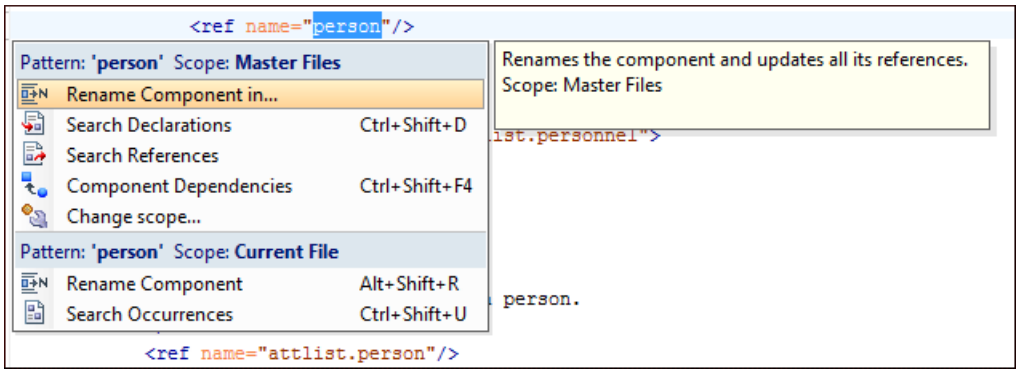


Figure 406: RNG Quick Assist Support

The *Quick Assist* support offers direct access to the following actions:

Rename Component in

Renames the component and all its dependencies.

Search Declarations

Searches the declaration of the component in a predefined scope. It is available only when the context represents a component name reference.

Search References

Searches all references of the component in a predefined scope.

Component Dependencies

Searches the component dependencies in a predefined scope.

Change Scope

Configures the scope that will be used for future search or refactor operations.

Rename Component

Allows you to rename the current component in-place.

Search Occurrences

Searches all occurrences of the component within the current file.

Related Information:

Component Dependencies View on page 665
Resource Hierarchy/Dependencies View on page 663
Searching and Refactoring Actions on page 666
Search and Refactor Operations Scope on page 480

Configuring a Custom Datatype Library for a RELAX NG Schema

A RELAX NG schema can declare a custom datatype library for the values of elements found in XML document instances. The datatype library must be developed in Java and it must implement the interface *specified on the www.thaiopensource.com website.*

The *JAR* file containing the custom library and any other dependent *JAR* file must be added to the classpath of the application, that is the *JAR* files must be added to the folder `[OXYGEN_INSTALL_DIR]/lib` .

To load the custom library, restart Oxygen XML Editor.

Editing NVDL Schemas

Some complex XML documents are composed by combining elements and attributes from namespaces. Furthermore, the schemas that define these namespaces are not even developed in the same schema language. In such cases, it is difficult to specify in the document all the schemas that must be taken into account for validation of the XML document or for content completion. An NVDL (Namespace Validation Definition Language) schema can be used. This schema allows the application to combine and interleave multiple schemas of different types (W3C XML Schema, RELAX NG schema, Schematron schema) in the same XML document.

Oxygen XML Editor offers support for editing NVDL schema files in the following editing modes:

- ***Text editing mode*** - Allows you to edit NVDL schema files in a source editing mode, along with a schema design pane with two tabs that offer a ***Full Model View*** and ***Logical Model View***.
- ***Grid editing mode*** - Displays NVDL schema files in a structured spreadsheet-like grid.
- **Author editing mode** - The visual **Author** mode is also available for Relax NG schema files, presenting them in a compact and easy to understand representation.

For information about applying and detecting schemas, see *Associating a Schema to XML Documents* on page 464.

Related Information:

Associating a Schema to XML Documents on page 464

NVDL Schema Diagram

This section explains how to use the graphical diagram of a NVDL schema.

Introduction to NVDL Schema Diagram Editor

Oxygen XML Editor provides a simple, expressive, and easy-to-read schema diagram editor for NVDL schemas.

With this new feature you can easily develop complex schemas, print them on multiple pages or save them as JPEG, PNG, and BMP images. It helps both schema authors in developing the schema and content authors that are using the schema to understand it.

Oxygen XML Editor is the only XML Editor to provide a side by side source and diagram presentation and have them real-time synchronized:

- The changes you make in the Editor are immediately visible in the Diagram (no background parsing).
- Changing the selected element in the diagram, selects the underlying code in the source editor.

Full Model View

When you create a schema document or open an existing one, the editor panel is divided in two sections: one containing the schema diagram and the second the source code. The diagram view has two tabbed panes offering a **Full Model View** and a ***Logical Model View***.

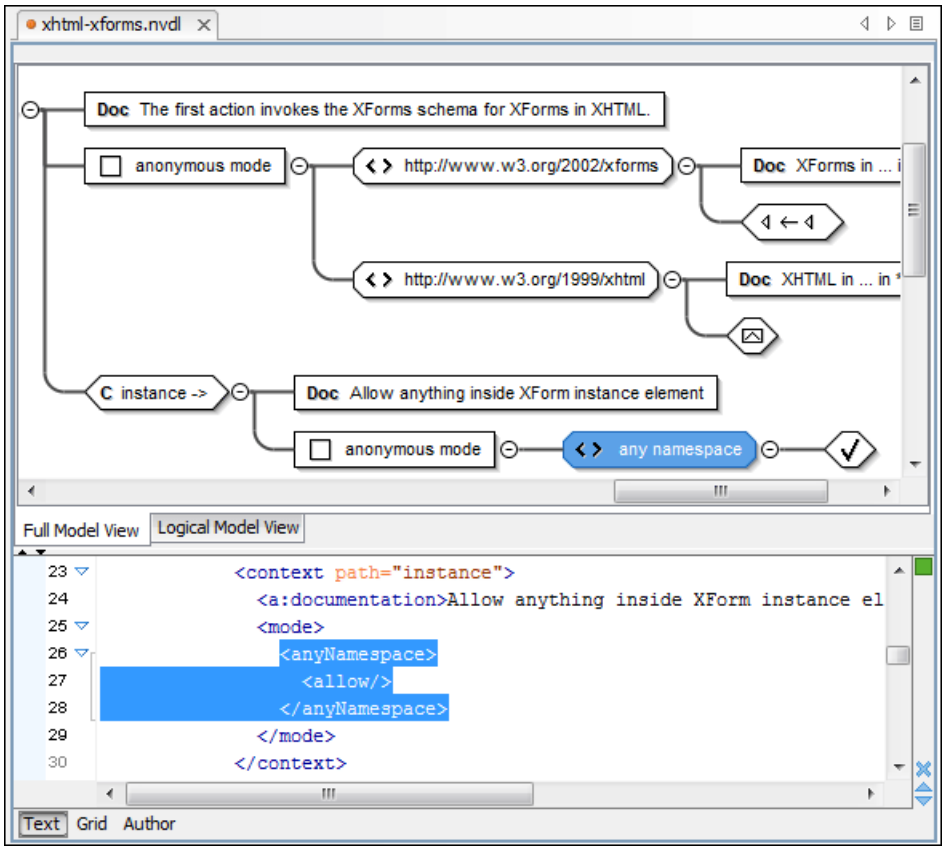


Figure 407: NVDL Schema Editor - Full Model View

The **Full Model View** renders all the NVDL elements with intuitive icons. This representation coupled with the synchronization support makes the schema navigation easy.

Double-click any diagram component to edit its properties.

Logical Model View

The **Logical Model View** presents the compiled schema in the form of a single pattern. The patterns that form the element content are defined as top level patterns with generated names. These names are generated depending of the elements name class.

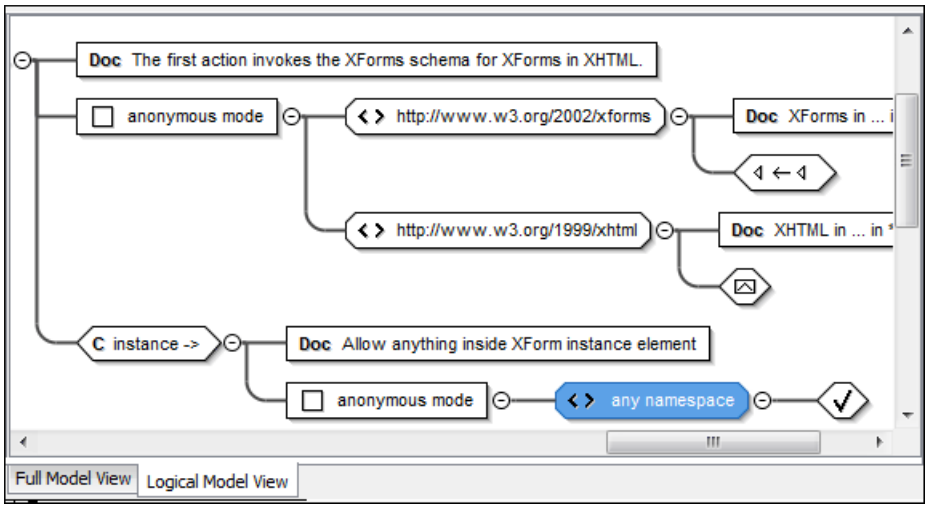


Figure 408: Logical Model View for an NVDL Schema

Actions Available in the Diagram Editor

The contextual menu offers the following actions:

Show only the selected component

Depending on its state (selected/not selected), either the selected component or all the diagram components are shown.

Show Annotations

Depending on its state (selected/not selected), the documentation nodes are shown or hidden.

Auto expand to references

This option controls how the schema diagram is automatically expanded. For instance, if you select it and then edit a top-level element or you trigger a diagram refresh, the diagram will be expanded until it reaches the referenced components. If this option is left unchecked, only the first level of the diagram is expanded, showing the top-level elements. For large schemas, the editor disables this option automatically.

Collapse Children

Collapses the children of the selected view.

Expand Children

Expands the children of the selected view.

Print Selection

Prints the selected view.

Save as Image

Saves the current selection as image, in JPEG, BMP, SVG or PNG format.

Refresh

Refreshes the schema diagram according to the changes in your code (changes in your imported documents or those that are not reflected automatically in the compiled schema).

If the schema is not valid, you see only an error message in the ***Logical Model View*** instead of the diagram.

Validating NVDL Schema Documents

By default, NVDL schema files are validated as you type. To change this, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Editor** > **Document Checking**, and deselect the ***Enable automatic validation** option*.

To validate an NVDL schema document manually, select the **Validate** action from the **Validation** toolbar drop-down menu or the **Document** > **Validate** menu. When Oxygen XML Editor validates an NVDL schema file, it expands all the included modules so the entire schema hierarchy is validated. The validation problems are highlighted directly in the editor, making it easy to locate and fix any issues.

Related Information:

Validating XML Documents Against a Schema on page 446

Content Completion in NVDL Schemas

The intelligent *Content Completion Assistant* allows you to quickly identify and insert elements, attributes, and attribute values that are valid in the current editing context. All available proposals are listed in a pop-up menu displayed at the current cursor position.

The *Content Completion Assistant* is enabled by default. To disable it, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Editor** > **Content Completion**, and deselect the ***Enable content completion** option*.

When active, the *Content Completion Assistant* displays a list of context-sensitive proposals valid at the current cursor position. It can be manually activated with the **Ctrl + Space (Command + Space on OS X)** shortcut. You can navigate through the list of proposals by using the **Up** and **Down** keys on your keyboard. For each selected item in the list, the *Content Completion Assistant* displays a documentation window. You can customize the size of the documentation window by dragging its top, right, and bottom borders.

To insert the selected content in **Text** mode, do one of the following:

- Press **Enter** or **Tab** to insert both the start and end tags and position the cursor inside the start tag in a position suitable for inserting attributes.
- Press **Ctrl + Enter (Command + Enter on OS X)** to insert both the start and end tags and positions the cursor between the tags in a position where you can start typing content.

If you are using the concept of *master files* to import/include modules, the *Content Completion Assistant* collects its components starting from the *master files*. The *master files* can be defined in the project or in the associated validation scenario. For more information about the *Master Files* support in Oxygen XML Editor, see *Defining Master Files at Project Level*.

The *Content Completion Assistant* also offers additional information for the element and attribute proposals in the form of schema annotations that is displayed in a tooltip.

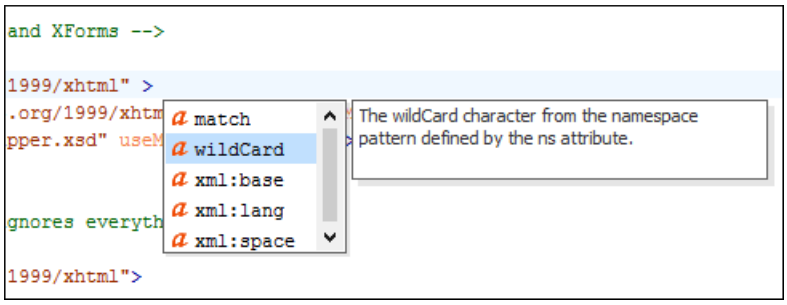


Figure 409: NVDL Content Completion Assistant

Syntax Highlighting in NVDL Schemas

Oxygen XML Editor supports syntax highlighting in NVDL schemas to improve the readability of the content and reduce the time it takes to internalize the semantics of the structured content.

To customize the colors or styles used for the syntax highlighting colors for NVDL schemas, follow these steps:

1. *Open the **Preferences** dialog box (**Options** > **Preferences**)*.
2. Go to ***Editor** > **Syntax Highlight***.
3. Select and expand the **XML** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the **XML** tab in the **Preview** pane to see the effects of your changes.

Tip: Oxygen XML Editor also allows you to specify syntax highlighting colors for specific XML elements and attributes with specific namespace prefixes. This can be done in the ***Editor** > **Syntax Highlight** > **Elements/ Attributes by Prefix** preferences page*.

Related Information:

Syntax Highlight Preferences on page 104

NVDL Outline View

The **Outline** view for NVDL schemas presents a list with the named or anonymous rules that appear in the diagram and it allows for quick access to a rule by name. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Component Dependencies View for NVDL Schemas

The **Component Dependencies** view allows you to see the dependencies for a selected NVDL named mode. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

If you want to see the dependencies of an NVDL mode, select the desired component in the editor and choose the **Component Dependencies** action from the contextual menu. The action is available for all named modes.

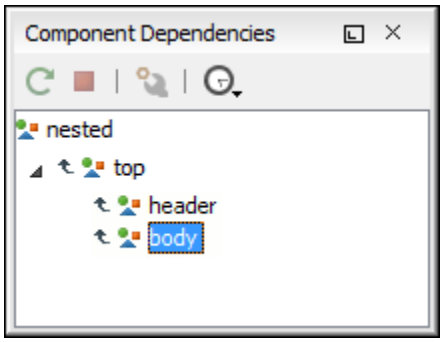


Figure 410: Component Dependencies View - Hierarchy for `test.nvdl`

In the **Component Dependencies** the following actions are available on the toolbar:

Refresh

Refreshes the dependencies structure.

Stop

Allows you to stop the dependencies computing.

Configure

Allows you to configure a search scope to compute the dependencies structure. If you decide to set the application to use automatically the defined scope for future operations, select the corresponding checkbox.

History

Repeats a previous dependencies computation.

The following actions are available in the contextual menu:

Go to First Reference

Selects the first reference of the referenced component from the current selected component in the dependencies tree.

Go to Component

Shows the definition of the current selected component in the dependencies tree.

Tip: If a component contains multiple references to another component, a small table containing all references is displayed. When a recursive reference is encountered it is marked with a special icon .

Searching and Refactoring Actions in NVDL Schemas

Search Actions

The following search actions can be applied on `name`, `useMode`, and `startMode` attributes and are available from the **Search** submenu in the contextual menu of the current editor or from the **Document** > **References** menu:

- **Search References** - Searches all references of the item found at current cursor position in the defined scope, if any. If a scope is defined, but the current edited resource is not part of the range of resources determined by this, a warning dialog box is displayed and you have the possibility to define another search scope.
- **Search References in** - Searches all references of the item found at current cursor position in the file or files that you specify when define a scope in the **Search References** dialog box.
- **Search Declarations** - Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined, but the current edited resource is not part of the range of resources determined by this, a warning dialog box will be displayed and you have the possibility to define another search scope.
- **Search Declarations in** - Searches all declarations of the item found at current cursor position in the file or files that you specify when you define a scope for the search operation.

- **Search Occurrences in File** - Searches all occurrences of the item at the cursor position in the currently edited file.

The following action is available from the contextual menu and the **Document** > **Schema** menu:

- **Show Definition** - Moves the cursor to its definition in the schema used by the NVDL to validate it.

  Note: You can also use the **Ctrl + Single-Click (Command + Single-Click on OS X)** shortcut on a reference to display its definition.

Refactoring Actions

The following refactoring actions can be applied on `name`, `useMode`, and `startMode` attributes and are available from the **Refactoring** submenu in the contextual menu of the current editor or from the **Document** > **Refactoring** menu:

- **Rename Component** - Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the **Esc** or **Enter** key on your keyboard.
- **Rename Component in** - Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the **Preview** button, you can view the files to be affected by the action.

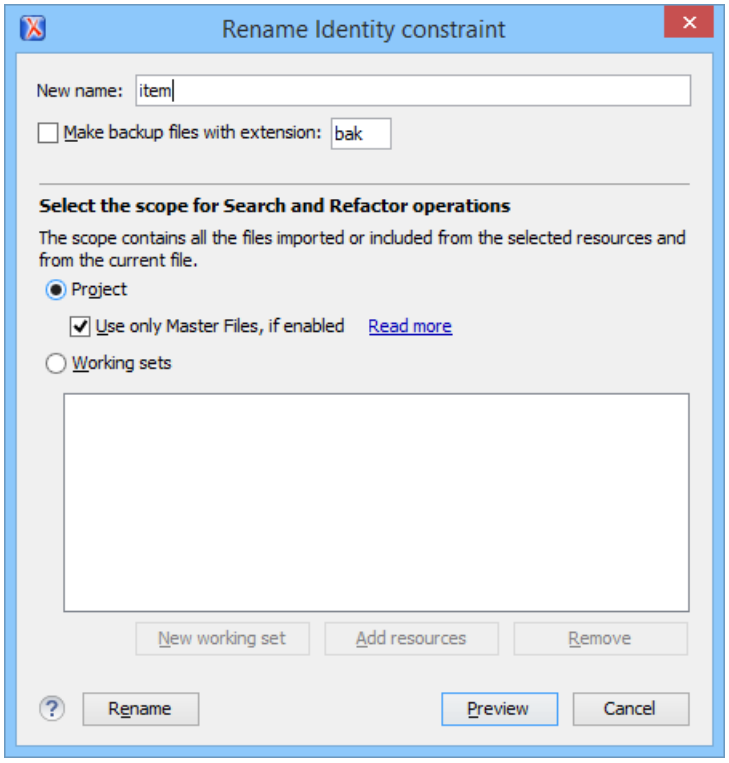


Figure 411: Rename Identity Constraint Dialog Box

Editing JSON Documents

This section explains the features of the Oxygen XML Editor JSON Editor and how to use them.

Editing JSON Documents in Text Mode

When editing JSON documents in the **Text** editing mode, the *usual text editing actions* are available, along with other editor specific actions, including:

- *Find / Replace*
- *Drag and Drop*
- *Validation*
- *Format and Indent (Pretty Print)*

Note: You can run XPath expressions on opened JSON documents, but in **Text** mode the XPath results cannot be mapped in the document. However, they can be mapped in the **Grid** editing mode. You can use the **Grid** button at the bottom of the editor panel to switch to that editing mode.

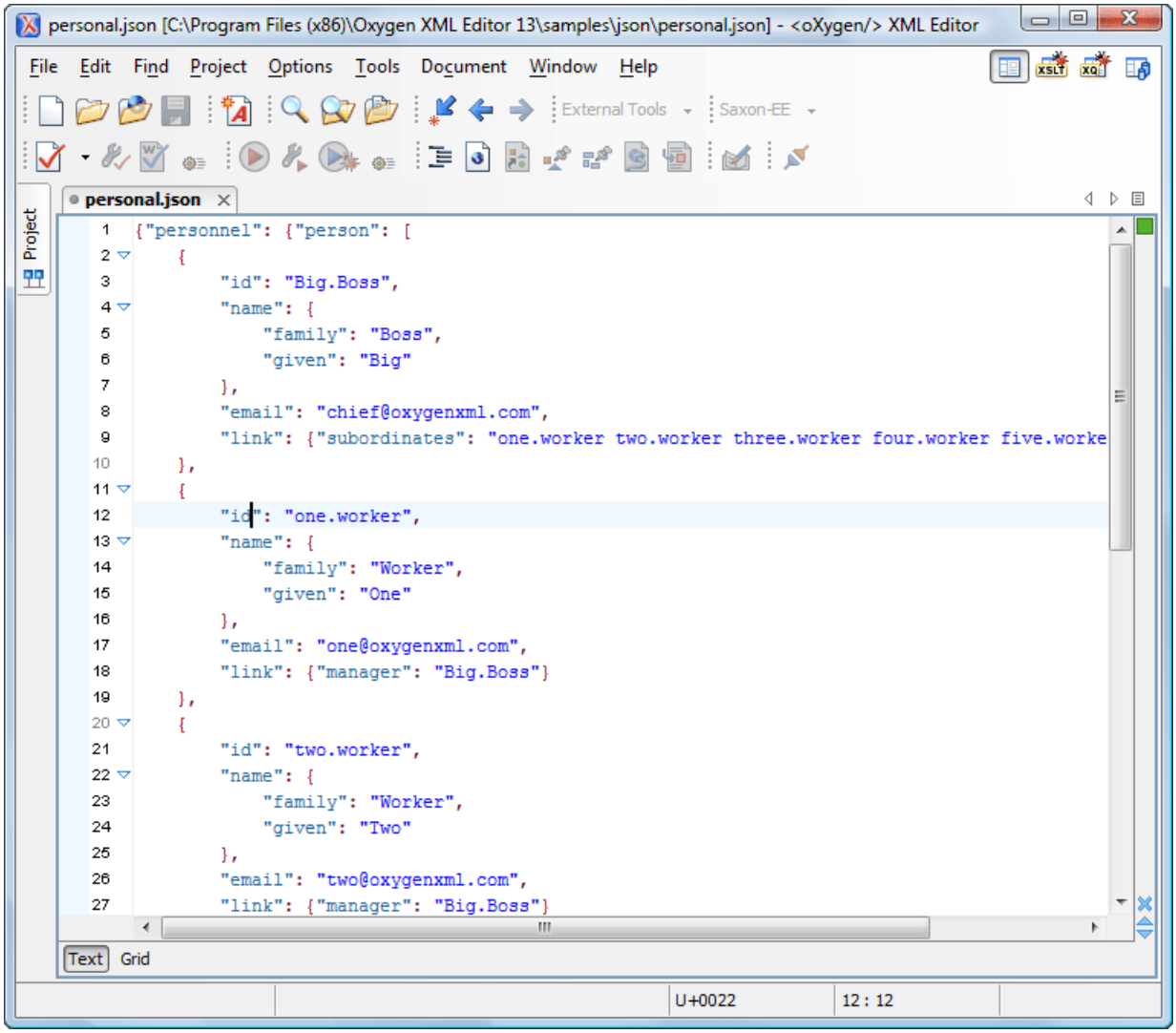


Figure 412: JSON Editor Text Mode

Related Information:

Text Preferences on page 78
Editing JSON Documents in Grid Mode on page 676

Editing JSON Documents in Grid Mode

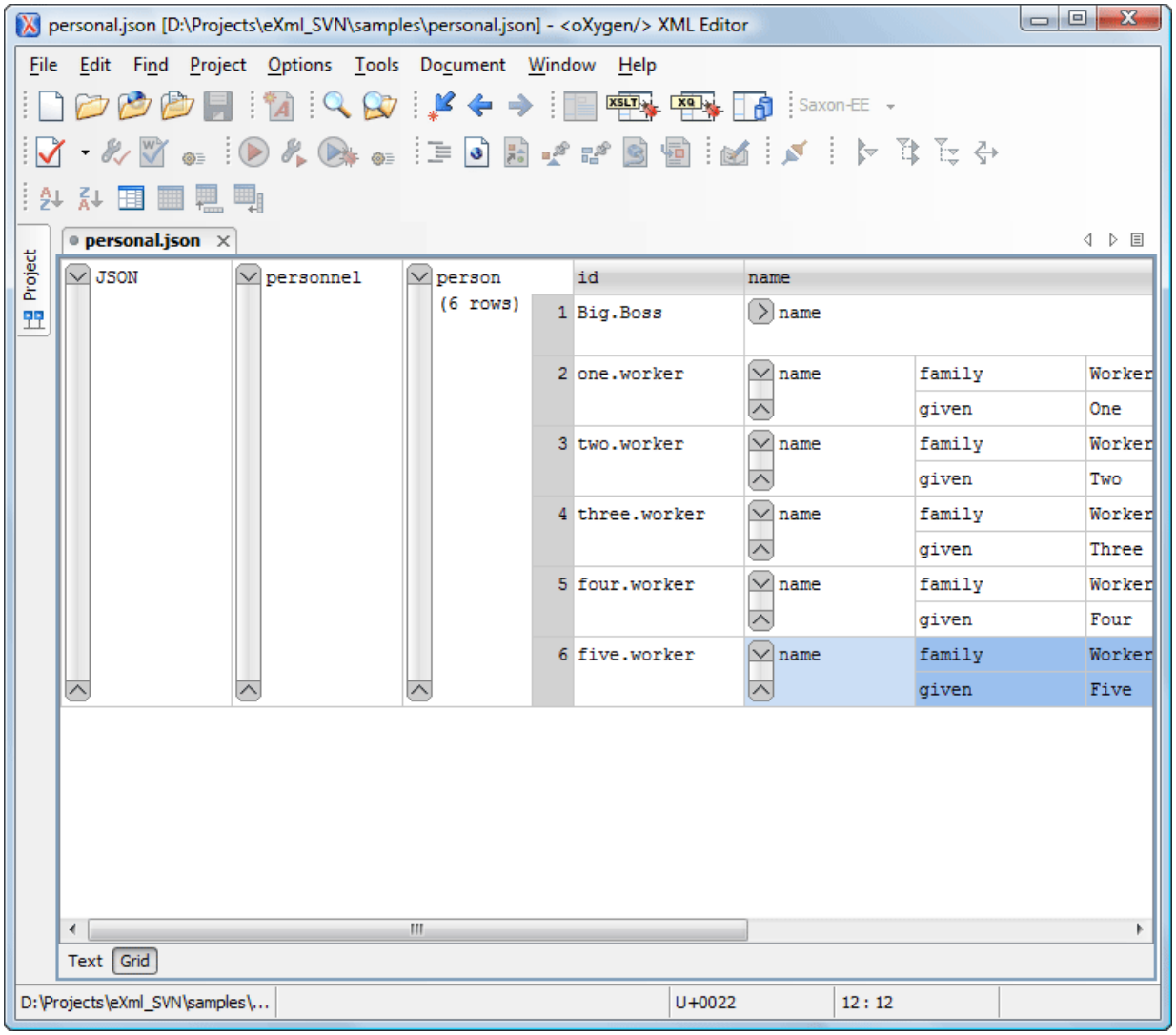


Figure 413: JSON Editor Grid Mode

Oxygen XML Editor allows you to view and edit the JSON documents in the ***Grid*** *mode*. The JSON is represented in **Grid** mode as a compound layout of nested tables in which the JSON data and structure can be easily manipulated with table-specific operations or drag and drop operations on the grid components. You can also use the following JSON-specific contextual actions:

Array

Useful when you want to convert a JSON *value* to *array*.

Insert value before

Inserts a value before the currently selected one.

Insert value after

Inserts a value after the currently selected one.

Append value as child

Appends a value as a child of the currently selected value.

You can *customize the JSON grid appearance* according to your needs. For instance, you can change the font, the cell background, foreground, or even the colors from the table header gradients. The default width of the columns can also be changed.

Related Information:

Grid Editing Mode on page 208

Grid Preferences on page 79

Validating JSON Documents

Oxygen XML Editor includes a built-in JSON validator (based on the free JAVA source code available on www.json.org), integrated with the general validation support.

Related Information:

Presenting Validation Errors in Text Mode on page 206
Document Checking Preferences on page 95

Syntax Highlighting in JSON Documents

Oxygen XML Editor supports syntax highlighting in JSON files to improve the readability of the content and reduce the time it takes to internalize the semantics of the structured content.

To customize the colors or styles used for the syntax highlighting colors for JSON files, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)***.
2. Go to ***Editor*** > ***Syntax Highlight***.
3. Select and expand the **JSON** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

Related Information:

Syntax Highlight Preferences on page 104

Folding in JSON

In a large JSON document, the data enclosed in the '{' and '}' characters can be collapsed so that only the needed data remain in focus. The *folding features available for XML documents* are available in JSON documents.

JSON Outline View

The **Outline** view for JSON documents displays the list of all the components of the JSON document you are editing. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

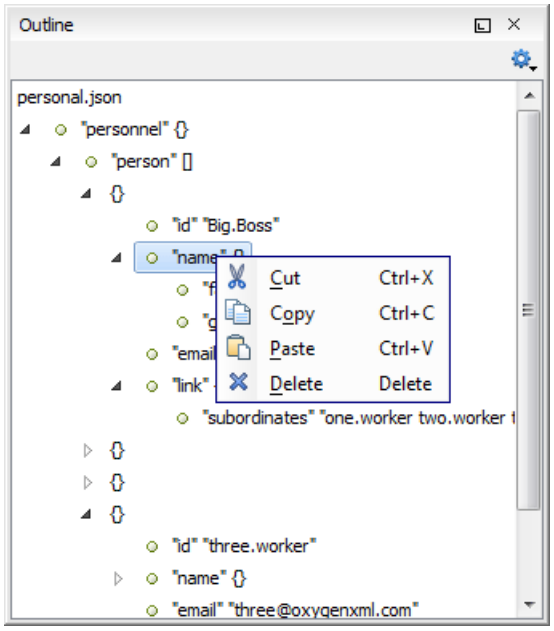


Figure 414: JSON Outline View

The following actions are available in the contextual menu of the JSON **Outline** view:

- **Cut**
- **Copy**
- **Paste**
- **Delete**

The **Settings** menu on the JSON **Outline** view toolbar includes a **Selection update on cursor move** option that allows you to control the synchronization between the **Outline** view and source the document. Oxygen XML Editor synchronizes the selection in the **Outline** view with the cursor moves or the changes you make in the JSON editor. Selecting one of the components from the **Outline** view also selects the corresponding item in the source document.

XML to JSON Converter

Oxygen XML Editor includes a useful and simple tool for converting XML files to JSON. It can be found in the **Tools** menu.

To convert an XML document to JSON, follow these steps:

1. Select the **XML to JSON** action from the **Tools** menu.
   The **XML to JSON** dialog box is displayed:

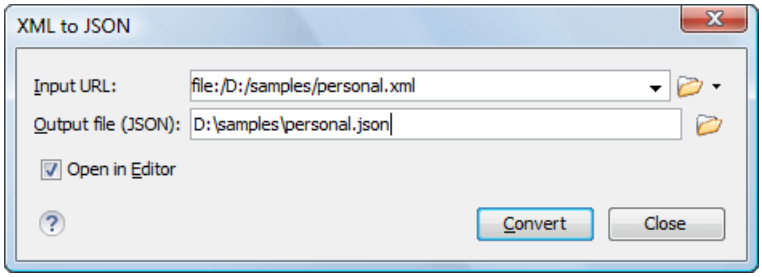


   Figure 415: XML to JSON Dialog Box
2. Choose or enter the **Input URL** of the XML document.
3. Choose the path of the **Output file** that will contain the conversion JSON result.
4. Select the **Open in Editor** option to open the JSON result of the conversion in the Oxygen XML Editor JSON Editor.
5. Click the **Convert** button.

The original XML document is now converted to a JSON document.

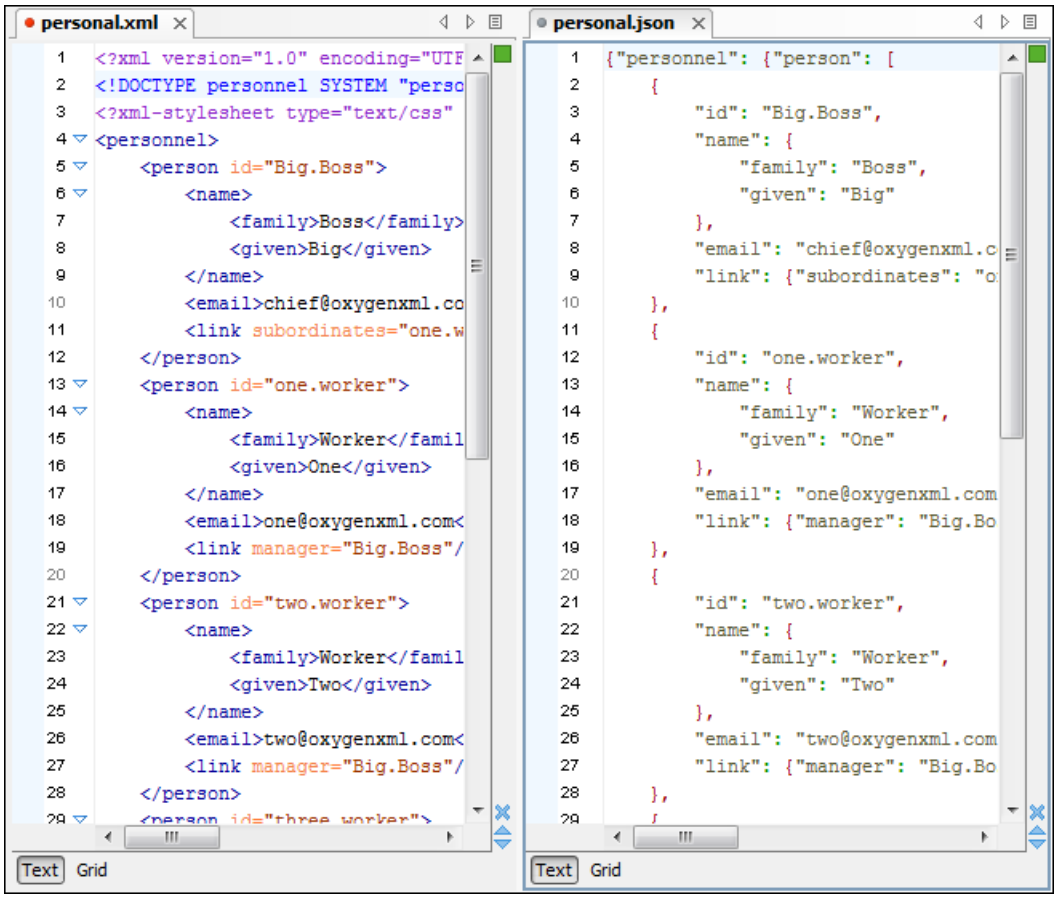


Figure 416: Example: XML to JSON Operation Result

Editing StratML Documents

Strategy Markup Language (StratML) is an XML vocabulary and schema for strategic plans. Oxygen XML Editor supports StratML Part 1 (Strategic Plan) and StratML Part 2 (Performance Plans and Reports) and provides templates for the following documents:

- **Strategic Plan** (StratML Part 1)
- **Performance Plan** (StratML Part 2)
- **Performance Report** - (StratML Part 2)
- **Strategic Plan** - (StratML Part 2)

You can view the components of a StratML document in the ***Outline*** *view*. Oxygen XML Editor implements a default XML with XSLT transformation scenario for this document type, called StratML to HTML.

Editing XLIFF Documents

XLIFF (*XML Localization Interchange File Format*) is an XML-based format that was designed to standardize the way multilingual data is passed between tools during a localization process. Oxygen XML Editor provides the following support for editing XLIFF documents:

XLIFF Version 1.2 and 2.0 Support:

- New file templates for XLIFF documents.
- A default CSS file (`xliff.css`) used for rendering XLIFF content in **Author** mode is stored in *`[OXYGEN_INSTALL_DIR]`*`/frameworks/xliff/css/`.

- Validation and content completion support using local catalogs. The default catalog (`catalog.xml`) for version 1.2 is stored in *[OXYGEN_INSTALL_DIR]*`/frameworks/xliff/xsd/1.2`, and for version 2.0 in *[OXYGEN_INSTALL_DIR]*`/frameworks/xliff/xsd/2.0`.

XLIFF Version 2.0 Enhanced Support:

- Support for validating XLIFF 2.0 documents using modules. The default modules are stored in *[OXYGEN_INSTALL_DIR]*`/frameworks/xliff/xsd/2.0/modules`.

Editing JavaScript Documents

This section explains the features of the Oxygen XML Editor JavaScript Editor and how you can use them.

JavaScript Editing Actions

Oxygen XML Editor allows you to create and edit JavaScript files and assists you with useful features such as syntax highlight, content completion, and outline view. To enhance your editing experience, you can select entire blocks (parts of text delimited by brackets) by double-clicking somewhere inside the brackets.

```
90    function change_sides(front) {
91      switch ($('#version-switch').text()) {
92        case 'Original':
93          $('#holder').html($('div .original[id]').html());
94          make_clickable();
95          $('#version-switch').text('Translation 1');
96          break;
97        case 'Translation 1':
98          $('#holder').html($('div .translation[id]').filter(':first').html());
99          $('#version-switch').text('Translation 2');
100         break;
101       case 'Translation 2':
102         $('#holder').html($('div .translation[id]').filter(':last').html());
103         $('#version-switch').text('Original');
104         break;
105       }
106     }
```

Figure 417: JavaScript Editor Text Mode

The contextual menu of the **JavaScript** editor offers the following actions:

Cut

Allows you to cut fragments of text from the editing area.

Copy

Allows you to copy fragments of text from the editing area.

Paste

Allows you to paste fragments of text in the editing area.

Toggle Comment

Allows you to comment a line or a fragment of the JavaScript document you are editing. This option inserts a single comment for the entire fragment you want to comment.

Toggle Line Comment

Allows you to comment a line or a fragment of the JavaScript document you are editing. This option inserts a comment for each line of the fragment you want to comment.

Go to Matching Bracket

Use this option to find the closing, or opening bracket, matching the bracket at the cursor position. When you select this option, Oxygen XML Editor moves the cursor to the matching bracket, highlights its row, and decorates the initial bracket with a rectangle.

Note: A rectangle decorates the opening or closing bracket that matches the current one, at all times.

Source

Allows you to select one of the following actions:

To Lower Case

Converts the selection content to lower case characters.

To Upper Case

Converts the selection content to upper case characters.

Capitalize Lines

Converts to upper case the first character of every selected line.

Join and Normalize Lines

Joins all the rows you select to one row and normalizes the content.

Insert new line after

Inserts a new line after the line at the cursor position.

Modify all matches

Use this option to modify (in-place) all the occurrences of the selected content. When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Open

Allows you to select one of the following actions:

- **Open File at Cursor** - select this action to open the source of the file located at the cursor position
- **Open File at Cursor in System Application** - select this action to open the source of the file located at the cursor position with the application that the system associates with the file

Compare

Select this option to open the **Compare Files** tool to compare the file you are editing with a file you choose in the dialog box.

Folding

When you invoke the contextual menu from the *folding* triangles in the stripe on the left side of the editor, the following actions are available:

Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on OS X))

Folds all the elements except the current element.

Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on OS X))

Folds the elements indented with one level inside the current element.

Expand Child Folds

Unfolds all child elements of the currently selected element.

Expand All (Ctrl + NumPad* (Command + NumPad* on OS X))

Unfolds all elements in the current document.

Validating JavaScript Files

You have the possibility to validate the JavaScript document you are editing. Oxygen XML Editor uses the Mozilla Rhino library for validation. For more information about this library, go to *http://www.mozilla.org/rhino/doc.html*. The JavaScript validation process checks for errors in the syntax. Calling a function that is not defined is not treated as an error by the validation process. The interpreter discovers this error when executing the faulted line. Oxygen XML Editor can validate a JavaScript document both on-request and automatically.

Content Completion in JavaScript Documents

When you edit a JavaScript document, the *Content Completion Assistant* presents you a list of the elements you can insert at the cursor position. It can be manually activated with the **Ctrl + Space (Command + Space on OS X)** shortcut.

For an enhanced assistance, JQuery methods are also presented. The following icons decorate the elements in the content completion list of proposals depending on their type:

- - function
- - variable
- - object
- - property
- - method

Note: These icons decorate both the elements from the content completion list of proposals and from the ***Outline*** *view*.

Figure 418: JavaScript Content Completion Assistant

The *Content Completion Assistant* collects:

- Method names from the current file and from the library files.
- Functions and variables defined in the current file.

If you edit the content of a function, the content completion list of proposals contains all the local variables defined in the current function, or in the functions that contain the current one.

Syntax Highlighting in JavaScript Documents

Oxygen XML Editor supports syntax highlighting in JavaScript files to improve the readability of the content and reduce the time it takes to internalize the semantics of the structured content.

To customize the colors or styles used for the syntax highlighting colors for JavaScript files, follow these steps:

1. *Open the* ***Preferences*** *dialog box* ***(Options*** *>* ***Preferences)****.*
2. Go to ***Editor*** *>* ***Syntax Highlight****.*
3. Select and expand the **JavaScript** section in the top pane.

4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

Related Information:

Syntax Highlight Preferences on page 104

JavaScript Outline View

Oxygen XML Editor present a list of all the components of the JavaScript document you are editing in the **Outline** view. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

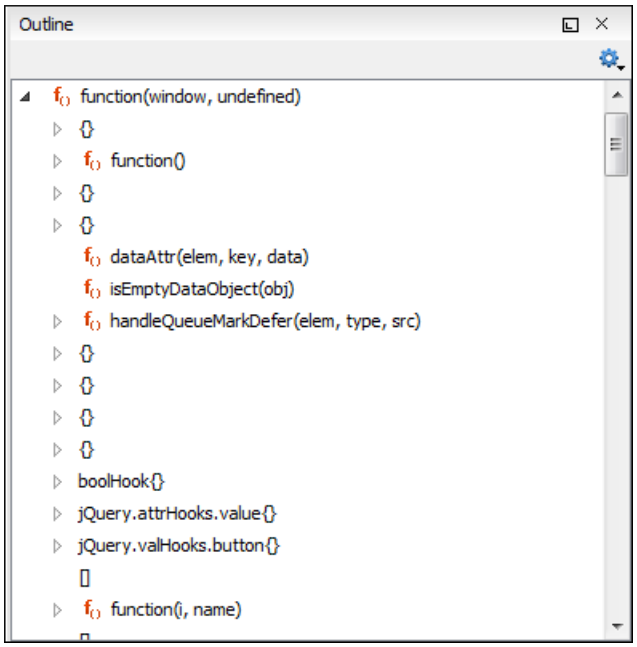


Figure 419: JavaScript Outline View

The following icons decorate the elements in the **Outline** view depending on their type:

- - function
- - variable
- - object
- - property
- - method

The contextual menu of the JavaScript **Outline** view contains the usual **Cut**, **Copy**, **Paste**, and **Delete** actions. From the **Settings** menu, you can select the **Update selection on cursor move** option to synchronize the **Outline** view with the editing area.

Editing XProc Scripts

An XProc script is edited as an XML document that is validated against a RELAX NG schema. If the script has an associated transformation scenario, then the XProc engine from the scenario is invoked as validating engine. The default engine for XProc scenarios is the Calabash engine that comes bundled with Oxygen XML Editor version 19.0.

XProc Content Completion

Oxygen XML Editor helps you edit a XProc scripts through the *Content Completion Assistant*, offering proposals that are valid at the cursor position. It can be manually activated with the **Ctrl + Space (Command + Space on OS X)** shortcut.

The content completion inside the element `input/inline` from the XProc namespace *http://www.w3.org/ns/xproc* offers elements from the following schemas depending both on the `port` attribute and the parent of the `input` element. When invoking the content completion inside the XProc element `inline`, the list of content completion proposals is populated as follows:

- If the value of the `port` attribute is `stylesheet` and the `xslt` element is the parent of the `input` elements, the *Content Completion Assistant* offers XSLT elements.
- If the value of the `port` attribute is `schema` and the `validate-with-relax-ng` element is the parent of the `input` element, the *Content Completion Assistant* offers RELAX NG schema elements.
- If the value of the `port` attribute is `schema` and the `validate-with-xml-schema` element is the parent of the `input` element, the *Content Completion Assistant* offers XML Schema schema elements.
- If the value of the `port` attribute is `schema` and the `validate-with-schematron` element is the parent of the `input` element , the *Content Completion Assistant* offers either ISO Schematron elements or Schematron 1.5 schema elements.
- If the above cases do not apply, then the *Content Completion Assistant* offers elements from all the schemas from the above cases.

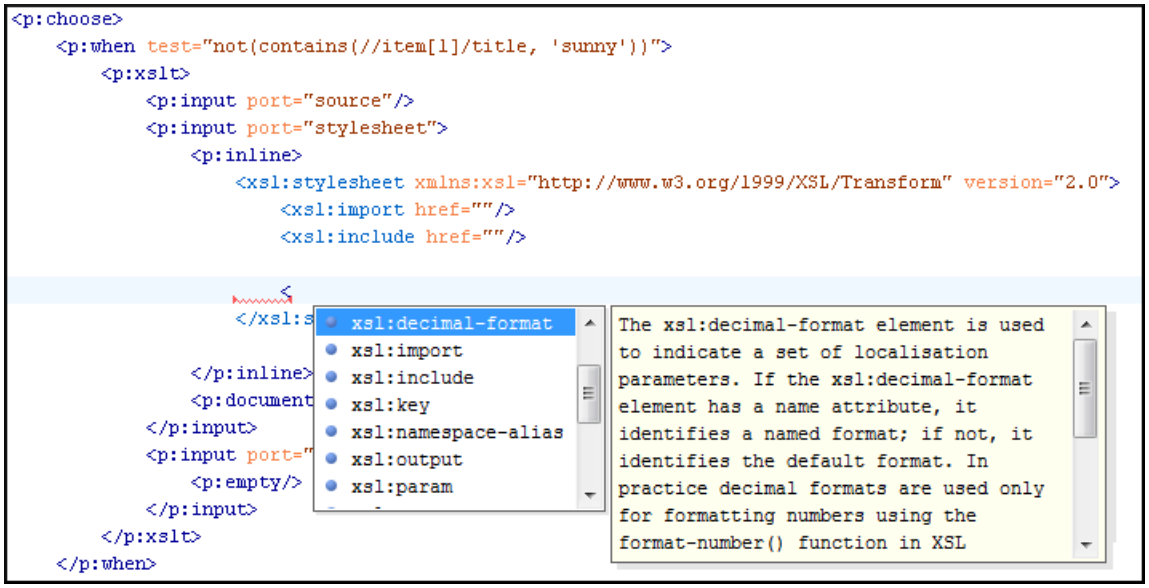


Figure 420: XProc Content Completion

XProc Syntax Highlighting

The XProc editor assists you in writing XPath expressions by offering dedicated coloring schemes for syntax highlighting.

To customize the colors or styles used for the syntax highlighting colors for XProc, follow these steps:

1. *Open the **Preferences** dialog box **(Options > Preferences)***.
2. Go to ***Editor > Syntax Highlight***.
3. Select and expand the **XML** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the **XML** tab in the **Preview** pane to see the effects of your changes.

Editing Schematron Schemas

Schematron is a simple and powerful Structural Schema Language for making assertions about patterns found in XML documents. It relies almost entirely on XPath query patterns for defining rules and checks. Schematron validation rules allow you to specify a meaningful error message. This error message is provided to you if an error is encountered during the validation stage.

Oxygen XML Editor assists you in editing Schematron documents with schema-based content completion, syntax highlight, search and refactor actions, and dedicated icons for the ***Outline*** *view*. You can create a new Schematron schema using one of the Schematron templates available in the ***New*** *document wizard*.

For information about applying and detecting Schematron schemas, see *Associating a Schema to XML Documents* on page 464.

Validating XML Documents Against Schematron

The Skeleton XSLT processor is used for validation and conforms with ISO Schematron or Schematron 1.5. It allows you to *validate XML documents against Schematron schemas* or against combined RELAX NG / W3C XML Schema and Schematron.

How to Specify the Query Language Binding

You can specify the query language binding to be used in the Schematron schema by doing the following:

- For embedded ISO schematron, *open the* ***Preferences*** *dialog box* ***(Options > Preferences)***, go to **XML** > **XML Parser** > **Schematron**, and select it in the ***Embedded rules query language binding*** *option*.
- For standalone ISO Schematron, specify the version by setting the query language to be used in a `queryBinding` attribute on the schema root element. For more information, see the *Query Language Binding section of the Schematron specifications*.
- For Schematron 1.5 (standalone and embedded), *open the* ***Preferences*** *dialog box* ***(Options > Preferences)***, go to **XML** > **XML Parser** > **Schematron**, and select the version in the ***XPath Version*** *option*.

Multi-Lingual Support in Schematron Messages

You can specify the desired language for the validation messages in *the* ***Schematron Preferences*** *page*. The Schematron validation messages can be presented in multiple languages by defining the language for each message using the Schematron `diagnostics`. For more information, see the *Use of Schematron for Multi-Lingual Schemas* specification.

How to Customize Color Schemes in Schematron

The Schematron editor renders the XPath expressions with dedicated coloring schemes . To customize the coloring schemes, *open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Editor** > **Syntax Highlight**.

Schematron Transformation Scenario

When you create a Schematron document, Oxygen XML Editor provides a built-in transformation scenario. You can use this scenario to obtain the XSLT style-sheet corresponding to the Schematron schema. You can apply this XSLT stylesheet to XML documents to obtain the Schematron validation results.

To watch our video demonstration about the Schematron support in Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/Schematron_Validation.html*, and to *https://www.oxygenxml.com/demo/Editing_Schematron_Schemas.html*.

Related Information:

Editing XML Documents in Text Mode on page 290

Associating a Schema to XML Documents on page 464

Editing Schematron Schema in the Master Files Context

Smaller interrelated modules that define a complex Schematron cannot be correctly edited or validated individually, due to their interdependency with other modules. For example, a diagnostic defined in a main schema document is not visible when you edit an included module. Oxygen XML Editor provides the support for defining the main module (or modules), thus allowing you to edit any of the imported/included schema files in the context of the larger schema structure.

You cat set a main Schematron document either using the *master files support from the **Project** view*, or using a validation scenario.

To set a main file using a validation scenario, add validation units that point to the main schemas. Oxygen XML Editor warns you if the current module is not part of the dependencies graph computed for the main schema. In this case, it considers the current module as the main schema.

The advantages of editing in the context of main file include:

- Correct validation of a module in the context of a larger schema structure.
- *Content Completion Assistant* displays all the referable components valid in the current context. This include components defined in modules other than the currently edited one.

Validating Schematron Documents

By default, a Schematron schema is validated as you type. To change this, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Editor** > **Document Checking**, and deselect the ***Enable automatic validation** option*.

To validate a Schematron document manually, select the **Validate** action from the **Validation** toolbar drop-down menu or the **Document** > **Validate** menu. When Oxygen XML Editor validates a Schematron schema, it expands all the included modules so the entire schema hierarchy is validated. The validation problems are highlighted directly in the editor, making it easy to locate and fix any issues.

Oxygen XML Editor offers an error management mechanism capable of pinpointing errors in XPath expressions and in the included schema modules.

Related Information:

Presenting Schematron Validation Issues

Presenting Schematron Validation Issues

The possible issues that might occur during the validation process when validating XML documents against Schematron are presented with colored underlines in the editing pane, colored markers in the right vertical stripe, and details about the issues are presented in the **Errors** panel at the bottom area of the Oxygen XML Editor window. Each error is flagged with a severity level that can be: *warning*, *error*, *fatal* or *info*.

To set a severity level, Oxygen XML Editor looks for the following information:

- The `role` attribute, which can have one of the following values:
  - `warn` or `warning` - Sets the severity level to *warning*. By default, underlined with a yellow squiggly line in the editing pane and a yellow marker in the right vertical stripe.
  - `error` - Sets the severity level to *error*. By default, underlined with a red squiggly line in the editing pane and a red marker in the right vertical stripe.
  - `fatal` - Sets the severity level to *fatal*. By default, underlined with a red squiggly line in the editing pane and a red marker in the right vertical stripe.
  - `info` or `information` - Sets the severity level to *info*. By default, underlined with a blue squiggly line in the editing pane and a blue marker in the right vertical stripe.
- The start of the message, after trimming leading white-spaces. Oxygen XML Editor looks to match the following exact string of characters (case sensitive):
  - `Warning:` - Sets the severity level to *warning*. By default, underlined with a yellow squiggly line in the editing pane and a yellow marker in the right vertical stripe.
  - `Error:` - Sets the severity level to *error*. By default, underlined with a red squiggly line in the editing pane and a red marker in the right vertical stripe.

- `Fatal:` - Sets the severity level to *fatal*. By default, underlined with a red squiggly line in the editing pane and a red marker in the right vertical stripe.
- `Info:` - Sets the severity level to *info*. By default, underlined with a blue squiggly line in the editing pane and a blue marker in the right vertical stripe.

- If none of the previous rules match, Oxygen XML Editor sets the severity level to *error*. By default, underlined with a red squiggly line in the editing pane and a red marker in the right vertical stripe.

Tip: You can configure the color for each type in the ***Document Checking*** *preferences page*.

Related Information:

Validating XML Documents Against a Schema on page 446
Validation Scenario on page 452
Associating a Schema to XML Documents on page 464
Presenting Validation Errors in Author Mode on page 236
Presenting Validation Errors in Text Mode on page 206

Content Completion in Schematron Documents

Oxygen XML Editor helps you edit a Schematron schema through the *Content Completion Assistant*, offering proposals that are valid at the cursor position. It can be manually activated with the **Ctrl + Space (Command + Space on OS X)** shortcut.

When you edit the value of an attribute that refers a component, the proposed components are collected from the entire schema hierarchy. For example, if the editing context is `phase/active/@pattern`, the *Content Completion Assistant* proposes all the defined patterns.

Note: For Schematron resources, the *Content Completion Assistant* collects its components starting from the *master files*. The *master files* can be defined in the project or in the associated validation scenario. For further details about the *Master Files* support go to *Defining Master Files at Project Level*.

If the editing context is an attribute value that is an XPath expression (such as `assert/@test` or `report/@test`), the *Content Completion Assistant* offers the names of XPath functions, the XPath axes, and user-defined variables.

The *Content Completion Assistant* displays XSLT 1.0 functions and optionally XSLT 2.0 / 3.0 functions in the attributes *path*, *select*, *context*, *subject*, *test* depending on *the Schematron options* that are set in Preferences pages. If the Saxon 6.5.5 namespace (`xmlns:saxon="http://icl.com/saxon"`) or the Saxon 9.7.0.15 namespace is declared in the Schematron schema (`xmlns:saxon="http://saxon.sf.net/"`) the content completion also displays the XSLT Saxon extension functions as in the following figure:

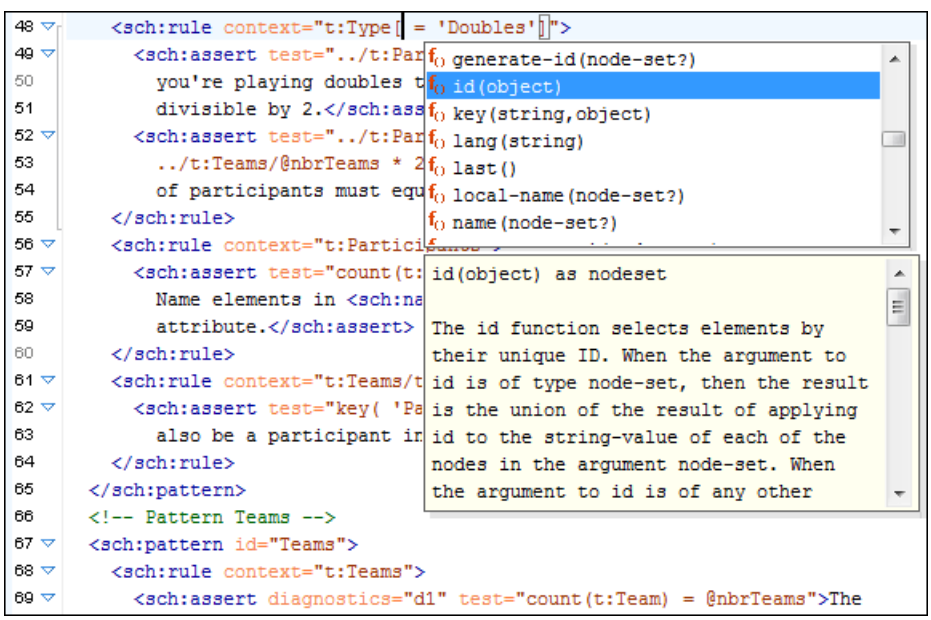


Figure 421: XSLT Extension Functions in Schematron Schema Content Completion

The *Content Completion Assistant* also includes *code templates that can be used to quickly insert code fragments* into Schematron documents.

Syntax Highlighting in Schematron

Oxygen XML Editor supports syntax highlighting in Schematron schemas to improve the readability of the content and reduce the time it takes to internalize the semantics of the structured content.

To customize the colors or styles used for the syntax highlighting colors for Schematron schemas, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)***.
2. Go to ***Editor*** > ***Syntax Highlight***.
3. Select and expand the **XML** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the **XML** tab in the **Preview** pane to see the effects of your changes.

Tip: Oxygen XML Editor also allows you to specify syntax highlighting colors for specific XML elements and attributes with specific namespace prefixes. This can be done in the ***Editor*** > ***Syntax Highlight*** > ***Elements/ Attributes by Prefix*** *preferences page*.

Related Information:

Syntax Highlight Preferences on page 104

XML Schema or RELAX NG with Embedded Schematron Rules

Schematron rules can be embedded into an XML Schema through annotations (using the `appinfo` element), or in any element on any level of a RELAX NG Schema (taking into account that the RELAX NG validator ignores all elements that are not in the RELAX NG namespace).

Oxygen XML Editor accepts such documents as Schematron validation schemas and it is able to extract and use the embedded rules.

Validating XML Documents with Relax NG and Embedded Schematron

To validate an XML document with both RELAX NG schema and its embedded Schematron rules, you need to associate the document with both schemas. For example:

```
<?xml-model href="percent.rng" type="application/xml"
            schematypens="http://relaxng.org/ns/structure/1.0"?>
<?xml-model href="percent.rng" type="application/xml"
            schematypens="http://purl.oclc.org/dsdl/schematron"?>
```

The second association validates your document with Schematron rules extracted from the RELAX NG Schema.

Validating XML Documents with XML Schema and Embedded Schematron

Similarly, you can specify an XML Schema having the embedded Schematron rules.

```
<?xml-model href="percent.xsd" type="application/xml"
            schematypens="http://purl.oclc.org/dsdl/schematron"?>
```

Note: When you work with XML Schema or Relax NG documents that have embedded Schematron rules Oxygen XML Editor provides two built-in validation scenarios: **Validate XML Schema with embedded Schematron** for XML schema , and **Validate Relax NG with embedded Schematron** for Relax NG. You can use one of these scenarios to validate the embedded Schematron rules.

Example: Embedded Schematron in Relax NG Schema

```
<sch:pattern>
  <sch:rule context="...">
    <sch:assert test="...">Message.</sch:assert>
  </sch:rule>
</sch:pattern>
```

Example: Embedded Schematron in XML Schema

```
<xsd:appinfo>
  <sch:pattern>
    <sch:rule context="...">
 <sch:assert test="...">Message.</sch:assert>
    </sch:rule>
  </sch:pattern>
</xsd:appinfo>
```

Related Information:

Embed Schematron Quick Fixes in Relax NG or XML Schema on page 702

Schematron Outline View

The **Outline** view for Schematron schemas presents a list of components in a tree-like structure and it allows for quick access to a component by name. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

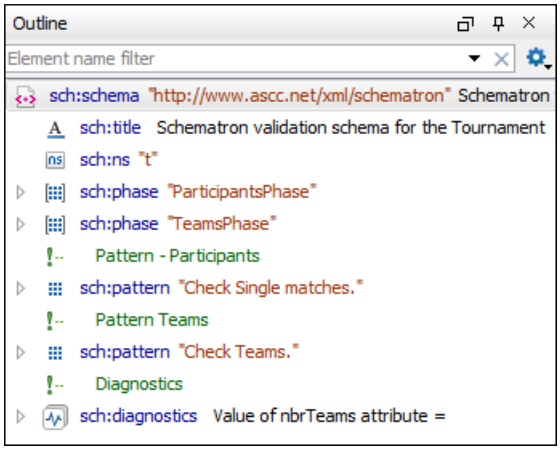


Figure 422: Schematron Outline View

The following actions are available in the **Settings** menu on the **Outline** view toolbar:

Filter returns exact matches

The text filter of the **Outline** view returns only exact matches.

Selection update on cursor move

Controls the synchronization between **Outline** view and source document. The selection in the **Outline** view can be synchronized with the cursor moves or the changes in the editor. Selecting one of the components from the **Outline** view also selects the corresponding item in the source document.

Flat presentation mode of the filtered results

When active, the application flattens the filtered result elements to a single level.

Show comments and processing instructions

Show/hide comments and processing instructions in the **Outline** view.

Show element name

Show/hide element name.

Show text

Show/hide additional text content for the displayed elements.

Show attributes

Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from *the Outline preferences panel*.

Configure displayed attributes

Displays the *XML Structured Outline preferences page*.

The following contextual menu actions are also available in the **Outline** view:

Append Child

Displays a list of elements that you can insert as children of the current element.

Insert Before

Displays a list of elements that you can insert as siblings of the current element, before the current element.

Insert After

Displays a list of elements that you can insert as siblings of the current element, after the current element.

Edit Attributes

Opens a dialog box that allows you to edit the attributes of the currently selected component.

Toggle Comment

Comments/uncomments the currently selected element.

Cut

Cuts the currently selected component.

Copy

Copies the currently selected component.

Delete

Deletes the currently selected component.

Expand More

Expands the structure of a component in the **Outline** view.

Collapse All

Collapses the structure of all the component in the **Outline** view.

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (*, ?) and separate multiple patterns with commas.

Schematron Resource Hierarchy/Dependencies View

The **Resource Hierarchy/Dependencies** view allows you to see the hierarchy/dependencies for a Schematron schema. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

If you want to see the hierarchy of a schema, select the desired schema in the ***Project*** *view* and choose **Resource Hierarchy** from the contextual menu.

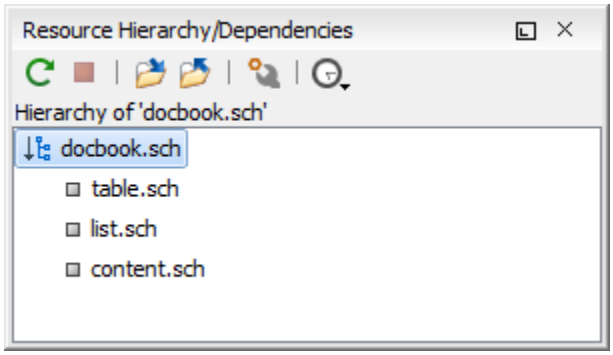


Figure 423: Resource Hierarchy/Dependencies View

If you want to see the dependencies of a schema, select the desired schema in the ***Project*** *view* and choose **Resource Dependencies** from the contextual menu.

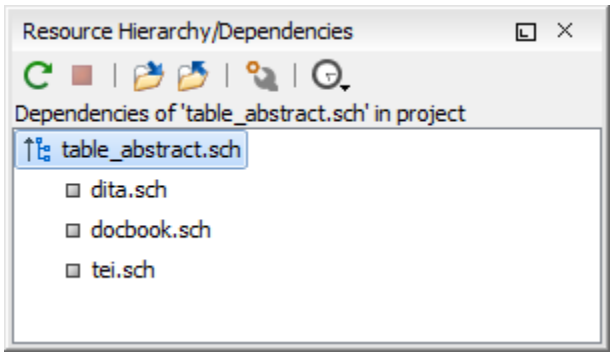


Figure 424: Resource Hierarchy/Dependencies View - Dependencies for `table_abstract.sch`

The following actions are available in the **Resource Hierarchy/Dependencies** view:

Refresh

Refreshes the Hierarchy/Dependencies structure.

Stop

Stops the hierarchy/dependencies computing.

Show Hierarchy

Allows you to choose a resource to compute the hierarchy structure.

Show Dependencies

Allows you to choose a resource to compute the dependencies structure.

Configure

Allows you to configure a scope to compute the dependencies structure. There is also an option for automatically using the defined scope for future operations.

History

Provides access to the list of previously computed dependencies. Use the **Clear history** button to remove all items from this list.

The contextual menu contains the following actions:

Open

Opens the resource. You can also double-click a resource in the Hierarchy/Dependencies structure to open it.

Copy location

Copies the location of the resource.

Move resource

Moves the selected resource.

Rename resource

Renames the selected resource.

Show Resource Hierarchy

Shows the hierarchy for the selected resource.

Show Resource Dependencies

Shows the dependencies for the selected resource.

Add to Master Files

Adds the currently selected resource in *the **Master Files** directory*.

Expand All

Expands all the children of the selected resource from the Hierarchy/Dependencies structure.

Collapse All

Collapses all children of the selected resource from the Hierarchy/Dependencies structure.

Tip: When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon .

Note: The **Move resource** or **Rename resource** actions give you the option to *update the references to the resource*.

Moving/Renaming Schematron Resources

You can move and rename a resource presented in the **Resource/Hierarchy Dependencies** view, using the **Rename resource** and **Move resource** refactoring actions from the contextual menu.

When you select the **Rename** action in the contextual menu of the **Resource/Hierarchy Dependencies** view, the **Rename resource** dialog box is displayed. The following fields are available:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references** - Select this option to update the references to the resource you are renaming.

When you select the **Move** action from the contextual menu of the **Resource/Hierarchy Dependencies** view, the **Move resource** dialog box is displayed. The following fields are available:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, in accordance with the new location and name.

If the **Update references of the moved resource(s)** option is selected, a **Preview** option (which opens the **Preview** dialog box) is available for both actions. The **Preview** dialog box presents a list with the resources that are updated.

Highlight Component Occurrences in Schematron Documents

When you position your mouse cursor over a component in a Schematron document, Oxygen XML Editor searches for the component declaration and all its references and highlights them automatically.

Customizable colors are used: one for the component definition and another one for component references. Occurrences are displayed until another component is selected.

To change the default behavior of **Highlight Component Occurrences**, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Mark Occurrences**. You can also trigger a search using the **Search** > **Search Occurrences in File** **Ctrl + Shift + U (Command + Shift + U on OS X)** action from contextual menu. Matches are displayed in separate tabs of the ***Results** view*.

Searching and Refactoring Operations in Schematron Documents

Search Actions

The following search actions can be applied on `pattern`, `phase`, or `diagnostic` types and are available from the **Search** submenu in the contextual menu of the current editor or from the **Document** > **References** menu:

- **Search References** - Searches all references of the item found at current cursor position in the defined scope, if any. If a scope is defined, but the current edited resource is not part of the range of resources determined by this, a warning dialog box is displayed and you have the possibility to define another search scope.
- **Search References in** - Searches all references of the item found at current cursor position in the file or files that you specify when define a scope in the **Search References** dialog box.
- **Search Declarations** - Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined, but the current edited resource is not part of the range of resources determined by this, a warning dialog box will be displayed and you have the possibility to define another search scope.
- **Search Declarations in** - Searches all declarations of the item found at current cursor position in the file or files that you specify when you define a scope for the search operation.

- **Search Occurrences in File** - Searches all occurrences of the item at the cursor position in the currently edited file.

Refactoring Actions

The following refactoring actions can be applied on `pattern`, `phase`, or `diagnostic` types and are available from the **Refactoring** submenu in the contextual menu of the current editor or from the **Document** > **Refactoring** menu:

- **Rename Component** - Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the **Esc** or **Enter** key on your keyboard.
- **Rename Component in** - Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the **Preview** button, you can view the files to be affected by the action.

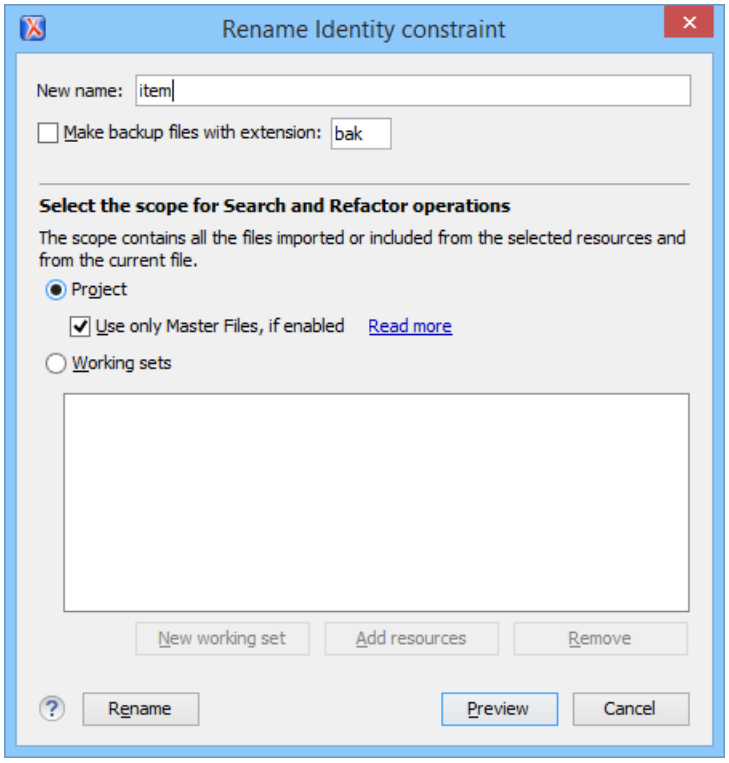


Figure 425: Rename Identity Constraint Dialog Box

Searching and Refactoring Operations Scope in Schematron Documents

The *scope* is a collection of documents that define the context of a search and refactor operation. To control it you can use the **Change scope** operation, available in the *Quick Assist* action set or on the **Resource Hierarchy/ Dependency View** toolbar. You can restrict the scope to the current project or to one or multiple *working sets*. The **Use only Master Files, if enabled** checkbox allows you to restrict the scope of the search and refactor operations to the resources from the **Master Files** directory. Click **read more** for details about the *Master Files support*.

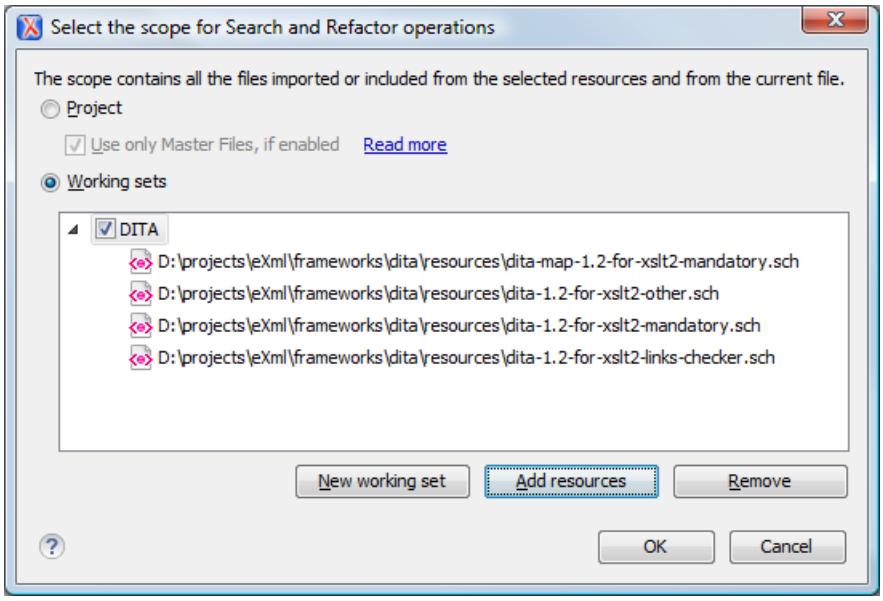


Figure 426: Change Scope Dialog Box

The scope you define is applied to all future search and refactor operations until you modify it. Contextual menu actions allow you to add or delete files, folders, and other resources to the *working set* structure.

Quick Assist Support in Schematron Documents

The *Quick Assist support* improves the development work flow, offering fast access to the most commonly used actions when you edit schema documents.

The *Quick Assist feature* is activated automatically when the cursor is positioned over the name of a component. It is accessible via a yellow bulb icon ( ) placed at the current line in the stripe on the left side of the editor. Also, you can invoke the *quick assist* menu by using the **Alt + 1** (**Meta + Alt + 1** on Mac OS X) keyboard shortcuts.

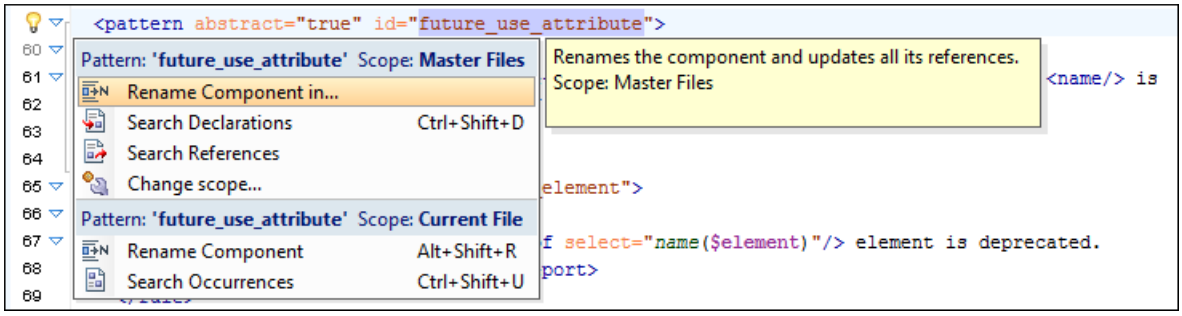


Figure 427: Schematron Quick Assist Support

The *Quick Assist* support offers direct access to the following actions:

Rename Component in

Renames the component and all its dependencies.

Search Declarations

Searches the declaration of the component in a predefined scope. It is available only when the context represents a component name reference.

Search References

Searches all references of the component in a predefined scope.

Component Dependencies

Searches the component dependencies in a predefined scope.

Change Scope

Configures the scope that will be used for future search or refactor operations.

Rename Component

Allows you to rename the current component in-place.

Search Occurrences

Searches all occurrences of the component within the current file.

Editing Schematron Quick Fixes

Oxygen XML Editor provides support for editing the *Schematron Quick Fixes*. You can define a library of *Quick Fixes* by editing them directly in the current Schematron file or in a separate file. Oxygen XML Editor assists you in editing Schematron *Quick Fixes* with schema-based content completion, syntax highlighting, and validation as you type.

For information about applying and detecting the Schematron schemas that include SQF, see *Associating a Schema to XML Documents* on page 464.

Related Information:

Oxygen XML Blog: Schematron Checks to Help Technical Writing

Schematron Quick Fixes (SQF)

Oxygen XML Editor provides support for Schematron *Quick Fixes* (SQF). They help you resolve issues that appear in XML documents that are validated against Schematron schemas by offering you solution proposals. The Schematron *Quick Fixes* are an extension of the Schematron language and they allow you to define fixes for Schematron validation messages. Specifically, they are associated with *assert* or *report* messages.

A typical use case is using Schematron *Quick Fixes* to assist content authors with common editing tasks. For example, you can use Schematron rules to automatically report certain validation warnings (or errors) when performing regular editing tasks, such as inserting specific elements or changing IDs to match specific naming conventions. For more details and examples, please see the following blog post: *http://blog.oxygenxml.com/2015/05/schematron-checks-to-help-technical.html*.

Displaying the Schematron Quick Fix Proposals

The defined Schematron *Quick Fixes* are displayed on validation errors in **Text** mode and **Author** mode.

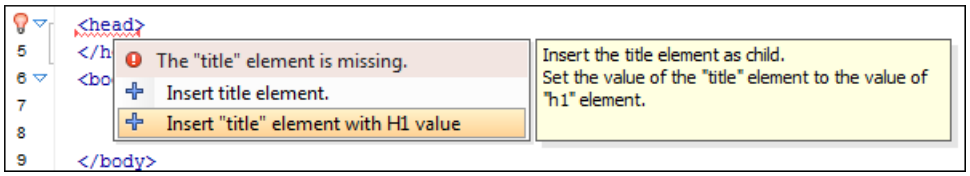


Figure 428: Example of a Schematron Quick Fix

Related Information:

Editing Schematron Quick Fixes on page 695

Schematron Quick Fix Specifications

Presenting Schematron Validation Issues

Defining Schematron Quick Fixes

You can define and customize Schematron *Quick Fixes* directly in the current Schematron file or in a separate Schematron file. The Schematron *Quick Fixes* are an extension of the Schematron language and they allow you to define fixes for Schematron error messages. You can refer the *Quick Fixes* from the `assert` or `report` elements in the values of the `sqf:fix` attributes.

Defining a Schematron Quick Fix

The basics of a Schematron *Quick Fix* is defined by an ID, name, description, and the operations to be executed.

- **ID** - Defined by the `id` attribute from the `fix` element and must be unique in the current context. It is used to refer the *Quick Fix* from a `report` or `assert` element.
- **Name** - The name of the *Quick Fix* is defined by the `title` element.
- **Description** - Defined by the text in the paragraphs (p) of the `description` element.
- **Operations** - The following basic types of *operations (elements)* are supported:
  - `<sqf:add>` Element - To add a new node or fragment in the document.
  - `<sqf:delete>` Element - To remove a node from the document.
  - `<sqf:replace>` Element - To replace a node with another node or fragment.
  - `<sqf:stringReplace>` Element - To replace text content with other text or a fragment.

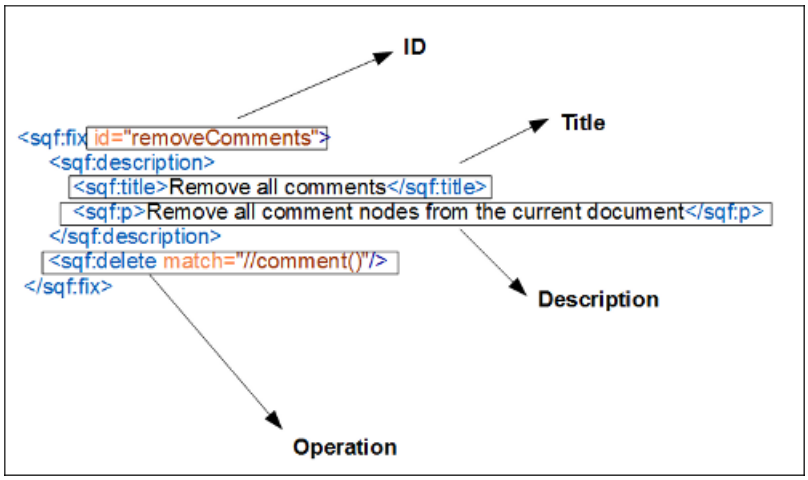


Figure 429: Schematron Quick Fix Components

The assertion message that generates the *Quick Fix* is added as the `description` of the problem to be fixed. The `title` is presented as the name of the *Quick Fix*. The content of the paragraphs (p) within the `description` element are presented in the tooltip message when the *Quick Fix* is selected.

Additional Elements Supported in the Schematron Quick Fixes

`<sqf:user-entry>`

This element defines a value that must be set manually by the user. For more information, see *User Entry SQF Operation* on page 699.

`<sqf:call-fix>`

This element calls another *Quick Fix* within a *Quick Fix*. The called *Quick Fix* must be defined globally or in the same Schematron rule as the calling *Quick Fix*. A calling *Quick Fix* adopts the activity elements of the called *Quick Fix* and should not include other activity elements. You can also specify which parameters are sent by using the `<sqf:with-param>` child element.

`<sqf:group>`

Allows you to group multiple *Quick Fixes* and refer them from an `assert` or `report` element.

`<sqf:fixes>`

Is defined globally and contains global fixes and groups of fixes.

`<sqf:keep>`

Used to copy the selected nodes that are specified by the `select` attribute.

Note: In Oxygen XML Editor the copied nodes cannot be manipulated by the current or other activity elements.

`<sqf:param>`

Defines a parameter for a *Quick Fix*. If the parameter is defined as `abstract` then the `type` and default value should not be specified and the fix can be called from an abstract pattern that defines this parameter.

Other SQF Notes

Note: The `sqf:default-fix` attribute is ignored in Oxygen XML Editor.

For more details on editing Schematron *Quick Fixes*, go to: *Schematron Quick Fix Specifications*

Basic Schematron Quick Fix Operations

There are four basic operations that can be executed in a Schematron *Quick Fix*: **Add**, **Delete**, **Replace**, and **String Replace**.

Add

The `<sqf:add>` element allows you to add a node to the instance. An *anchor* node is required to select the position for the new node. The *anchor* node can be selected by the `match` attribute. Otherwise, it is selected by the `context` attribute of the rule.

The `target` attribute defines the name of the node to be added. It is required if the value of the `node-type` attribute is set to anything other than "comment".

The `<sqf:add>` element has a `position` attribute and it determines the position relative to the *anchor* node. The new node could be specified as the first child of the *anchor* node, the last child of the *anchor* node, before the *anchor* node, or after the *anchor* node (`first-child` is the default value). If you want to add an attribute to the *anchor* node, do not use the `position` attribute.

Note: If you insert an element and its content is empty, Oxygen XML Editor will insert the required element content.

An Example of the `<sqf:add>` Element:

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
   xmlns:sqf="http://www.schematron-quickfix.com/validator/process"
 queryBinding="xslt2">
   <pattern>
     <rule context="head">
        <assert test="title" sqf:fix="addTitle">title element missing.</assert>
        <sqf:fix id="addTitle">
            <sqf:description>
                <sqf:title>Insert title element.</sqf:title>
            </sqf:description>
           <sqf:add target="title" node-type="element">Title text</sqf:add>
        </sqf:fix>
     </rule>
   </pattern>
</schema>
```

Specific Add Operations:

- **Insert Element** - To insert an element, use the `<sqf:add>` element, set the value of the `node-type` to "element", and specify the element *QName* with the `target` attribute. If the element has a prefix, it must be defined in the Schematron using a namespace declaration (`<ns uri="namespace" prefix="prefix"/>`).
- **Insert Attribute** - To insert an attribute, use the `<sqf:add>` element, set the value of the `node-type` to "attribute", and specify the attribute *QName* with the `target` attribute. If the attribute has a prefix, it must be defined in the Schematron using a namespace declaration (`<ns uri="namespace" prefix="prefix"/>`).
- **Insert Fragment** - If the `node-type` is not specified, the `<sqf:add>` element will insert an XML fragment. The XML fragment must be well formed. You can specify the fragment in the `add` element or by using the `select` attribute.
- **Insert Comment** - To insert a comment, use the `<sqf:add>` element and set the value of the `node-type` to "comment".
- **Insert Processing Instruction** - To insert a processing instruction, use the `<sqf:add>` element, set the value of the `node-type` to "pi" or "processing-instruction", and specify the name of the processing instruction in the `target` attribute.

Delete

The `<sqf:delete>` element allows you to remove any type of node (such as elements, attributes, text, comments, or processing instructions). To specify nodes for deletion the `<sqf:delete>` element can include a `match` attribute that is an XPath expression (the default value is `.`). If the `match` attribute is not included, it deletes the context node of the Schematron rule.

An Example of the `<sqf:delete>` Element:

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2"
    xmlns:sqf="http://www.schematron-quickfix.com/validator/process">
    <pattern>
        <rule context="*[@xml:lang]">
            <report test="@xml:lang" sqf:fix="remove_lang">
                The attribute "xml:lang" is forbidden.</report>
            <sqf:fix id="remove_lang">
                <sqf:description>
                    <sqf:title>Remove "xml:lang" attribute</sqf:title>
                </sqf:description>
                <sqf:delete match="@xml:lang"/>
            </sqf:fix>
        </rule>
    </pattern>
</schema>
```

Replace

The `<sqf:replace>` element allows you to replace nodes. Similar to the `<sqf:delete>` element, it can include a `match` attribute. Otherwise, it replaces the context node of the rule. The `<sqf:replace>` element has three tasks. It identifies the nodes to be replaced, defines the replacing nodes, and defines their content.

An Example of the `<sqf:replace>` Element:

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
    xmlns:sqf="http://www.schematron-quickfix.com/validator/process"
 queryBinding="xslt2">
    <pattern>
        <rule context="title">
            <report test="exists(ph)" sqf:fix="resolvePh" role="warn">
                ph element is not allowed in title.</report>
            <sqf:fix id="resolvePh">
                <sqf:description>
                    <sqf:title>Change the ph element into text</sqf:title>
                </sqf:description>
                <sqf:replace match="ph">
                    <value-of select="."/>
                </sqf:replace>
            </sqf:fix>
        </rule>
    </pattern>
</schema>
```

Other Attributes for Replace Operations:

- **`node-type`** - Determines the type of the replacing node. The permitted values include:
  - `keep` - Keeps the node type of the node to be replaced.
  - `element` - Replaces the node with an element.
  - `attribute` - Replaces the node with an attribute.
  - `pi` - Replaces the node with a processing instruction.
  - `comment` - Replaces the node with a comment.
- **`target`** - By using a *QName* it gives the replacing node a name. This is necessary when the value of the `node-type` attribute is anything other than "comment".
- **`select`** - Allows you to choose the content of the replacing nodes. You can use XPath expressions with the `select` attribute. If the `select` attribute is not specified then the content of the `<sqf:replace>` element is used instead.

String Replace

The `<sqf:stringReplace>` element is different from the others. It can be used to find a sub-string of text content and replace it with nodes or other strings.

Attributes for the String Replace Operation:

- **`match`** - Allows you to select text nodes that contain the sub-strings you want to replace.

- **`select`** - Allows you to select the replacing fragment, in case you do not want to set it in the content of the `stringReplace` element.
- **`regex`** - Matches the sub-strings using a regular expression.

  Note: Consider the following information about using regular expressions in the `stringReplace` element:

  - The regular expressions from this operation are compiled as Java regular expressions. For more information, see *https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html*.
  - The ***j flag*** allows you to use the standard Java regular expression engine, which allows native Java regular expression syntax. This allows, for example, the use of **`\b`** in a regular expression to match word boundaries. For more information, see *http://www.saxonica.com/html/documentation/functions/fn/matches.html*.
  - Regular expressions in the `<sqf:stringReplace>` element always have the *dot matches all* flag set to "true". Therefore, the line terminator will also be matched by the regular expression.

Attention: The context of the content within the `<sqf:stringReplace>` element is set to the whole text node, rather than the current sub-string.

An Example of the `<sqf:stringReplace>` Element:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
    xmlns:sqf="http://www.schematron-quickfix.com/validator/process"
 queryBinding="xslt2">
    <sch:pattern>
        <sch:rule context="text()">
            <sch:report test="matches(., '[oO][xX]ygen')"
 sqf:fix="changeWord">The oXygen word is not allowed</sch:report>
            <sqf:fix id="changeWord">
                <sqf:description>
                    <sqf:title>Replace word with product</sqf:title>
                </sqf:description>
                <sqf:stringReplace regex="[oO][xX]ygen"><ph keyref="product"/>
                </sqf:stringReplace>
            </sqf:fix>
        </sch:rule>
    </sch:pattern>
</sch:schema>
```

Related Information:

User Entry SQF Operation on page 699
Restricting Quick Fix Operations on page 699

User Entry SQF Operation

The `<sqf:user-entry>` element defines a value that must be set manually by the user. If multiple `user-entry` elements are defined, Oxygen XML Editor will display a dialog box for each one, in which the user can specify values. Also, the `<user-entry>` element can be used as an XPath variable where the XPath variable is the name of the `user-entry`.

An Example of the `<sqf:user-entry>` Element:

```
<sqf:fix id="duplicate">
   <sqf:description>
      <sqf:title>Change the name of the element</sqf:title>
   </sqf:description>
   <sqf:user-entry name="newName" default="product">
      <sqf:description>
          <sqf:title>Enter the new name of the element</sqf:title>
      </sqf:description>
   </sqf:user-entry>
   <sqf:replace node-type="element" target="{$newName}" select="child::node()"/>
</sqf:fix>
```

Related Information:

Basic Schematron Quick Fix Operations on page 697

Restricting Quick Fix Operations

To restrict a *Quick Fix* or a specific operation to only be available if certain conditions are met, the `use-when` attribute can be included in the `<sqf:fix>` element or any of the SQF operation elements. The condition of the

use-when attribute is an XPath expression and the fix or operation will be performed only if the condition is satisfied. In the following example, the use-when condition is applied to the <sqf:fix> element:

```
<sqf:fix id="last" use-when="$colWidthSummarized - 100 lt $lastWidth"
 role="replace">
   <sqf:description>
       <sqf:title>Subtract excessive width from the last element.</sqf:title>
   </sqf:description>
   <let name="delta" value="$colWidthSummarized - 100"/>
   <sqf:add match="html:col[last()]" target="width" node-type="attribute">
    <let name="newWidth" value="number(substring-before(@width,'%')) - $delta"/>
       <value-of select="concat($newWidth,'%')"/>
   </sqf:add>
</sqf:fix>
```

Related Information:

Basic Schematron Quick Fix Operations on page 697

Formatting/Indenting Content Inserted by SQF Operations

Content that is inserted by the **Add**, **Replace**, or **String Replace** Schematron *Quick Fix* operations is automatically indented unless you set the value of the xml:space attribute to preserve on the operation element. There are several methods available to format the content that is inserted:

- **xsl:text** - You can use an xsl:text element to format the inserted content and keep the automatic indentation, as in the following example:

```
<sqf:add position="last-child">
    <row><xsl:text>
    </xsl:text>
        <entry>First column</entry><xsl:text>
        </xsl:text>
        <entry>Second column</entry><xsl:text>
        </xsl:text>
    </row><xsl:text>
    </xsl:text>
</sqf:add>
```

- **xml:space** - Use the xml:space attribute and set its value to preserve to format the content and specify the spacing between elements, as in the following example:

```
<sqf:add node-type="element" target="codeblock" xml:space="preserve">
    /* a long sample program */
    Do forever
     Say "Hello, World"
    End</sqf:add>
```

Related Information:

Basic Schematron Quick Fix Operations on page 697

Executing Schematron Quick Fixes in Other Documents

You can apply Schematron *Quick Fixes* over nodes from referenced documents (using XInclude or external entities), and you can access them as nodes in your current document.

Also, you can apply the *Quick Fixes* over other documents using the doc( ) function in the value of the match attribute. For example, you can add a new key in the keylist.xml file using the following operation:

```
<sqf:add match="doc('keylist.xml')/KeyList" target="Key"
 node-type="element" select="Key2">
```

Validating Schematron Quick Fixes

By default, Schematron *Quick Fixes* are validated as you edit them within the Schematron file or while editing them in a separate file. To change this, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Editor** > **Document Checking**, and deselect the ***Enable automatic validation** option*.

To validate Schematron *Quick Fixes* manually, select the **Validate** action from the **Validation** toolbar drop-down menu or the **Document** > **Validate** menu. The validation problems are highlighted directly in the editor, making it easy to locate and fix any issues.

Related Information:

Validating XML Documents Against a Schema on page 446
Validation Scenario on page 452
Presenting Validation Errors in Author Mode on page 236
Presenting Validation Errors in Text Mode on page 206

Content Completion in SQF

Oxygen XML Editor helps you edit Schematron *Quick Fixes* embedded in a Schematron document by offering proposals that are valid at the cursor position in a *Content Completion Assistant*. It can be manually activated with the **Ctrl + Space (Command + Space on OS X)** shortcut.

When you edit the value of an attribute that references a *Quick Fix ID*, the ids are collected from the entire definition scope. For example, if the editing context is `assert/@sqf:fix`, the *Content Completion Assistant* proposes all fixes defined locally and globally.

If the editing context is an attribute value that is an XPath expression (such as `sqf:add/@match` or `replace/@select`), the *Content Completion Assistant* offers the names of XPath functions, the XPath axes, and user-defined variables and parameters.

The *Content Completion Assistant* displays XSLT 1.0 functions (and optionally XSLT 2.0 / 3.0 functions) in the attributes *path*, *select*, *context*, *subject*, and *test*, depending on *the Schematron options* that are set in Preferences pages. If the Saxon 6.5.5 namespace (`xmlns:saxon="http://icl.com/saxon"`) or the Saxon 9.7.0.15 namespace is declared in the Schematron schema (`xmlns:saxon="http://saxon.sf.net/"`) the content completion also displays the XSLT Saxon extension functions.

Highlight Quick Fix Occurrences in SQF

When you position your mouse cursor over a *Quick Fix* ID in a Schematron document, Oxygen XML Editor searches for the *Quick Fix* declaration and all its references and highlights them automatically.

Customizable colors are used: one for the *Quick Fix* definition and another one for its references. Occurrences are displayed until another *Quick Fix* is selected.

To change the default behavior of **Highlight Component Occurrences**, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Mark Occurrences**. You can also trigger a search using the **Search** > **Search Occurrences in File (Ctrl + Shift + U (Command + Shift + U on OS X))** action from contextual menu. Matches are displayed in separate tabs of the ***Results** view*.

Searching and Refactoring Operations in SQF

Search Actions

The following search actions can be applied on *Quick Fix* IDs and are available from the **Search** submenu in the contextual menu of the current editor or from the **Document** > **References** menu:

- **Search References** - Searches all references of the item found at current cursor position in the defined scope, if any. If a scope is defined, but the current edited resource is not part of the range of resources determined by this, a warning dialog box is displayed and you have the possibility to define another search scope.
- **Search References in** - Searches all references of the item found at current cursor position in the file or files that you specify when define a scope in the **Search References** dialog box.
- **Search Declarations** - Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined, but the current edited resource is not part of the range of resources determined by this, a warning dialog box will be displayed and you have the possibility to define another search scope.
- **Search Declarations in** - Searches all declarations of the item found at current cursor position in the file or files that you specify when you define a scope for the search operation.
- **Search Occurrences in File** - Searches all occurrences of the item at the cursor position in the currently edited file.

Refactoring Actions

The following refactoring actions can be applied on *Quick Fix* IDs and are available from the **Refactoring** submenu in the contextual menu of the current editor or from the **Document** > **Refactoring** menu:

- **Rename Component** - Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the **Esc** or **Enter** key on your keyboard.
- **Rename Component in** - Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the **Preview** button, you can view the files to be affected by the action.

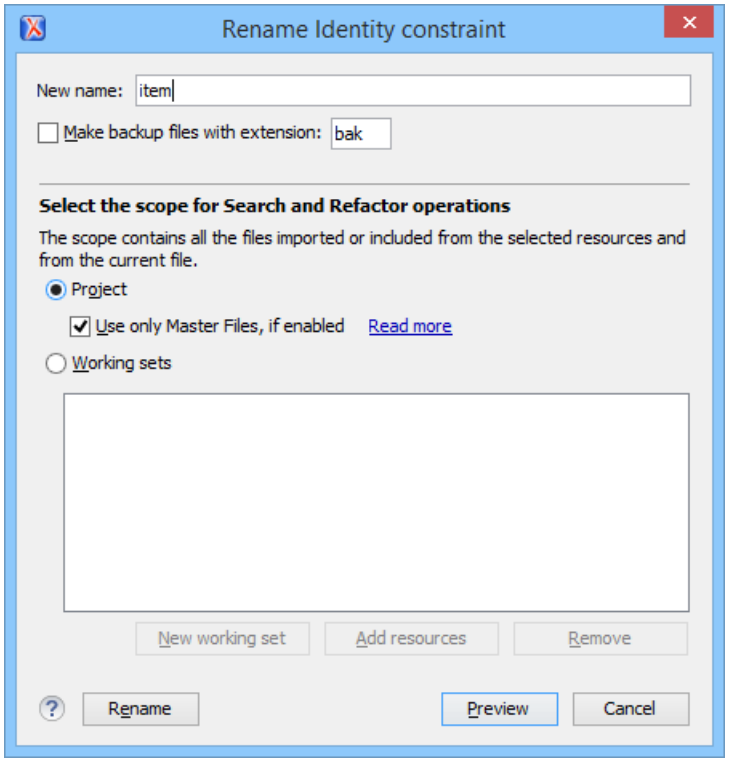


Figure 430: Rename Identity Constraint Dialog Box

Embed Schematron Quick Fixes in Relax NG or XML Schema

Schematron *Quick Fixes* can be embedded into an XML Schema through annotations (using the `appinfo` element), or in a schematron rule embedded in the RELAX NG Schema. For more information about embedding schematron in XML Schema or Relax NG, see *XML Schema or RELAX NG with Embedded Schematron Rules* on page 688.

Oxygen XML Editor is able to extract and use the embedded Schematron *Quick Fixes*. To make the embedded Schematron *Quick Fixes* available, follow these steps:

1. Define a *validation against a schema*.
2. For the **Schema type**, choose `XML Schema` or `Relax NG`.
3. Select the **Embedded schematron rules** option.

Example: Embedded Schematron Quick Fix in XML Schema

```
<xsd:appinfo>
  <sch:pattern>
    <sch:rule context="...">
 <sch:assert test="..." sqf:fix="fixId">Message.</sch:assert>
   <sqf:fix id="fixId">
     ......
   </sqf:fix>
    </sch:rule>
```

```
  </sch:pattern>
</xsd:appinfo>
```

Example: Embedded Schematron Quick Fix in Relax NG

```
<sch:pattern>
  <sch:rule context="...">
    <sch:assert test="..." sqf:fix="fixId">Message.</sch:assert>
 <sqf:fix id="fixId">
   .....
 </sqf:fix>
  </sch:rule>
</sch:pattern>
```

Tip: For more extensive examples, see our samples in the *[OXYGEN_INSTALL_DIR]*`/samples/schematron` folder.

Related Information:

XML Schema or RELAX NG with Embedded Schematron Rules on page 688

Defining Schematron Quick Fixes on page 695

Integrating SQF in a Framework

You can use Schematron *Quick Fixes* to assist your content authors by imposing rules for an entire *framework* (document type) and offering fixes when a rule violation is detected.

For example, if you are using DITA, you may want your contributors to avoid inserting a figure (`fig` element) inside a paragraph (p element) that contains other content since it may result in undesirable placement or spacing in the output. The Schematron rule and its *Quick Fix* for this particular use-case could look like this:

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
    xmlns:sqf="http://www.schematron-quickfix.com/validator/process"
 queryBinding="xslt2">
   <pattern id="check.figure.location">
     <rule context="p/fig">
         <report test="true()" role="warn" sqf:fix="moveAfter">
         A figure inside a paragraph doesn't transform well into PDF. </report>
         <sqf:fix id="moveAfter">
             <sqf:description>
                 <sqf:title>Move after the paragraph.</sqf:title>
             </sqf:description>
             <let name="figToMove" value="."/>
             <sqf:add match="parent::p" select="$figToMove" position="after"/>
             <sqf:delete match="."/>
         </sqf:fix>
     </rule>
   </pattern>
</schema>
```

The result of this example would be that the user will see a warning if they insert a `fig` element inside a p element and they are presented with the option of selecting the *Quick Fix* that would move the figure outside the paragraph.

How to Integrate SQF in a Framework

To integrate a Schematron *Quick Fix* in a *framework*, follow these steps:

1. *Define a Schematron Quick Fix for a rule* in an existing or new Schematron file.
2. Save it somewhere in your *framework* directory. For example, the default *framework* directory for DITA is located in: *[OXYGEN_INSTALL_DIR]*`/frameworks/dita/`.
3. Add a reference to the Schematron file that includes the SQF in your *framework* by following the procedure in *Associating a Schema in Validation Scenarios Defined in the Document Type* on page 467.
4. Open a document in your *framework* and test the new rule and *Quick Fix*.
5. You can continue to refine the rule and develop additional rules as needed.

Related Information:

Defining Schematron Quick Fixes on page 695

Basic Schematron Quick Fix Operations on page 697

Associating a Schema in Validation Scenarios Defined in the Document Type on page 467

Editing SVG Files

SVG (Scalable Vector Graphics) is a platform for two-dimensional graphics. It has two parts: an XML-based file format and a programming API for graphical applications. Some of the key features include support for shapes, text, and embedded raster graphics with many painting styles, scripting through languages such as *ECMAScript*, and support for animation.

SVG is a vendor-neutral, open standard that has important industry support. Companies such as Adobe, Apple, and IBM have contributed to its W3C specifications. Many documentation *frameworks* (including DocBook) have support for SVG by defining the graphics directly in the document.

Oxygen XML Editor adds SVG support by using the *Batik distribution* package (an open source project developed by the Apache Software Foundation) and the *default XML Catalog* resolves the SVG DTD.

Note: Batik partially supports SVG 1.1. For a detailed list of supported elements, attributes, and properties, see the *Batik Implementation Status* page.

How to Render SVG Images that Use Java Scripting

1. Copy the `js.jar` library from the *Batik distribution* into the Oxygen XML Editor `lib` folder.
2. Restart the application.

SVG 1.2 Rendering Issues

Oxygen XML Editor uses the *Apache Batik* open source library to render SVG images and it only has partial support for SVG 1.2. For more information, see *http://xmlgraphics.apache.org/batik/dev/svg12.html*.

This partial support could lead to some rendering issues in Oxygen XML Editor. For example, if you are using the *Inkscape* SVG editor, it is possible for it to save the SVG as 1.1, while using SVG 1.2 elements (such as `flowRoot`) inside it. This means that the image will not be properly rendered inside the application.

Note: SVG images shown in the **Author** visual editing mode are rendered as static images, without support for animations and JavaScript.

Standalone SVG Viewer

Oxygen XML Editor includes a simple **SVG Viewer** that allows you to work with SVG images.

To open the viewer, select **SVG Viewer** from the **Tools** menu.

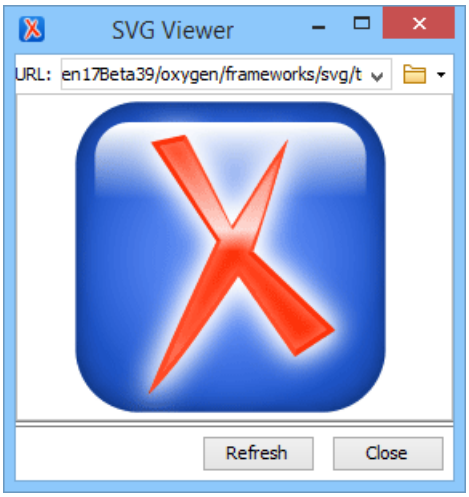


Figure 431: SVG Viewer

You can browse for and open any SVG file that has the `.svg` or `.svgz` extension.

If the file is included in the current project, you can open it in the viewer by right-clicking the image file in the ***Project** view* and selecting **Open with** > **SVG Viewer**.

Actions Available in the SVG Viewer

The following actions are available in the **SVG Viewer**:

Zoom in

To zoom in on an image, use any of the following methods:

- Scroll **forward** with the mouse wheel.
- Select **Zoom in** from the contextual menu.
- Use the **Ctrl + I (Command + I on OS X)** keyboard shortcut.

Zoom out

To zoom in on an image, use any of the following methods:

- Scroll **backward** with the mouse wheel.
- Use the **Ctrl + O (Command + O on OS X)** keyboard shortcut.
- Select **Zoom out** from the contextual menu.

Rotate

To rotate an image, use either of the following methods:

- Use the **Ctrl + Right-Click + Drag (Command + Right-Click + Drag on OS X)** shortcut.
- Select **Rotate** from the contextual menu. This rotates the image exactly 90 degrees clockwise.

Refresh

To refresh (or reset) an image, use either of the following methods:

- Use the **Ctrl + T (Command + T on OS X)** keyboard shortcut.
- Select **Refresh** from the contextual menu.

Move

To move an image, use either of the following methods:

- Use the **Arrow Keys** on your keyboard.
- Use the **Shift + Left-Click + Drag** shortcut.

Pan

To pan an image, **click and drag** the image with your mouse.

Related Information:

Editing SVG Files on page 704

Integrated SVG Viewer in the Results Panel

Oxygen XML Editor includes an integrated **SVG Viewer** that can render the results of an XSLT transformation scenario that generates SVG images in the ***Results** panel* at the bottom of the editor. This is useful for developing XSL stylesheets with the capability of producing SVG graphics.

To enable this feature, select **Show in results view as** > **SVG** in the ***Output** tab of the XSLT transformation scenario configuration dialog box*. When you run the transformation, the SVG result is then rendered in an integrated SVG viewer in the ***Results** panel*.

Example of a Use-Case

Suppose you have an XML document that describes the evolution of your sales over a time period and you want to create a graphic for it. You could use the following steps to accomplish this task:

1. Start with a static SVG image, written directly in Oxygen XML Editor or exported from a external graphics tool.
2. Extract the parts that are dependent upon the data from the XML document and create an XSL template to produce the image.
3. Create an ***XML transformation with XSLT** scenario*.
4. While configuring the transformation scenario, select ***Show in results view as** > **SVG** in the **Output** tab* of the configuration dialog box.

5. Run the transformation.

The SVG image is rendered in an integrated SVG viewer in the ***Results** panel* at the bottom of the editor.

Figure 432: Integrated SVG Viewer

Editing XHTML Documents

Oxygen XML Editor provides support for editing XHTML files. Oxygen XML Editor includes XHTML catalogs and document templates to help you get started. You can also find a sample file in the `[OXYGEN_INSTALL_DIR]/samples/xhtml` folder.

The XHTML editing features include:

- **Source Editing** - You can edit XHTML files in the ***Text** editing mode* (XML source editor) using all of its *useful features*.
- **Visual Editing** - You can edit XHTML files in the *visual **Author** editing mode* using all of its *authoring features*.
- **Validation** - Easily identify errors and their location with the Oxygen XML Editor *XML validation features*.
- **Content Completion** - The *Content Completion Assistant* displays a list of context-sensitive proposals that are valid at the current cursor position. In **Text** mode, it can be manually activated with the **Ctrl + Space (Command + Space on OS X)** shortcut, while in **Author** mode, it is activated by simply pressing **Enter**.
- **Import HTML as XHTML** - Oxygen XML Editor includes support for *importing HTML files as an XML document*.
- **Remote Editing** - Oxygen XML Editor has built-in support for *editing documents that are stored on remote servers through FTP, SFTP, and WebDAV protocols*, allowing you to edit XHTML pages from your web server.
- **Syntax Highlighting** - XHTML documents with embedded CSS, JS, PHP, and JSP scripts are rendered with dedicated coloring schemes. To customize them, *open the **Preferences** dialog box **(Options > Preferences)*** and go to ***Editor > Syntax Highlight***. Select and expand the **XHTML** section in the top pane and you can see the effects of your changes in the **Preview** pane.

Related Information:

XHTML Document Type (Framework) on page 833

Editing Markdown Documents

Markdown was created as a lightweight markup language with plain text formatting syntax designed to provide a syntax that is very easy to read and write, and to convert it to HTML and other formats. It is often used by content contributors who want a quick and easy way to write content without having to take their fingers off the keyboard and without having to learn numerous codes or shortcuts, and it can easily be shared interchangeably between virtually any types of contributor and system.

Oxygen XML Editor provides a built-in Markdown editor that allows you to convert Markdown syntax to HTML or DITA and it includes a preview panel to help you visualize the final output. Aside from the plain text syntax that is common amongst most Markdown applications, the editor in Oxygen XML Editor also integrates many other

powerful features that content authors are accustomed to using for other types of documents. Some of these additional unique features include:

- Additional toolbar and contextual menu actions.
- Automatic validation to help keep the syntax valid.
- Dedicated syntax highlighting to make Markdown documents even easier to read and write.
- Unique features for creating Markdown documents directly in *DITA maps* and converting Markdown documents to DITA topics.
- Specialized syntax rules to combine popular syntax features from several specifications.

Markdown Editor

Oxygen XML Editor provides an intuitive, dynamic, and easy-to-use Markdown editor for writing and converting Markdown documents. It is a split-screen editor with two panels that are synchronized in real-time. The left side is a simple text editor that is specially designed for writing Markdown syntax. The right side is a WYSIWYG style preview of how changes will look in the output.

Markdown Text Editor Pane (Left Side)

The left pane is a simple text editor that is refined to accept Markdown syntax. At the same time, you still have many of the actions, options, and features that you are used to when editing any other type of document in Oxygen XML Editor.

The features of this special editor that are unique for Markdown documents include:

- **Unique Markdown Syntax Rules** - The Markdown editor in Oxygen XML Editor uses *an integration of rules* that combine rules from common default Markdown syntax along with many of the rules used in the GitHub Flavored Markdown syntax.
- **Syntax Highlighting** - The Oxygen XML Editor syntax highlighting feature is integrated into the Markdown text editor to make it easier to read and write Markdown syntax. You can even *customize the colors and styles for the syntax highlighting*.
- **Automatic Spell Checking** - The Markdown editor supports the Oxygen XML Editor *automatic spell checking feature* that reports possible misspelled words as you type. You simply need to select the ***Automatic spell check** option in the **Spell Check** preferences page*, then click the **Select editors** button and select **Markdown Editor**.
- **Helpful Toolbar and Contextual Menu Actions** - A *variety of unique actions* are available from the toolbar to help you insert proper Markdown syntax. The contextual menu also includes some common editing actions, as well as unique actions to export (convert) Markdown documents to HTML or DITA.
- **Shortcut Keys** - Many of the *shortcut keys that are most commonly used* in Oxygen XML Editor also work in the Markdown editor.

WYSIWYG Preview Pane (Right Side)

The right pane is a WYSIWYG *Preview* pane that shows a visual representation of how changes made in the left-side text editor will be converted to HTML or DITA output. The changes you make in the text editor are parsed continually and they are immediately visible in the *Preview* pane. There are two tabs available in the *Preview* pane, one for visualizing DITA output and one for visualizing HTML output. You can switch between the two tabs at the bottom of the pane.

The *Preview* pane includes the following unique features:

- **WYSIWYG Visualization** - This pane presents the Markdown syntax from the left-side text editor in a visual WYSIWYG style interface that is automatically synchronized as you type.
- **Export Options** - The **DITA** tab includes a contextual menu action to *export (convert) the current Markdown document to a DITA topic*. Similarly, the **HTML** tab includes a contextual menu action to *export (convert) it to an XHTML document*.
- **Automatic Validation** - As you edit Markdown documents, they are *validated automatically*. The conversion engine constantly tries to parse your changes and if a change results in an error that prevents the parser from converting the syntax, an error message will be displayed in the *Preview* pane or ***Results** view* at the bottom of the editor.

- **Print Feature** - The Markdown editor includes a **Print** action that is available in the contextual menu and it allows you to configure options for printing the current document as you see it in the *Preview* pane.
- **Preview Markup** - The Markdown editor includes a ***Tags Display Mode** drop-down menu* that is available on the toolbar and it allows you to control the amount of markup that is displayed in the *Preview* pane.
- **Specialized DITA Features** - The Markdown editor includes some unique, *specialized features to integrate it with the powerful DITA support* in Oxygen XML Editor.

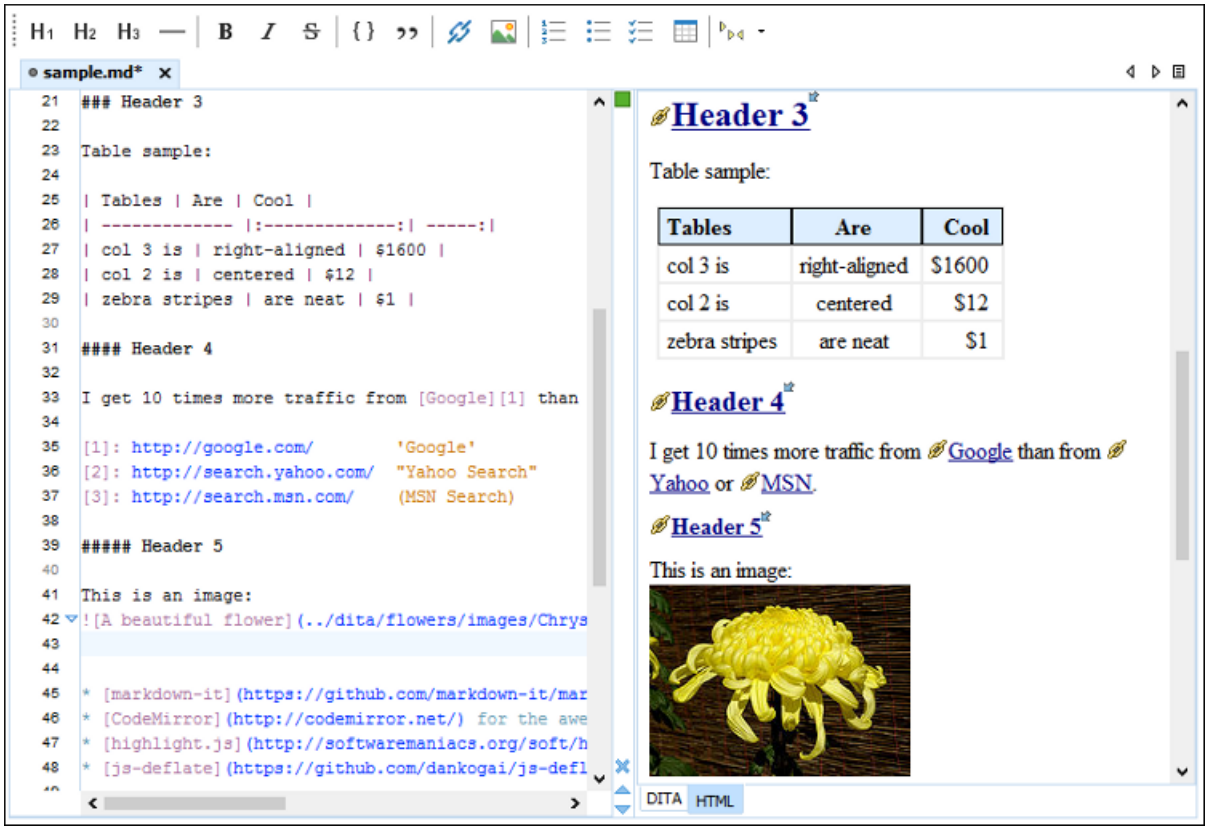


Figure 433: Markdown Editor in Oxygen XML Editor

Related Information:

Markdown Editor Syntax Rules and Specifications on page 715
Actions Available in the Markdown Editor on page 709
Working with Markdown Documents in DITA on page 714
Creating New Markdown Documents on page 708

Creating New Markdown Documents

To create a new Markdown document in Oxygen XML Editor, follow these steps:

1. Click the **New** button on the toolbar or select **File** > **New**.
2. Select the **Markdown** file template (in the **New Document** folder).
3. Click the **Create** button.

   Result: A new Markdown document is created and it is opened in the specialized *Markdown Editor*.

Related Information:

Markdown Editor on page 707

Actions Available in the Markdown Editor

Aside from the actions that are available in Oxygen XML Editor for any type of document (such as the actions in the various menus and the common sections of the toolbar), a variety of unique actions are also available in the Markdown editor, from the toolbar and contextual menu.

Toolbar Actions

The following default actions are readily available on the toolbar when editing Markdown documents:

Header (1st Level)

Inserts an *atx-style first level header* at the cursor position.

Header (2st Level)

Inserts an *atx-style second level header* at the cursor position.

Header (3rd Level)

Inserts an *atx-style third level header* at the cursor position.

Horizontal Rule

Inserts a *horizontal rule* at the cursor position.

Bold (Strong)

Marks the selected text with *bold*.

Italic (Emphasis)

Marks the selected text with *italics*.

Strikethrough

Marks the selected text with a *strikethrough*.

Code Block

Inserts (or surrounds selected text in) a *codeblock*.

Blockquote

Inserts a *blockquote* at the cursor position.

Insert Link

Opens the **Insert Link** dialog box that allows you to define a *link* to insert at the cursor position.

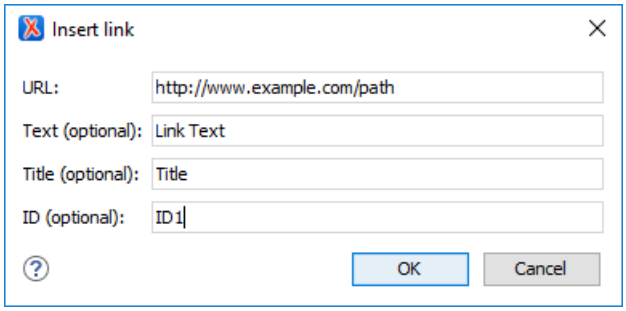


Figure 434: Insert Link Dialog Box

Insert Image

Opens the **Insert Image** dialog box that allows you to define an *image* to insert at the cursor position. You can type the URL of the image you want to insert or use browsing tools in the **Browse** drop-down menu.

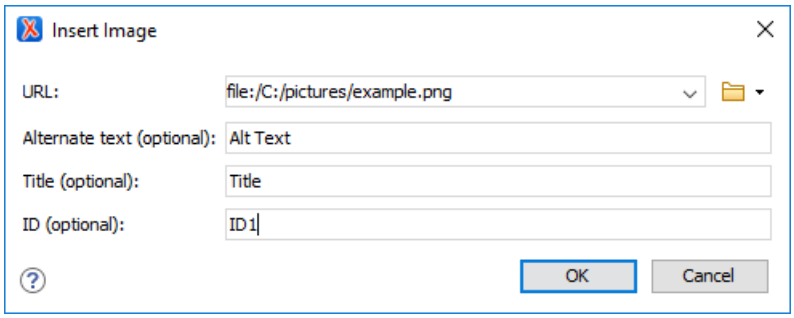


Figure 435: Insert Link Dialog Box

Insert Ordered List

Inserts an *ordered list* at the cursor position. Three child list items are also automatically inserted by default.

Insert Unordered List

Inserts an *unordered list* at the cursor position. Three child list items are also automatically inserted by default.

Insert Task List

Inserts a *task list* at the cursor position. Three child list items are also automatically inserted by default.

Insert Table

Inserts a *table* at the cursor position.

Tags Display Mode drop-down menu

Allows you to control the amount of markup that is displayed in the *Preview* pane and offers the following choices:

Full Tags with Attributes

Displays full tag names with attributes for both *block* and *inline* elements.

Full Tags

Displays full tag names without attributes for both *block* and *inline* elements.

Block Tags

Displays full tag names for *block elements* and simple tags without names for *inline elements*.

Inline Tags

Displays full tag names for *inline elements*, while *block elements* are not displayed.

Partial Tags

Displays simple tags without names for *inline elements*, while *block elements* are not displayed.

No Tags

No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Configure Tags Display Mode

Opens the ***Author*** *preferences page* where you can configure options in regards to *tags*, such as the default **Tags Display Mode**, **Tags Background Color**, **Tags Foreground Color**, and **Tags Font**.

Contextual Menu Actions

The following default actions are available in the contextual menu when editing Markdown documents:

Cut, Copy, Paste

Use these actions to execute the typical editing actions on the currently selected content.

Source submenu

This submenu includes the following actions:

To Upper Case

Converts the content selection to upper case characters.

To Lower Case

Converts the content selection to lower case characters.

Capitalize Lines

It capitalizes the first letter found on every new line that is selected. Only the first letter is affected, the rest of the line remains the same. If the first character on the new line is not a letter then no changes are made.

Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on OS X))

Converts a sequence of hexadecimal characters to the corresponding Unicode character. The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed of 2 to 4 hexadecimal characters and may or may not be preceded by the `0x` or `0X` prefix. Examples of valid sequences: `0x0045`, `0X0125`, `1253`, `265`, `43`.

Base64 Encode/Decode submenu

This submenu includes the following actions for encoding or decoding **base64** schemes:

Import File to Encode and Insert

Encodes a file and then inserts the encoded content into the current document at the cursor position.

Decode Selection and Export to File

Decodes a selection of text from the current document and then exports (saves) the result to another file.

Encode Selection

Replaces a selection of text with the result of encoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the ***Encoding for Base64, Base32, Hex conversions*** *option in the* ***Encoding*** *preferences page* will be used. Likewise, the same is true if the ***Show the dialog box for choosing the encoding for Base64, Base 32, Hex conversions*** *option is not selected in the* ***Messages*** *preference page*.

Decode Selection

Replaces a selection of text with the result of decoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the ***Encoding for Base64, Base32, Hex conversions*** *option in the* ***Encoding*** *preferences page* will be used. Likewise, the same is true if the ***Show the dialog box for choosing the encoding for Base64, Base 32, Hex conversions*** *option is not selected in the* ***Messages*** *preference page*.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Base32 Encode/Decode submenu

This submenu includes the following actions for encoding or decoding **base32** schemes:

Import File to Encode and Insert

Encodes a file and then inserts the encoded content into the current document at the cursor position.

Decode Selection and Export to File

Decodes a selection of text from the current document and then exports (saves) the result to another file.

Encode Selection

Replaces a selection of text with the result of encoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the ***Encoding for Base64, Base32, Hex conversions*** *option in the* ***Encoding*** *preferences page* will be used. Likewise, the same is true if the ***Show the dialog box for choosing the encoding for Base64, Base 32, Hex conversions*** *option is not selected in the* ***Messages*** *preference page*.

Decode Selection

Replaces a selection of text with the result of decoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the ***Encoding for Base64, Base32, Hex conversions*** *option in the* ***Encoding*** *preferences page* will be used. Likewise, the same is true if the ***Show the dialog box for choosing the encoding for Base64, Base 32, Hex conversions*** *option is not selected in the* ***Messages*** *preference page*.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Hex Encode/Decode submenu

This submenu includes the following actions for encoding or decoding **hex** schemes:

Import File to Encode and Insert

Encodes a file and then inserts the encoded content into the current document at the cursor position.

Decode Selection and Export to File

Decodes a selection of text from the current document and then exports (saves) the result to another file.

Encode Selection

Replaces a selection of text with the result of encoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the ***Encoding for Base64, Base32, Hex conversions*** *option in the* ***Encoding*** *preferences page* will be used. Likewise, the same is true if the ***Show the dialog box for choosing the encoding for Base64, Base 32, Hex conversions*** *option is not selected in the* ***Messages*** *preference page*.

Decode Selection

Replaces a selection of text with the result of decoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the ***Encoding for Base64, Base32, Hex conversions*** *option in the* ***Encoding*** *preferences page* will be used. Likewise, the same is true if the ***Show the dialog box for choosing the encoding for Base64, Base 32, Hex conversions*** *option is not selected in the* ***Messages*** *preference page*.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Join and Normalize Lines (Ctrl + J (Command + J on OS X))

For the current selection, this action joins the lines by replacing the *line separator* with a single space character. It also normalizes the whitespaces by replacing a sequence of such characters with a single space.

Insert new line after (Ctrl + Alt + Enter (Command + Alt + Enter on OS X))

This action has the same result as moving the cursor to the end of the current line and pressing the *ENTER* key.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Open submenu

The following actions are available in this submenu:

Open File at Cursor

Opens the file at the cursor position in a new panel. If the file path represents a directory path, it will be opened in system file browser. If the file at the specified location does not exist, an error dialog box is displayed and it includes a **Create new file** button that starts the **New document** wizard. This allows you to choose the type or the template for the file. If the action succeeds, the file is created with the referenced location and name and is opened in a new editor panel.

Open File at Cursor in System Application

Opens the file (identified by its link) or web page (identified by a web link) found at cursor position. The target is opened in the default system application associated with that file type.

Compare

Opens the current file in *the **Compare Files** tool*.

Show/Hide Preview

A toggle action that shows or hides the *Preview* pane.

Export as DITA Topic

Converts the current Markdown document into a DITA topic.

Export as HTML

Converts the current Markdown document into an XHTML document.

Print (Available in the *Preview* pane)

Opens a page setup dialog box that allows you to configure printing options for the current document.

Related Information:

Markdown Editor on page 707
Working with Markdown Documents in DITA on page 714
Markdown Editor Syntax Rules and Specifications on page 715

Syntax Highlighting in the Markdown Editor

Oxygen XML Editor supports syntax highlighting in the Markdown editor to improve the readability of the content and make it easier to internalize the semantics of the content.

To customize the colors or styles used for the syntax highlighting colors for Markdown documents, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)***.
2. Go to ***Editor*** > ***Syntax Highlight***.
3. Select and expand the **Markdown** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

Related Information:

Syntax Highlight Preferences on page 104
Markdown Editor on page 707

Automatic Validation in Markdown Documents

Markdown documents are validated automatically as you type. The conversion engine constantly tries to parse your changes to display the results in the *Preview* pane. If a change results in an error that prevents the parser from converting the syntax, an error message will be displayed in the **DITA** tab or in the ***Results** view* at the bottom of the editor.

The types of errors that will be reported include the following:

- Improper order of headers.
- The syntax in a documents begins with something other than a 1st level header.
- Tags are not closed properly if XHTML markup is used in the document.
- Invalid tag names if XHTML markup is used in the document.
- Markup is not well formed if XHTML markup is used in the document.

Related Information:

Markdown Editor on page 707

Working with Markdown Documents in DITA

Oxygen XML Editor includes some unique features that allow you to easily integrate Markdown documents in a DITA project. This is especially helpful for teams that have contributors who are familiar with the Markdown syntax, but they want their output to be generated from DITA projects. The integration between the Markdown editor and DITA includes actions to export or convert Markdown documents to DITA topics and the **DITA** tab in the *Preview* pane provides a visualization of how the topic will look after conversion.

Export Markdown as a DITA Topic

The Markdown editor includes an option to quickly convert the current Markdown document into a DITA topic. The **Export as DITA Topic** action is available in the contextual menu of the left-side text editor and the right-side *Preview* pane when the **DITA** tab selected.

The conversion creates a new XML file that is defined as a DITA topic and opens it in the **Text** editing mode. You can then work with the document as you would with any other DITA topic, although you may need to manually correct some issues where the parser could not properly map Markdown syntax to DITA markup.

Working with Markdown Documents in the DITA Maps Manager

Oxygen XML Editor has some specialized features that allow you to integrate Markdown documents directly into your DITA project using the ***DITA Maps Manager***. The following features are available for Markdown documents in the **DITA Maps Manager** view:

- **Insert Reference to Markdown Document** - You can use the **New**, **Reference**, and **Reference to the currently edited file** actions from the **Append Child**, **Insert Before**, or **Insert After** submenu when invoking the contextual menu in the **DITA Maps Manager** to insert a reference to a Markdown document at the selected location in the map. Markdown documents will be inserted as a topic reference (`topicref` element) with the `format` attribute set to `markdown`.
- **Validate Markdown Documents in DITA Maps** - When you use the **Validate and Check for Completeness** action from the **DITA Maps Manager** toolbar to check the integrity of the structure of a *DITA map*, Markdown documents that are referenced in the *DITA map* will be converted to DITA topics in the background and validated the same as any other DITA topic.
- **Transforming DITA Maps with Markdown Documents** - When transforming *DITA maps* that have Markdown documents referenced, the transformation will convert the Markdown documents to normal DITA output without you needing to manually convert the Markdown documents to DITA topics.
- **Manually Convert Markdown Documents to DITA Topics** - If you need to use DITA semantics that are not possible in Markdown syntax (such as content references, related links, and other DITA-specific syntax), you

can manually convert the Markdown document into a DITA topic. To do so, right-click the Markdown document in the **DITA Maps Manager** and select **Refactoring** > **Convert Markdown to DITA Topic**. This will open a dialog box that allows you to configure options for converting the document to an XML file that is defined as a DITA topic.

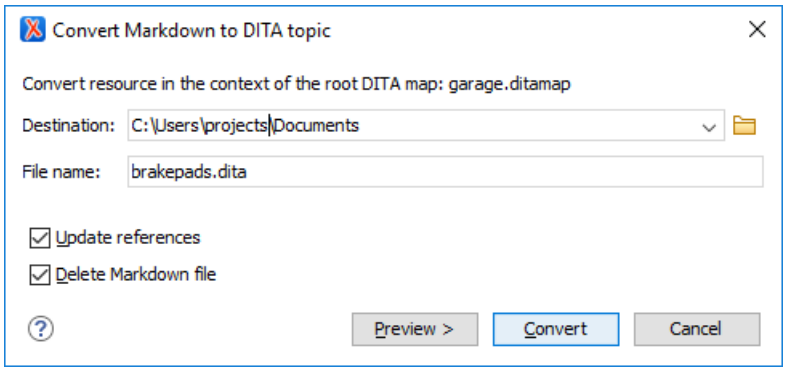


Figure 436: Convert Markdown to DITA Topic Dialog Box

This dialog box includes the following options:

Destination

The destination path for the new DITA topic.

File Name

Presents the current name and allows you to change it.

Update references

Select this option to update all references of the file in the *DITA map* and in the files referenced from the *DITA map*.

Delete Markdown file

If selected, the Markdown version of the file is deleted when the document is converted into a DITA file. If deselected (default value), when the document is converted into a DITA file, the original Markdown file is also preserved in its current location.

Preview

Select this button to display a preview of the changes Oxygen XML Editor is about to make.

Convert

Select this button to perform the conversion.

Tip: Oxygen XML Editor comes with a sample ditamap project for converting Markdown to DITA. Go to the ***Project*** *view*, open the `sample.xpr` project, and navigate to the `dita/markdown-dita` folder.

Related Information:

Markdown Editor on page 707
Actions Available in the Markdown Editor on page 709
Markdown Editor Syntax Rules and Specifications on page 715
Automatic Validation in Markdown Documents on page 714

Markdown Editor Syntax Rules and Specifications

The Markdown editor in Oxygen XML Editor uses rules that were integrated from the most common set of *default Markdown syntax rules* along with many of the *GitHub Flavored Markdown rules*.

The Oxygen XML Editor implementation of the most commonly used syntax rules are as follows:

Headers

The Markdown editor supports two styles of headers, *Setext* and *Atx*.

- **Setext Style**

Setext-style headers are underlined using equal signs (for first-level headers) and dashes (for second-level headers). Any number of equal signs or dashes will result in the same output.

Example: Setext Style Headers

```
First-Level Header (H1)
========

Second-Level Header (H2)
------------
```

- **Atx Style**

  Atx-style headers use 1-6 hash characters at the start of the line, corresponding to header levels 1-6. Optionally, you may close atx-style headers. This is purely cosmetic and the closing hashes do not need to match the number of hashes used to open the header. It is the number of opening hashes that determines the header level.

  Example: Atx Style Headers

```
# H1 text #
## H2 text
### H3 text ######
#### H4 text
##### H5 text ###
###### H6 text
```

Horizontal Rules (for HMTL output only)

You can produce a horizontal rule tag (`<hr />`) by placing three or more hyphens, asterisks, or underscores on a line by themselves (they also need to be preceded and followed by a blank line). Optionally, they can be separated by spaces.

- **Example: Horizontal Rules**

```
* * *

*****

---------------------------------------

_ _ _ _
```

Paragraphs and Line Breaks

A paragraph is simply one or more consecutive lines of text, separated by one or more blank lines. The text at the beginning of a paragraph should not be indented with spaces or tabs. To create a new paragraph, simply insert a blank line in between them.

Important: When converting to HTML, if you break a paragraph on multiple lines (without a blank line in between them), it will create a break tag (`<br />`. When converting to DITA, the text is kept in a single paragraph in this case and a blank line is required to break a paragraph. This behavior differs slightly from the default Markdown rules.

- **Example: Paragraphs**

```
This is a paragraph that contains
two lines of text. (In HTML, a break tag is created in between the two lines)

This is a new paragraph.
```

Styling Text

The Markdown editor supports some syntax rules for styling text (such as bold, italic, or strikethrough).

- **Italic (Emphasis)** - Text wrapped with one asterisk or underscore produces an italic (emphasis) tag.

  ```
  *italic*
  _italic_
  ```
- **Bold (Strong)** - Text wrapped with two asterisks or underscores produces a bold (strong) tag.

  ```
  **bold**
  __bold__
  ```
- **Strikethrough** - In HTML only, text wrapped with two tildes (~~) produces a strikethrough tag.

  ```
  ~~strikethrough~~
  ```

Tip: You can also combine these styling rules. For example, `**BoldText _ItalicText_ BoldText**` would produce italicized text within bold text. Also, if you surround an asterisk or underscore with spaces, it will be treated as a literal asterisk or underscore. To produce a literal asterisk or underscore at a position where it would otherwise be used as an styling delimiter, you can escape it with a backslash (for example, `\*literal asterisks\*`.

Links

The Markdown editor supports two types of links, *inline* and *reference*. In both cases, it begins with link text that is delimited by [square brackets].

- **Inline Links**

  To create an inline link, use a set of regular parentheses immediately after the closing square bracket for the link text. Inside the parentheses, put the URL where you want the link to point, and optionally a title surrounded in quotes. Also, if you referencing a local resource on the same server, you can use relative paths.

  Examples: Inline Link

  With a title:

  ```
  Text with [example link text](http://www.example.com/path "Title") an inline link
   with a title.
  ```

  Without a title:

  ```
  Text with [example link text](http://www.example.com/path) an inline link without
   a title.
  ```

  Relative path:

  ```
  Text with [example link text](/relative_path/) an inline link with relative path.
  ```
- **Reference Links**

  Reference-type links use a second set of square brackets that include a label (link identifier) to identify the link (it may consist of letters, numbers, spaces, and punctuation and it is not case sensitive). You can optionally use a space to separate the sets of brackets. The labels (link identifiers) are only used for creating the links and do not appear in the output.

  ```
  Text with [link text1][id 1] a reference-type link and [link text2][id_2] another
   one.
  ```

  Then, somewhere in the document, you need to define your link label on a line by itself. The link identifier must be within square brackets followed by a colon, then after one or more spaces the URL for the link. Optionally this can be followed by a title enclosed in single quotes, double quotes, or parentheses. Also, the link may optionally be enclosed in angle brackets (`< >`).

  ```
  [id 1]: http://example1.com/ "Optional Title"
  [id_2]: <http://example2.com/> "Optional Title2"
  ```

  Other notes about Reference Links:

- You can put the title on a second line and use extra spaces or tabs for padding. This is useful for aesthetics when the URL is long.

  ```
  [id]: http://example.com/long/path/to/resource/here
      "Optional Title Here"
  ```
- The label (link identifier) can be missing, in which case the link text (in square brackets) is used as the name.

  ```
  [My Link][]
  ```

  and then defined as:

  ```
  [My Link]: http://example.com/
  ```

Automatic Links

The Markdown editor supports a shortcut style for creating automatic links for URLs and email addresses. You simply surround the URL or email address with angle brackets.

Note: These automatic links only work properly in HTML conversions. The *Preview* pane may display them properly in the DITA tab, but the DITA output will not properly recognize the format.

- **URLs**

  By surrounding a URL with angle brackets, you can show the actual text of the URL while also making it clickable in the output.

  ```
  <http://example.com/>
  ```

  For example, in HTML it is converted to:

  ```
  <a href="http://example.com/">http://example.com/</a>
  ```
- **Email Addresses**

  Automatic links for email addresses work similarly, except that Markdown will also perform a bit of randomized decimal and hex entity-encoding to help obscure your address from address-harvesting *spambots*.

  ```
  <address@example.com>
  ```

  In HTML, it is converted to something like:

  ```
  <a href="&#x6D;&#x61;i&#x6C;&#x74;&#x6F;:&#x61;&#x64;&#x64;&#x72;&#x65;
  &#115;&#115;&#64;&#101;&#120;&#x61;&#109;&#x70;&#x6C;e&#x2E;&#99;&#111;
  &#109;">&#x61;&#x64;&#x64;&#x72;&#x65;&#115;&#115;&#64;&#101;&#120;&#x61;
  &#109;&#x70;&#x6C;e&#x2E;&#99;&#111;&#109;</a>
  ```

Images

The Markdown editor uses an image syntax that is intended to resemble the syntax for links, allowing for the same two types: *inline* and *reference*. In both cases, the syntax for images begin with an exclamation mark, followed by `alt` attribute text surrounded by square brackets., and then followed by a set of parentheses that contain the URL or path to the image.

- **Inline Images**

  For inline images, use a set of regular parentheses immediately after the closing square bracket for the `alt` attribute text. Inside the parentheses, put the URL or path of the image, and optionally a title surrounded in quotes.

  Examples: Inline Images

  With a title:

```
Text with ![Alt text](/path/to/img.jpg "Optional title") an inline image with a
 title.
```

Without a title:

```
Text with ![Alt text](/path/to/img.jpg) an inline link without a title.
```

- **Reference Images**

  For reference-type images, use a second set of square brackets that include a label (image identifier) to identify the image (it may consist of letters, numbers, spaces, and punctuation and it is not case sensitive). You can optionally use a space to separate the sets of brackets. The labels (image identifiers) do not appear in the output.

```
Text with ![Alt text1][id] a reference-type image.
```

  Then, somewhere in the document, you need to define your image label on a line by itself. The image identifier must be within square brackets followed by a colon, then after one or more spaces the URL or path of the image. Optionally this can be followed by a title enclosed in single quotes, double quotes, or parentheses.

```
[id]: url/to/image "Optional Title"
```

Blockquotes

The Markdown editor uses email-style greater than characters (>) for *blockquotes*. You only need to put the > before the first line of a hard-wrapped paragraph, but it looks better (and is more clear) if you put a > before every line.

- **Example: Blockquotes**

```
> This is a blockquote with two paragraphs. Lorem ipsum dolor sit amet,
> consectetuer adipiscing elit. Aliquam hendrerit mi posuere lectus.
> Vestibulum enim wisi, viverra nec, fringilla in, laoreet vitae, risus.
>
> Donec sit amet nisl. Aliquam semper ipsum sit amet velit. Suspendisse
> id sem consectetuer libero luctus adipiscing.
```

- *Blockquotes* can be nested by adding additional levels of > characters.

  Example: Nested Blockquotes

```
> This is the first level of quoting.
>
> > This is nested blockquote.
>
> Back to the first level.
```

- *Blockquotes* can also contain other Markdown elements (such as headers, lists, and code blocks).

  Example: Blockquotes with Other Markdown Elements

```
> ## This is a header.
>
> 1.   This is the first list item.
> 2.   This is the second list item.
>
> Here's some example code:
>
>     return shell_exec("echo $input | $markdown_script")
```

Quoting Code (Inline and Code Blocks)

The Markdown editor supports quoting code or commands inline within a sentence or in distinct blocks.

- **Inline**

You can quote or emphasize code within a sentence (inline) with single backticks (`). The text within the backticks will not be formatted.

Example: Inline Code Emphasis

```
This is a normal sentence with a `code` in the middle.
```

- **Code Blocks**

  You can format code or text into its own distinct block by inserting a blank line before and after the content and using at least 4 spaces (or 1 tab), or by using opening and closing triple backticks (```) on separate lines.

  Example: Code Block

  ```
  This is a normal paragraph:

      This is a code block

  This is a normal paragraph:

  ```
  This is a code block
  ```
  ```

  One level of indentation is removed from each line of a codeblock and it continues until it reaches a line that is not indented (or until the closing backticks).

  Example: Code Block with Indentation

  ```
      tell application "something"
          beep
      end tell
  ```

  For example, in HTML the result would look like this:

  ```
  <pre><code>tell application "Foo"
      beep
  end tell
  </code></pre>
  ```

  You can also add an optional language identifier to enable syntax highlighting in your code blocks. The Oxygen XML Editor Markdown editor supports the following languages: *Java*, *JavaScript*, *CSS*, and *Python*.

  Example: Syntax Highlighting in Code Block

  ```
  ```css
  input[type="submit"] {
      color: white;
      font-weight: bold;
  ```
  ```

Inline XHTML (for HMTL output only)

The Markdown editor supports writing inline XHTML. Since Markdown is just a writing format, it requires a conversion for publishing purposes. If you are using the HTML conversion, for any markup that is not covered by Markdown syntax, you can simply use XHTML syntax.

- **Example: Inline XHTML**

  ```
  This is a regular paragraph.

  <table>
      <tr>
          <td>Col 1</td>
          <td>Col 2</td>
      </tr>
  </table>
  ```

```
This is another regular paragraph.
```

Lists

The Markdown editor supports ordered and unordered lists. You can also insert *blockquotes* and *code blocks* inside list items. List markers typically start at the left margin, but may be indented by up to three spaces.

- **Unordered Lists**

  For unordered lists, you can use asterisks (*), plus signs (+), and hyphens (-) interchangeably.

  ```
  * List item 1
  + List item 2
  - List item 3
  ```

- **Ordered Lists**

  For ordered lists, use numbers followed by periods. The actual numbers you use have no effect on the output. It simply converts them to list items within an ordered list an the actual number of list items will determine the numbers in the output.

  ```
  1. List item 1
  8. List item 2
  5. List item 3
  ```

- **Nested Lists**

  You can create nested lists by indenting lines by two spaces.

  ```
  1. Ordered list item 1
    1. Nested ordered list item 1
    2. Nested ordered list item 2
      * 2nd level nested unordered list item 1
      * 2nd level nested unordered list item 2
        * 3rd level nested unordered list item 1
  2. Ordered list item 2
  ```

- **Paragraphs Inside Lists**

  If list items are separated by blank lines, Markdown will wrap the items in a paragraph in the output.

  ```
  * List item 1

  * List item 2
  ```

  For both DITA and HTML output, this would result in:

  ```
  <ul>
  <li><p>List item 1</p></li>
  <li><p>List item 2</p></li>
  </ul>
  ```

- **Multiple Paragraphs Inside Lists**

  List items may consist of multiple paragraphs. Each subsequent paragraph in a list item must be indented by either 4 spaces or one tab. Optionally, you can also indent each line of a paragraph to make it look nicer.

  ```
  1.  This is a list item with two paragraphs. Lorem ipsum dolor
      sit amet, consectetuer adipiscing elit. Aliquam hendrerit
      mi posuere lectus.

      Vestibulum enim wisi, viverra nec, fringilla in, laoreet
      vitae, risus. Donec sit amet nisl. Aliquam semper ipsum
      sit amet velit.

  2.  Suspendisse id sem consectetuer libero luctus adipiscing.
  ```

- **Blockquotes Inside Lists**

To put a *blockquote* within a list item, the blockquote delimiters (>) need to be indented so that it is under the first letter of the text after the list item marker.

```
*   A list item with a blockquote:
    > This is a blockquote
    > inside a list item.
```

- **Code Blocks Inside Lists**

  To put a code block within a list item, insert an empty line in between the list item and the code block, and the code block needs to be indented twice (with 8 spaces or 2 tabs), or if you are using the triple backticks method, the opening triple backtick needs to be indented with 4 spaces or 1 tab.

```
*   A list item with a code block:

        This is a code block inside a list item

    ```
    This is a code block inside a list item using the backticks method
    ```
```

Task Lists

You can create task lists by prefacing list items with a hyphen followed by a space followed by square brackets (`- [ ]`). To mark a task as complete, use `- [x]`.

- **Example: Task Lists**

```
- [ ] Unfinished task 1
- [x] Finished task 2
```

Definition Lists

You can create definition lists by using a colon plus a space for each list item.

- **Example: Definition Lists**

```
Term 1
: Definition A
: Definition B
```

Tables

You can create tables in the Markdown editor by using pipes (`|`) and hyphens (`-`).

- **Creating a Table**

  Pipes are used to separate each column, while hyphens are used to create column headers. The pipes on either end of the table are optional. Cells can vary in width and do not need to be perfectly aligned within columns, but there must be at least three hyphens in each column of the header row.

```
| First Header  | Second Header |
| ------------- | ------------- |
| Column 1 Row 1 Cell  | Column 2 Row 1 Cell  |
| Column 1 Row 2 Cell  | Column 2 Row 2 Cell  |
```

- **Formatting Rules in Table Cells**

  You can use formatting rules inside the cells of the table (such as links, inline code blocks, and text styling).

```
| First Header  | Second Header |
| --- | --- |
| `inline code`  | Content with **bold text** inside cell  |
```

- **Aligning Text in Tables**

You can align text to the left, right, or center of a column by including colons (`:`) to the left, right, or on both sides of the hyphens within the header row.

```
| Left-aligned | Center-aligned | Right-aligned |
| :---         |     :---:      |          ---: |
| Content Cell | Content Cell   | Content Cell  |
```

- **Joining Cells (Span a Cell Over Multiple Columns)**

  You can join cells horizontally (span a cell over multiple columns) by using multiple consecutive pipe characters (`|`) to the right of the particular cell. The number of consecutive pipes indicate the number of columns the cell will span (`||` for two, `|||` for three, and so on).

```
| First Header | Second Header | Third Header | Fourth Header |
| ------------ | ------------- | ------------ | ------------- |
| Content Cell | *Cell Span Over 3 Columns* |||
```

Emoji

You can add *emoji* in the Markdown editor by surrounding the EMOJICODE with colons (`:EMOJICODE:`).

- **Example: Emoji**

```
:smile:
:laughing:
```

  The resulting emoticons will appear in the output, but they are not displayed in the *Preview* pane.

  For a full list of available emoji codes, see *Emoji Cheat Sheet*.

Backslash Escapes

You can ignore Markdown formatting by using backslash escapes (`\`) to generate literal characters that would otherwise have special meaning in the Markdown syntax. For example, if you want to surround a word with literal asterisks (instead of an italic or emphasis tag), you can use backslashes to escape the asterisks.

```
\*literal asterisks\*
```

The Markdown editor provides backslash escapes for the following characters:

```
\   backslash
`   backtick
*   asterisk
_   underscore
{}  curly braces
[]  square brackets
()  parentheses
#   hash mark
+   plus sign
-   minus sign (hyphen)
.   dot
!   exclamation mark
```

Automatic Escaping for Special Characters

The Markdown editor includes support for automatically escaping special characters such as angle brackets (< >) and ampersands (&). If you want to use them as literal characters, you must escape them as entities, as in the table below. The exception to this is in HTML output, if the special characters for a valid tag (for example, <b>), they are treated as literal characters and escaping is not necessary.

Literal Character	Escaping Code
<	**<**
>	**>**

Literal Character	Escaping Code
&	&

Footnotes

The Markdown editor in Oxygen XML Editor supports normal and inline footnotes. The following examples show the required syntax.

- **Example: Normal Footnote**

```
Here is a footnote reference,[^1]

[^1]: Here is the footnote.
```

- **Example: Normal Footnote with Multiple Blocks**

```
Here is a footnote reference,[^longnote]

[^longnote]: Here is the footnote with multiple blocks.

    Subsequent paragraphs are indented with 4 spaces or 1 tab to show that they
belong to the previous footnote.
```

- **Example: Inline Footnote**

```
Here is an inline note.^[Inlines notes are easier to write, since
you don't have to pick an identifier and move down to type the
note.]
```

Related Information:

Default Markdown Syntax

GitHub Flavored Markdown Rules

Markdown Editor on page 707

Actions Available in the Markdown Editor on page 709

Editing Non-XML Files

While Oxygen XML Editor specializes in XML-related technologies, you can also use it to create and edit various types of non-XML files. Non-XML files are opened in a simple text editor and many of the helpful features that are commonly used when editing XML files in the Oxygen XML Editor **Text** editing mode are available in this simple editor.

Types of Non-XML Files That are Supported

Some of the types of non-XML files that can be created and edited in Oxygen XML Editor include:

- *Markdown*
- Java
- C++
- C
- PHP
- Perl
- Properties
- SQL
- Shell executables
- Batch
- Python
- Text

Features Available in the Simple Text Editor

When editing non-XML files in the simple text editor, the features that are available include the following:

- **Project Support** - The unique *features that are designed to help you work with projects* are available for all types of files.
- **Shortcut Actions** - Many of the *shortcut actions that are available in **Text** mode* are also available in the simple text editor.
- **Drag and Drop** - The normal drag and drop support is available in the simple text editor.
- **Content Selection Features** - The *content selection shortcuts* that are available in **Text** mode (including the *Rectangular Selection* feature) are also available in the simple text editor.
- **Bookmarks** - You can use *bookmarks to mark positions* in any type of file so that you can return to it later.
- **Convert Hexadecimal Characters** - You can *convert a sequence of hexadecimal characters to the corresponding Unicode character*.
- **Encoding/Decoding Actions** - Contextual menu actions are available to *encode or decode Base64, Base 32, and Hex schemes*.
- **Code Templates** - You can define your own *code templates* for any type of file and use the *Content Completion Assistant* to invoke them.
- **Syntax Highlighting** - Non-XML files also support syntax highlighting with dedicated coloring schemes. To customize them, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to ***Editor** > **Syntax Highlight***. Select and expand the appropriate section in the top pane for the type of file you are editing and you can see the effects of your changes in the **Preview** pane.
- **Find/Replace** - You can use the ***Find/Replace** action* to find or replace all the occurrences of a word or string of characters in any type of file that you are editing.
- **File Comparison Tool** - The ***Compare Files** tool* can also be used to compare non-XML files.

Spell Checking

Oxygen XML Editor includes an *automatic (as-you-type) spell checking feature*, as well as a manual spell checking action to open a **Spelling** dialog box that offers a variety of options.

To manually check spelling in the current document, use the **Check Spelling** action on the toolbar or from the **Edit** menu.

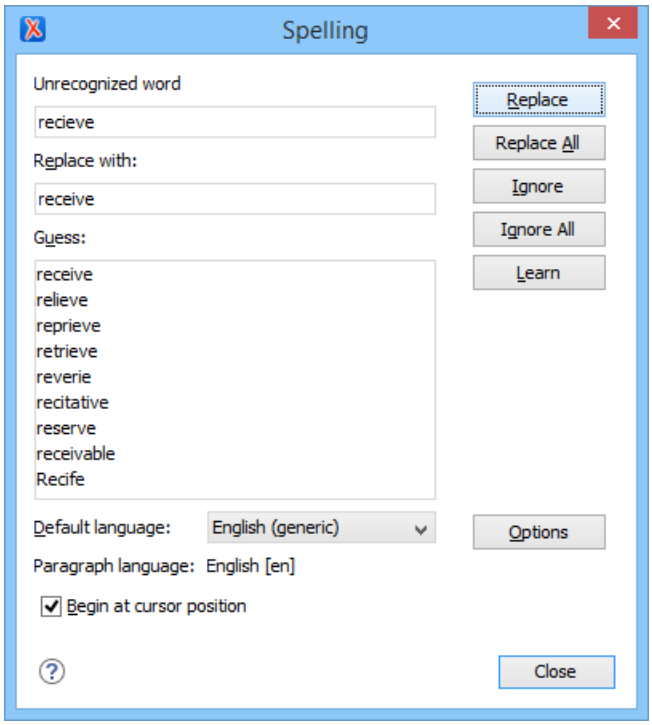


Figure 437: Check Spelling Dialog Box

The **Spelling** dialog box contains the following:

Unrecognized word

Displays the word that cannot be found in the selected dictionary. The word is also highlighted in the XML document.

Replace with

The character string that will replace the misspelled word.

Guess

Displays a list of words suggested to replace the unknown word. Double-click a word to automatically insert it in the document and resume the spell checking process.

Default language

Allows you to select the default language dictionary used by the spelling engine.

Paragraph language

In an XML document, you can mix content written in multiple languages. You can set the language code in the `lang` or `xml:lang` attribute for any particular section and Oxygen XML Editor will automatically instruct the spell checker engine to apply the appropriate language dictionary for that section.

Begin at cursor position

Instructs the spell checker to begin checking the document starting from the current cursor position.

Action Buttons

Replace

Use this button to replace the unrecognized word with the selected word from the **Replace with** field.

Replace All

Use this button to replace all occurrences of the unrecognized word with the selected word from the **Replace with** field.

Ignore

Ignores the first occurrence of the unrecognized word and allows you to continue checking the document. Oxygen XML Editor skips the content of the XML elements *marked to be ignored*.

Ignore All

Ignores all instances of the unrecognized word in the current document.

Learn

Adds the unrecognized word to the list of valid words.

Options

Opens the ***Spell Check*** *preferences page* where you can configure various options in regards to the feature.

Related Information:

AutoCorrect Misspelled Words on page 733

Spell Check Dictionaries and Term Lists

Oxygen XML Editor uses the **Hunspell** engine for the spell checking feature. The Hunspell spell checking engine is open source and has an LGPL license. It is designed for languages with rich morphology and complex compounding or character encoding. Each language-country variant combination have their own specific dictionaries. Oxygen XML Editor includes the following built-in dictionaries for the spell checker:

- English (US) [`en_US`]
- English (UK) [`en_GB`]
- French [`fr`]
- German [`de_DE`]
- Spanish [`es_ES`]

Other Hunspell Dictionaries

You can also download Hunspell dictionaries for other languages and add them to the Oxygen XML Editor spell checker. An example of a website that includes numerous dictionary files is: *http://extensions.services.openoffice.org/dictionary*.

If you cannot find a Hunspell dictionary that is already built for your language, you can build the dictionary you need. To build a full Hunspell dictionary, follow *these instructions* and then add the dictionary to the Oxygen XML Editor spell checker by following *this procedure*.

Personalized Term Lists

Authoring in certain areas of expertise (for example, the pharmaceutical or automobile industries) might require the use of specific terms that are not part of the standard spell checker dictionary. To avoid marking these terms as errors, Oxygen XML Editor provides a way of *adding personalized term lists* to the spell check engine. This involves creating a term list file that the spell checker will recognize and it is similar to the file Oxygen XML Editor uses for storing *learned words*.

The term list files are specific for each language and can be specific to each domain or area of expertise (for example, *legal*, *medical*, *automotive*). They can also be used to control forbidden words.

Related Information:

Adding Spell Check Dictionaries on page 728
Adding Spell Check Term Lists on page 729
Building and Testing Hunspell Dictionaries

Adding Custom Dictionaries and Term Lists

The Oxygen XML Editor spell checker allows you to add customized Hunspell dictionaries and personalized term lists. The Hunspell dictionary mechanism requires a dictionary file (with a `.dic` file extension) and an affix file (with an `.aff` file extension). The personalized term lists are custom files (with a `.tdi` file extension) that you can create to include specialized terms or specify forbidden words in the Oxygen XML Editor spell checker.

You can *add dictionaries* and *personalized term lists* to the default folder where they are stored or specify your own custom locations. You can view the default storage location in the ***Spell Check Dictionaries*** *preferences page* and the ***Include dictionaries and term list from*** *option* allows you to choose a custom storage location. All the dictionaries and term lists for a particular language that are found in either location are merged and used by the spell checker in Oxygen XML Editor.

Related Information:
Replacing a Spell Check Dictionary on page 730

Adding Spell Check Dictionaries

There are three possible scenarios for adding Hunspell dictionaries to the Oxygen XML Editor spell checker:

- You can download a pre-built Hunspell dictionary and add it to the spell checking mechanism.
- You can create a custom Hunspell dictionary file that defines your own list of words and add it to the spell checking mechanism.
- You can build your own full Hunspell dictionary and add it to the spell checking mechanism.

Download and Add a Pre-Built Hunspell Dictionary

To add a downloaded pre-built dictionary, follow these steps:

1. Download the files needed for your dictionary. You will need a *dictionary* file (with a `.dic` file extension) and an *affix* file (with an `.aff` file extension). If the dictionary does not include an affix file (`.aff`), you can create one and leave it empty, but it is needed for the mechanism to work properly. An example of a website that includes numerous dictionary files is: *http://extensions.services.openoffice.org/dictionary*.

   Important: The name of the files should begin with a two letter prefix for the language code, followed by an underscore or hyphen, then two letters that indicate the country code, followed by another underscore or hyphen, and then a descriptive name (for example, `en_US_medical.dic` for a medical dictionary in the US version of the English language, or for a less specific English medical dictionary, you could omit the country code like this: `en_medical.dic`). For a list of language codes, see *https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes*.
2. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to ***Editor** > **Spell Check** > **Dictionaries***.
3. Choose one of the following two options for adding the downloaded files.
   a. Copy both files (`.dic` and `.aff`) to the default directory displayed in the ***Dictionaries and term lists default folder** option*.
   b. Copy both files (`.dic` and `.aff`) to any other directory, select the ***Include dictionaries and term list from** option*, and select that directory. If you choose this option, make sure you read *this important note*.
4. Restart the application for the spell checker to start using the new dictionary.

Create a Custom Hunspell Dictionary that Defines a List of Words

To create a custom Hunspell dictionary that defines your own list of words, follow these steps:

1. Create a *dictionary* file (with a `.dic` file extension) and an *affix* file (with an `.aff` file extension). The affix file (`.aff`) can be left empty, but it is needed for the mechanism to work properly.

   Important: The name of the files should begin with a two letter prefix for the language code, followed by an underscore or hyphen, then two letters that indicate the country code, followed by another underscore or hyphen, and then a descriptive name (for example, `en_US_medical.dic` for a medical dictionary in the US version of the English language, or for a less specific English medical dictionary, you could omit the country code like this: `en_medical.dic`). For a list of language codes, see *https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes*.
2. In the dictionary file (`.dic` extension), add the words you want to be included in your custom dictionary. Add one word per row and the first line needs to contain the number of words, as in the following example:

   ```
   2
   parabola
   asimptotic
   ```
3. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to ***Editor** > **Spell Check** > **Dictionaries***.
4. Choose one of the following two options for saving the files.
   a. Save both files (`.dic` and `.aff`) to the default directory displayed in the ***Dictionaries and term lists default folder** option*.
   b. Save both files (`.dic` and `.aff`) to any other directory, select the ***Include dictionaries and term list from** option*, and select that directory. If you choose this option, make sure you read *this important note*.

5. Restart the application for the spell checker to start using the new dictionary.

Build and Add a Full Hunspell Dictionary

To build and add a full Hunspell dictionary, follow these steps:

1. Follow these instructions: *Building and Testing Hunspell Dictionaries*.

   Result: You should end up with a *dictionary* file (with a `.dic` file extension) and an *affix* file (with an `.aff` file extension). The affix file (`.aff`) can be empty, but it is needed for the mechanism to work properly.

   Important: The name of the files should begin with a two letter prefix for the language code, followed by an underscore or hyphen, then two letters that indicate the country code, followed by another underscore or hyphen, and then a descriptive name (for example, `en_US_medical.dic` for a medical dictionary in the US version of the English language, or for a less specific English medical dictionary, you could omit the country code like this: `en_medical.dic`). For a list of language codes, see *https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes*.
2. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to ***Editor** > **Spell Check** > **Dictionaries***.
3. Choose one of the following two options for saving the files.
   a. Save both files (`.dic` and `.aff`) to the default directory displayed in the ***Dictionaries and term lists default folder** option*.
   b. Save both files (`.dic` and `.aff`) to any other directory, select the ***Include dictionaries and term list from** option*, and select that directory. If you choose this option, make sure you read *this important note*.
4. Restart the application for the spell checker to start using the new dictionary.

Related Information:

Adding Spell Check Term Lists on page 729

Adding Spell Check Term Lists

You can create personalized term lists that are used to store specialized terms or control forbidden words. They can then be added to one of the directories that store the spell check dictionaries and the spell checker will be merge them with all the dictionaries and other term lists for a particular language.

Create and Add Personalized Term Lists

To create and add a personalized term list, follow these steps:

1. Create a *term list* file (with a `.tdi` file extension). The name of the file must begin with a two letter prefix that indicates the language it should be attached to, followed by an underscore or hyphen, and then a descriptive name (for example, `en_US_myterms.tdi` for term list in the US version of the English language or `en_myterms.tdi` for a less specific English term list). For a list of language codes, see *https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes*.
2. In the term list file (`.tdi` extension), add the terms you want to be included in your custom dictionary. If you need to specify forbidden terms, those words simply need to be preceded by an asterisk. Add one word per row, as in the following example:

```
parabola
asimptotic
*hyperbola
```

3. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to ***Editor** > **Spell Check** > **Dictionaries***.
4. Choose one of the following two options for saving the file.
   a. Save the file (`.tdi`) to the default directory displayed in the ***Dictionaries and term lists default folder** option*.
   b. Save the file (`.tdi`) to any other directory, select the ***Include dictionaries and term list from** option*, and select that directory. If you choose this option, make sure you read *this important note*.
5. Restart the application for the spell checker to start using the new term list.

Related Information:

Adding Spell Check Dictionaries on page 728

Replacing a Spell Check Dictionary

There are several possible scenarios for replacing an existing Hunspell dictionary for the Oxygen XML Editor spell checker:

- You can download a pre-built Hunspell dictionary and replace an existing dictionary with it.
- You can build your own full Hunspell dictionary and replace an existing dictionary with it.

Download a Pre-Built Hunspell Dictionary and Replace an Existing One

To replace an existing dictionary with a downloaded pre-built dictionary, follow these steps:

1. Download the files needed for your dictionary. You will need a *dictionary* file (with a `.dic` file extension) and an *affix* file (with an `.aff` file extension). If the dictionary does not include an affix file (`.aff`), you can create one and leave it empty, but it is needed for the mechanism to work properly. An example of a website that includes numerous dictionary files is: *http://extensions.services.openoffice.org/dictionary*.
2. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to ***Editor** > **Spell Check** > **Dictionaries***.
3. Choose one of the following two options to replace existing files.
   a. Replace the existing files (`.dic` and `.aff`) for the particular language in the default directory displayed in the ***Dictionaries and term lists default folder** option*. Leave the **Include dictionaries and term list from** option deselected.
   b. Replace existing files (`.dic` and `.aff`) for the particular language in a directory specified in the ***Include dictionaries and term list from** option*. If you choose this option, make sure you read *this important note*.

   Important: Do not alter the naming convention. The name of the files must begin with a two letter prefix that indicates the language it should be attached to (for example, `en_US.dic` for a US English dictionary or `en.dic` for a less specific English dictionary). For a list of language codes, see *https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes*.
4. Restart the application for the spell checker to start using the new dictionary.

Build a Full Hunspell Dictionary and Replace an Existing One

To replace an existing dictionary with a full Hunspell dictionary that you build, follow these steps:

1. Follow these instructions: *Building and Testing Hunspell Dictionaries*.

   Result: You should end up with a *dictionary* file (with a `.dic` file extension) and an *affix* file (with an `.aff` file extension). The affix file (`.aff`) can be empty, but it is needed for the mechanism to work properly.
2. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to ***Editor** > **Spell Check** > **Dictionaries***.
3. Choose one of the following two options to replace existing files.
   a. Replace the existing files (`.dic` and `.aff`) for the particular language in the default directory displayed in the ***Dictionaries and term lists default folder** option*. Leave the **Include dictionaries and term list from** option deselected.
   b. Replace existing files (`.dic` and `.aff`) for the particular language in a directory specified in the ***Include dictionaries and term list from** option*. If you choose this option, make sure you read *this important note*.
4. Restart the application for the spell checker to start using the new dictionary.

Related Information:

Adding Custom Dictionaries and Term Lists on page 727

Learned Words

Spell checker engines rely on dictionaries to decide if a word is spelled correctly. To instruct the spell checker engine that an unknown word is actually correctly spelled, you need to add that word to a list of learned words. There are two ways to do this:

- Invoke the contextual menu on an unknown word, then select **Learn word**.
- Press the **Learn** button from the ***Spelling** dialog box* that is invoked by using the **Check Spelling** action on the toolbar.

Note: To delete items from the list of learned words, use the **Delete learned words** option in the ***Editor*** > ***Spell Check*** > ***Dictionaries*** *preferences page*.

Ignored Words (Elements)

There are certain XML elements (such as `programlisting`, `codeblock`, or `screen`) in which you may want the content to always be skipped during the spell check process. This can be done in one of several ways:

- You can skip through them manually, word by word, using the **Ignore** button in *the* ***Spelling*** *dialog box* that is invoked by using the **Check Spelling** action on the toolbar.
- You can automatically skip the content of certain elements by maintaining a set of known element names that should never be checked. You can manage this set of element names by using the ***Ignore elements*** *section* in the **Spell Check** preferences page.

Automatic Spell Check

Oxygen XML Editor includes an option to automatically check the spelling as you type. Not only does it check spelling when you are typing in the main editor, but also when you are typing in a *comment*. This feature is disabled by default, but it can be enabled and configured in the ***Spell Check*** *preferences page*. When the ***Automatic Spell Check*** *option* is selected, unknown words are underlined and some actions are available in the contextual menu to help you correct the word or prevent the word from being reported in the future.

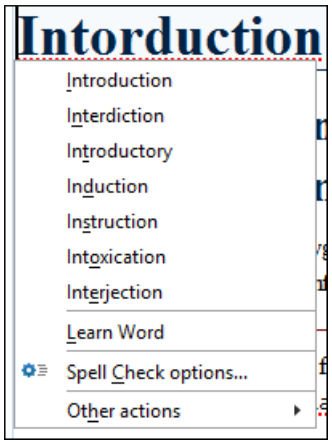


Figure 438: Automatic Spell Checking in Author Mode

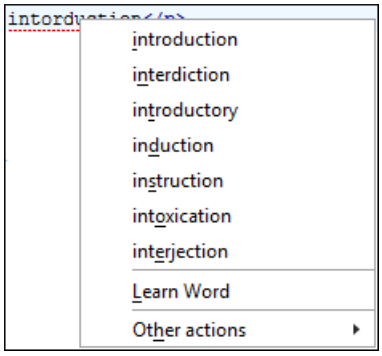


Figure 439: Automatic Spell Checking in Text Mode

The contextual menu includes the following actions:

Delete Repeated Word

Allows you to delete words that were repeated in consecutive order.

List of Suggestions

A list of words suggested by the spell checking engine as possible replacements for the unknown word.

Learn Word

Allows you to add the current unknown word to the persistent dictionary of *learned words*.

Spell check options (Available in Author mode only)

Opens the ***Spell Check*** *preferences page*.

Other actions

This submenu give you access to all the usual contextual menu actions.

Related Information:

Learned Words on page 730

Spell Check Multiple Files

The **Check Spelling in Files** action allows you to check the spelling on multiple local or remote documents. This action is available in the following locations:

- The **Edit** menu.
- The contextual menu of the ***Project*** *view*.
- When editing DITA documents, the contextual menu of the ***DITA Maps Manager*** *view*.

This action opens the **Check Spelling in Files** dialog box that allows you to define the scope and several other options. After you configure the settings for the operation, click the **Check All** button to check the spelling in all specified files. The spelling corrections are displayed in the ***Results*** *view* at the bottom of the editor and you can group the reported errors as a tree with two levels.

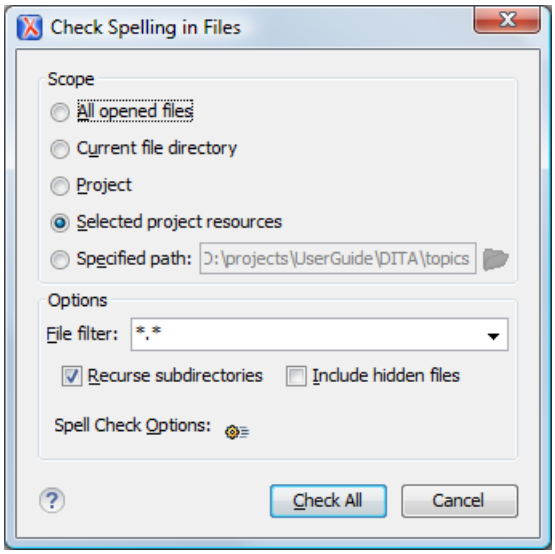


Figure 440: Check Spelling in Files Dialog Box

The following scopes are available:

- **All opened files** - The spell check is performed in all opened files.
- **Directory of the current file** - All the files in the folder of the current edited file.
- **Project files** - All files from the current project.
- **Selected project files** - The selected files from the current project.
- **Specified path** - Checks the spelling in the files located at a path that you specify.

The **Options** section includes the following options:

- **File filter** - Allow you to filter the files from the selected scope.
- **Recurse subdirectories** - When selected, the spell check is performed recursively for the specified scope. The one exception is that this option is ignored if the scope is set to **All opened files**.
- **Include hidden files** - When selected, the spell check is also performed in the hidden files.

- **Spell Check Options** - The spell check processor uses the options available in the ***Spell Check*** *preferences panel* .

When working with DITA documents, if you invoke the **Check Spelling in Files** action in the ***DITA Maps Manager*** *view*, a slightly different version of the dialog box is displayed:

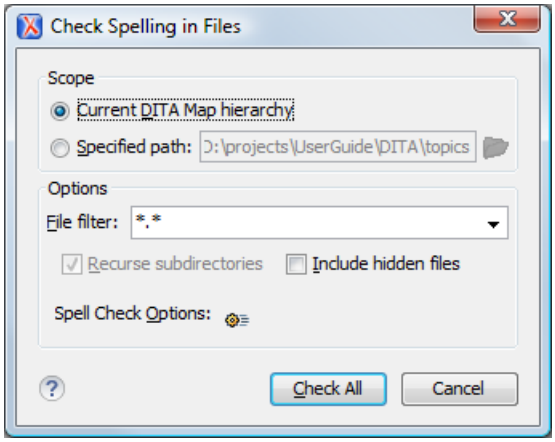


Figure 441: Check Spelling in Files Dialog Box (Invoked from the DITA Maps Manager View)

The following scopes are available when you check the spelling in files from the ***DITA Maps Manager***:

- **Current DITA Map hierarchy** - All the files referenced in the currently selected *DITA map*, opened in the **DITA Maps Manager** view.
- **Specified path** - Checks the spelling in the files located at a path that you specify.

AutoCorrect Misspelled Words

Oxygen XML Editor includes an *AutoCorrect* feature to automatically correct misspelled words, as well as to insert certain symbols or other text, as you type in **Author** mode. Oxygen XML Editor includes a default list of commonly misspelled words and symbols, but you can modify the list to suit your needs. You can also choose to have the *AutoCorrect* feature use suggestions from the main spell checker. The suggestions will only be used if the misspelled words are not found in the *Replacements Table*.

When enabled, the *AutoCorrect* feature can be used to do the following:

- Automatically correct misspelled words while you edit in **Author** mode.
- Easily insert symbols. For example, if you want to insert a ® character, you would type `(R)`.
- Quickly insert text fragments.

AutoCorrect is enabled by default. To configure this feature, *open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Editor** > **Edit Modes** > **Author** > **AutoCorrect**.

The actual operation of replacing a word is triggered by a space, dash, or certain punctuation characters (`, . ; : ? ! ' " ) ] }`). After a correction, the affected string is highlighted. The highlight is removed upon the next editing action (text insertion or deletion). If you hover over the highlight, a small widget appears below the word. If you hover over the widget, it expands and you can click it to present a drop-down list that includes the following options:

- **Change back to "[original word]"** - Reverts the correction back to its original form.
- **Stop Automatically Correcting "[original word]"** - This option is presented if the correction is performed based on the *AutoCorrect Replacements Table* and selecting it will delete the corresponding entry from the Replacements Table.
- **Learn Word "[original word]"** - This option is presented if the ***Use additional suggestions from the spell checker*** *option* is selected in the ***AutoCorrect*** *preferences page* and the correction is performed based on the *Spell Checker*. Selecting this option will add the item to the list of *learned words*.
- **AutoCorrect options** - Opens the ***AutoCorrect*** *preferences page* that allows you to configure the feature.

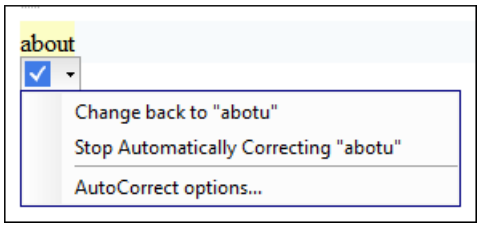


Figure 442: AutoCorrect Widget

The *AutoCorrect* feature results in the following types of substitutions in regards to case-sensitivity:

- Words with all lower-case characters will be replaced with lower-case substitutions (for example, "abotu" is replaced with "about").
- Words with irregular-case characters will be replaced with lower-case substitutions ("ABotU" is replaced with "about").
- Words with all upper-case characters will be replaced with upper-case substitutions ("ABOTU" is replaced with "ABOUT").
- Words starting with an upper-case character will be replaced with substitutions having the same pattern ("Abotu" is replaced with "About").

The *AutoCorrect* feature also uses the list of ***ignored elements from the Spell Check preferences page***. All elements (along with their descendant elements) included in this list will be ignored by the *AutoCorrect* engine.

Related Information:

Spell Checking on page 725

Add Dictionaries for the AutoCorrect Feature

To add new dictionaries for the *AutoCorrect mechanism*, or to replace an existing one, follow these steps:

1. Download an *AutoCorrect* dictionary file for the desired language. The file needs to have a `.dat` file extension. An example of a website that includes some *AutoCorrect* dictionary files is: *OpenOffice Extensions Search Page*.
2. *Open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to ***Editor*** > ***Edit Modes*** > ***Author*** > ***AutoCorrect*** > ***Dictionaries***.
3. Choose one of the following two options for adding the downloaded files.
   a. Copy the downloaded `.dat` file to the default directory displayed in the ***Dictionaries default folder*** *option.*. Note that if you are replacing an existing dictionary file, this is the best option.
   b. Copy the downloaded `.dat` file to any other directory, select the ***Include dictionaries from*** *option*, and select that directory. If you choose this option, make sure you read *this important note*.
4. Restart the application for the *AutoCorrect* mechanism to start using the new dictionary.

Loading Large Documents

When you open a document with a file size larger than the limit configured in ***Open/Save*** *preferences*, Oxygen XML Editor prompts you to choose whether you want to optimize the loading of the document for large files or for huge files.

If your file has a size smaller than 300 MB, the recommended approach is ***Optimize loading for large files***. For documents that exceed 300 MB the recommended approach is ***Optimize loading for huge files***.

File Sizes Smaller than 300 MB

For editing large documents (file size up to 300 Megabytes), a special memory optimization is implemented on loading such a file so that the total memory allocated for the application is not exceeded.

A temporary buffer file is created on disk so you have to make sure that the available free disk space is at least double the size of the large file that you want to edit. For example, Oxygen XML Editor can load a 200-Megabytes file using a minimum memory setting of 512 Megabytes and at least 400-Megabytes free disk space.

The increase of the maximum size of editable files includes the following restrictions:

- A file larger than the value of the above option is edited only in **Text** mode.
- The *automatic validation* is not available when editing a very large file.
- The XPath filter is disabled in *the Find/Replace dialog box*.
- The bidirectional Unicode support (right-to-left writing) is disabled.
- *The **Format and indent the document on open** option* is deselected for non-XML documents. For XML documents, it is done optimizing the memory usage but without respecting the options set in *the **Format** preferences page*.
- Less precise localizations for the results of an *XPath expression*.

File Sizes Greater than 300 MB

Files tend to become larger and larger mostly because they are frequently used as a format for database export or for porting between different database formats. Traditional text editors simply cannot handle opening these huge export files, some having sizes exceeding one gigabyte, because all the file content must be loaded in memory before the user can actually view it.

The file is split in multiple pages (each having about 1MB in size). Each page is individually loaded (and edited) in the **Text** mode by using the special horizontal slider located at the top of the editing area. The loading operation is very fast and has no upper limit for the size of the loaded file.

The increase of the maximum size of editable files includes the following restrictions:

- For XML files, only the UTF-8, UTF-16, and ASCII encodings are handled; for all non-XML files, the content is considered to be UTF-8.
- Files can be edited in **Text** editing mode only.
- The *automatic validation* is disabled.
- The XPath filter is disabled in the ***Find/Replace** dialog box*.
- The bidirectional Unicode support (right-to-left writing) is disabled.
- *The **Format and indent the document on open** option* is deselected for non-XML documents. For XML documents, the format and indent operation it is done optimizing the memory usage, but it ignores the options set in *the **Format** preferences page*.
- The ***Outline** view* is not supported.
- The file content is soft wrapped by default.
- The ***Find/Replace** dialog box* only supports the **Find** action.
- Saving changes is possible if **Safe save** is activated.
- The **undo** operation is not available if you go to other pages and come back to the modified page. the Undo operation loses its previous states if the back and forth between

Scratch Buffer

The **Scratch Buffer** view can be used for storing fragments of arbitrary text during the editing process. It can be used to drop bits of paragraphs (including arbitrary XML markup fragments) while rearranging and editing the document and also to drag and drop fragments of text from the Scratch Buffer to the editor panel. The **Scratch Buffer** is basically a text area offering XML syntax highlight. The view's contextual menu contains basic edit actions such as **Cut**, **Copy**, and **Paste**.

If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Handling Read-Only Files

If a file marked as read-only is opened in Oxygen XML Editor you can by default perform modifications to it. This behavior is controlled by the ***Can edit read only files** option*. When attempting to save such files you will be prompted to save them to another location.

You can check out the read-only state of the file by looking in the ***Properties*** *view*. If you modify the file properties from the operating system and the file becomes writable, you can modify it on the spot without having to reopen it.

The read-only state is marked with a lock decoration that appears in the editor tab and specified in the tooltip for a certain tab.

Editing Documents with Long Lines

When working with documents that contain lines of text that exceed the boundaries of your monitor, you might want to see the text wrapped. To do so, use one of the following methods:

- Press **Ctrl + Shift + Y (Command + Shift + Y on OS X)** to toggle the line wrap feature for the current document only.
- Select the ***Line wrap*** option in the **Text** preferences page to apply the line wrap to all documents.

Features that Might be Affected by Wrapping Lines of Text

Documents that contain thousands of characters per line can affect the performance of Oxygen XML Editor **Text** mode. When a certain line length limit is reached (controlled from the *Optimize loading for documents with lines longer than (Characters)* on page option), Oxygen XML Editor prompts you to wrap the lines of text. By doing so, the following features may be affected to maintain a reasonable level of productivity:

- The editor uses the `Monospaced` font.
- You cannot set font styles.
- *Automatic validation* is disabled.
- *Automatic spell checking* is disabled.
- When editing XML documents, the **XPath** field is disabled in the ***Find/Replace*** *dialog box*.
- Less precise localization for executed XPath expressions in XML documents. The XPath executions use SAX sources for a smaller memory footprint. We recommend using XPath 2.0 instead of XPath 1.0 because it features an increased execution speed and uses a smaller memory footprint. Running an XPath expression requires additional memory of about 2 or 3 times the size of the document on disk.

XML Digital Signatures

This chapter explains how to apply and verify digital signatures on XML documents.

Digital Signatures Overview

Digital signatures are widely used as security tokens, not just in XML. A *digital signature* provides a mechanism for assuring integrity of data, the authentication of its signer, and the non-repudiation of the entire signature to an external party:

- A *digital signature* must provide a way to verify that the data has not been modified or replaced to ensure integrity.
- The *signature* must provide a way to establish the identity of the data's signer for authentication.
- The *signature* must provide the ability for the data's integrity and authentication to be provable to a third party for non-repudiation.

A *public key system* is used to create the digital signature and it's also used for verification. The signature binds the signer to the document because digitally signing a document requires the originator to create a hash of the message and then encrypt that hash value with their own private key. Only the originator has that private key and that person is the only one who can encrypt the hash so that it can be unencrypted using their public key. The recipient, upon receiving both the message and the encrypted hash value, can decrypt the hash value, knowing the originator's public key. The recipient must also try to generate the hash value of the message and compare the newly generated hash value with the unencrypted hash value received from the originator. If the hash values are identical, it proves that the originator created the message, because only the actual originator could encrypt the hash value correctly.

XML Signatures can be applied to any digital content (data object), including XML (see W3C Recommendation, *XML-Signature Syntax and Processing*). An XML Signature may be applied to the content of one or more resources:

- Enveloped or enveloping signatures are applied over data within the same XML document as the signature
- Detached signatures are applied over data external to the signature element; the signature is "detached" from the content it signs. This definition typically applies to separate data objects, but it also includes the instance where the signature and data object reside within the same XML document but are sibling elements.

The *XML Signature* is a method of associating a key with referenced data. It does not normatively specify how keys are associated with persons or institutions, nor the meaning of the data being referenced and signed.

The original data is not actually signed. Instead, the signature is applied to the output of a chain of *canonicalization* and transformation algorithms, which are applied to the data in a designated sequence. This system provides the flexibility to accommodate whatever "normalization" or desired preprocessing of the data that might be required or desired before subjecting it to being signed.

Since the signature is dependent on the content it is signing, a signature produced from a *non-canonicalized* document could possibly be different from one produced from a *canonicalized* document. The *canonical* form of an XML document is physical representation of the document produced by the method described in this specification. The *XML canonicalization* method is the algorithm defined by this specification that generates the canonical form of a given XML document or document subset. *XML canonicalization* is designed to be useful for applications that require the ability to test whether or not the information content of a document or document subset has been changed. This is done by comparing the *canonical* form of the original document before application processing with the *canonical* form of the document result of the application processing.

A digital signature over the *canonical* form of an XML document or document subset would allows the signature digest calculations to be oblivious to changes in the original document's physical representation. During signature generation, the digest is computed over the *canonical* form of the document. The document is then transferred to the relying party, which validates the signature by reading the document and computing a digest of the *canonical* form of the received document. The equivalence of the digests computed by the signing and relying parties (hence, the equivalence of the *canonical* forms for which they were computed) ensures that the information content of the document has not been altered since it was signed.

The following *canonicalization algorithms* are used in Oxygen XML Editor:

- **Canonical XML (or Inclusive XML Canonicalization)** (*XMLC14N*) - Used for XML where the context doesn't change.

  Inclusive Canonicalization copies all the declarations, even if they are defined outside of the scope of the signature, and all the declarations you might use will be unambiguously specified. *Inclusive Canonicalization* is useful when it is less likely that the signed data will be inserted in other XML document and it is the safer method from the security standpoint because it requires no knowledge of the data that are to be secured to safely sign them. A problem may occur if the signed document is moved into another XML document that has other declarations because the *Inclusive Canonicalization* will copy them and the signature will be invalid.
- **Exclusive XML Canonicalization** (*EXCC14N*) - Designed for *canonicalization* where the context might change.

  Exclusive Canonicalization just copies the namespaces you are actually using (the ones that are a part of the XML syntax). It does not look into attribute values or element content, so the namespace declarations required to process these are not copied. This is useful if you have a signed XML document that you want to insert into other XML documents (or you need self-signed structures that support placement within various XML contexts), as it will ensure the signature is verified correctly each time.

The *canonicalization* method can specify whether or not comments should be included in the *canonical* form output by the *XML canonicalization* method. If a *canonical* form contains comments corresponding to the comment nodes in the input node-set, the result is called *canonical* XML with comments. In an uncommented *canonical* form comments are removed, including delimiter for comments outside document element.

The three operations. ***Canonicalize***, ***Sign***, and ***Verify Signature***, are available from the **Source** submenu when invoking the contextual menu in **Text** mode or from the **Tools** menu.

Related Information:

Certificates on page 738

Canonicalizing Files on page 738

Signing Files on page 739
Verifying Signature on page 740
Example of How to Digitally Sign XML Files or Content on page 741

Certificates

A certificate is a digitally signed statement from the issuer (an individual, an organization, a website or a firm), saying that the public key (and some other information) of some other entity has a particular value. When data is digitally signed, the signature can be verified to check the data integrity and authenticity. Integrity means that the data has not been modified. Authenticity means the data comes indeed from the entity that claims to have created and signed it. Certificates are kept in special repositories called *keystores*.

All *keystore* entries (key and trusted certificate entries) are accessed via unique aliases. An alias must be assigned for every new entry of either a key or certificate as a reference for that entity. No *keystore* can store an entity if its alias already exists in that *keystore* and cannot store trusted certificates generated with keys in its *keystore*.

Oxygen XML Editor provides two types of *keystores*: Java Key Store (JKS) and Public-Key Cryptography Standards version 12 (PKCS-12). A *keystore* file is protected by a password. In a PKCS 12 *keystore* you should not store a certificate without alias together with other certificates, with or without alias, as in such a case the certificate without alias cannot be extracted from the *keystore*.

To configure the options for a certificate or to validate it, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **XML** > **XML Signing Certificates**. This opens *the certificates preferences page*.

Related Information:
Digital Signatures Overview on page 736

Canonicalizing Files

You can select the *canonicalization* algorithm to be used for a document from the dialog box that is displayed by using the **Canonicalize** action that is available from the **Source** submenu when invoking the contextual menu in **Text** mode or from the **Tools** menu.

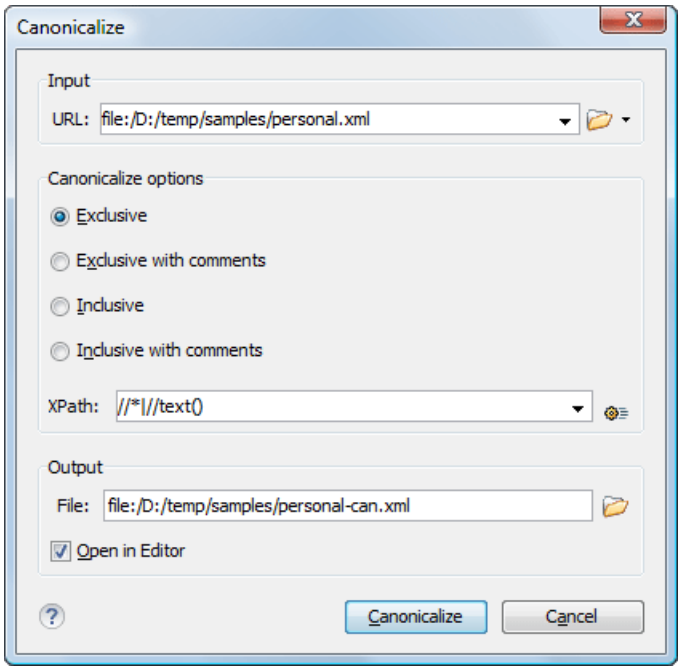


Figure 443: Canonicalization Settings Dialog Box

The **Canonicalize** dialog box allows you to set the following options:

- **Input URL** - Available if the **Canonicalize** action was selected from the **Tools** menu. It allows you to specify the location of the input file.

- **Exclusive** - If selected, the exclusive (uncommented) *canonicalization* method is used.

  Note: *Exclusive Canonicalization* just copies the namespaces you are actually using (the ones that are a part of the XML syntax). It does not look into attribute values or element content, so the namespace declarations required to process these are not copied. This is useful if you have a signed XML document that you want to insert into other XML documents (or you need self-signed structures that support placement within various XML contexts), as it will ensure the signature is verified correctly each time.
- **Exclusive with comments** - If selected, the exclusive with comments *canonicalization* method is used.
- **Inclusive** - If selected, the inclusive (uncommented) *canonicalization* method is used.

  Note: *Inclusive Canonicalization* copies all the declarations, even if they are defined outside of the scope of the signature, and all the declarations you might use will be unambiguously specified. Inclusive *Canonicalization* is useful when it is less likely that the signed data will be inserted in other XML document and it is the safer method from the security standpoint because it requires no knowledge of the data that are to be secured to safely sign them. A problem may occur if the signed document is moved into another XML document that has other declarations because the Inclusive *Canonicalization* will copy them and the signature will be invalid.
- **Inclusive with comments** - If selected, the inclusive with comments *canonicalization* method is used.
- **XPath** - The XPath expression provides the fragments of the XML document to be signed.
- **Output** - Available if the **Canonicalize** action was selected from the **Tools** menu. It allows you to specify the output file path where the signed XML document will be saved.
- **Open in editor** - If selected, the output file will be opened in the editor.

Related Information:

Digital Signatures Overview on page 736

Signing Files

You can select the type of signature to be used for documents from a signature settings dialog box. To open this dialog box, select the **Sign** action from the **Source** submenu when invoking the contextual menu in **Text** mode or from the **Tools** menu.

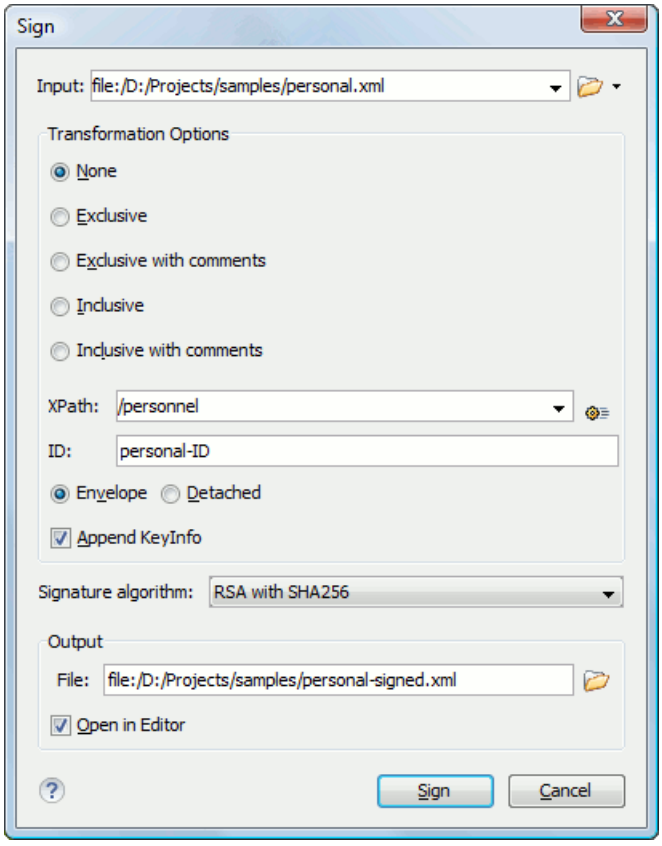


Figure 444: Signature Settings Dialog Box

The following options are available:

Note: If Oxygen XML Editor could not find a valid certificate, a link is provided at the top of the dialog box that opens the *XML Signing Certificates preferences page* where you can configure a valid certificate.

- **Input** - Available if the **Sign** action was selected from the **Tools** menu. Specifies the location of the input URL.
- **Transformation Options** - See the *Digital Signature Overview* section for more information about these options.
  - **None** - If selected, no *canonicalization* algorithm is used.
  - **Exclusive** - If selected, the exclusive (uncommented) *canonicalization* method is used.

    Note: *Exclusive Canonicalization* just copies the namespaces you are actually using (the ones that are a part of the XML syntax). It does not look into attribute values or element content, so the namespace declarations required to process these are not copied. This is useful if you have a signed XML document that you want to insert into other XML documents (or you need self-signed structures that support placement within various XML contexts), as it will ensure the signature is verified correctly each time.
  - **Exclusive with comments** - If selected, the exclusive with comments *canonicalization* method is used.
  - **Inclusive** - If selected, the inclusive (uncommented) *canonicalization* method is used.

    Note: *Inclusive Canonicalization* copies all the declarations, even if they are defined outside of the scope of the signature, and all the declarations you might use will be unambiguously specified. Inclusive *Canonicalization* is useful when it is less likely that the signed data will be inserted in other XML document and it is the safer method from the security standpoint because it requires no knowledge of the data that are to be secured to safely sign them. A problem may occur if the signed document is moved into another XML document that has other declarations because the Inclusive *Canonicalization* will copy them and the signature will be invalid.
  - **Inclusive with comments** - If selected, the inclusive with comments *canonicalization* method is used.
- **XPath** - The XPath expression provides the fragments of the XML document to be signed.
- **ID** - Provides ID of the XML element to be signed.
- **Envelope** - If selected, the *enveloped* signature is used. See the *Digital Signature Overview* for more information.
- **Detached** - If selected, the *detached* signature is used. See the *Digital Signature Overview* for more information.
- **Append KeyInfo** - If this option is selected, the `ds:KeyInfo` element will be added in the signed document.
- **Signature algorithm** - The algorithm used for signing the document. The following options are available: **RSA with SHA1**, **RSA with SHA256**, **RSA with SHA384**, and **RSA with SHA512**.
- **Output** - Available if the **Sign** action was selected from the **Tools** menu. Specifies the path of the output file where the signed XML document will be saved.
- **Open in editor** - If selected, the output file will be opened in Oxygen XML Editor.

Related Information:

Digital Signatures Overview on page 736
Verifying Signature on page 740
Example of How to Digitally Sign XML Files or Content on page 741

Verifying Signature

You can verify the signature of a file by selecting the **Verify Signature** action from the **Source** submenu when invoking the contextual menu in **Text** mode or from the **Tools** menu. The **Verify Signature** dialog box then allows you to specify the location of the file whose signature is verified.

If the signature is valid, a dialog box displays the name of the signer. Otherwise, an error shows details about the problem.

Related Information:

Digital Signatures Overview on page 736
Signing Files on page 739
Example of How to Digitally Sign XML Files or Content on page 741

Example of How to Digitally Sign XML Files or Content

Suppose you want to digitally sign an XML document, but more specifically, suppose you have multiple instances of the same element in the document and you just want to sign a specific ID. Oxygen XML Editor includes a signature tool that allows you to digitally sign XML documents or specific content.

The Oxygen XML Editor installation directory includes a `samples` folder that contains a file called `personal.xml`. For the purposes of this example, this file will be used to demonstrate how to digitally sign specific content. Notice that this file has multiple `person` elements inside the `personnel` element. Suppose you want to digitally sign the specific `person` element that contains the `id=robert.taylor`. To do this, follow this procedure:

1. Open the `personal.xml` file in Oxygen XML Editor in **Text** editing mode.
2. Right-click anywhere in the editor and select the **Sign** action from the **Source** submenu.
   The **Sign** dialog box is displayed.

   Tip: If you want to sign a file but create a new output file so that the original file remains unchanged, use the **Sign** action from the **Tools** menu. Selecting the action from this menu will allow you to choose an input file and output file in the **Sign** dialog box.
3. If Oxygen XML Editor cannot find a valid certificate, click the link at the top of the dialog box to **configure a valid certificate**. This opens the ***XML Signing Certificates*** *preferences page* that allows you to configure and validate a certificate.
4. Once a valid certificate is recognized, continue to configure the **Sign** dialog box.
   a) Select one of *the* ***Transformation Options***. For the purposes of this example, select the **Inclusive with comments** option.
   b) Specify the appropriate **XPath** expression for the specific element that needs to be signed. For this example, type `/personnel/person` in the **XPath** text box.
   c) Enter the specific **ID** that needs to be signed. For this example, type `robert.taylor` in the **ID** field.
   d) Select *the* ***Envelope*** *option* and leave the other options as their default values.

   The digital signature is added at the end of the XML document, just before the end tag. It is always added at the end of the document, even if you only sign specific content within the document.
5. You can verify the signature by choosing the **Verify Signature** action from the **Source** submenu of the contextual menu.

Related Information:

Digital Signatures Overview on page 736
Signing Files on page 739
Verifying Signature on page 740

Compare Files or Directories

Oxygen XML Editor provides a simple means of performing file and folder comparisons. You can see the differences in your files and folders and merge the changes. You can also use the file comparison to compare fragments or files inside zip-based archives.

There are two types of comparison tools: **Compare Directories** or **Compare Files**. These utilities are available from the **Tools** menu or can be opened as stand-alone applications from the Oxygen XML Editor installation folder (`diffDirs.exe` and `diffFiles.exe`).

Starting the Tools from a Command Line

The comparison tools can also be started by using command-line arguments. In the installation folder there are two executable shells (`diffFiles.bat` and `diffDirs.bat` on Windows, `diffFiles.sh` and `diffDirs.sh` on Unix/Linux, `diffFilesMac.sh` and `diffDirsMac.sh` on OS X). To specify files or directories to compare, you can pass command-line arguments to each of these shells. The arguments can point to file or folder paths in directories or archives (supported formats: *zip*, *docx*, and *xlsx*).

Directory Comparison Example

To start a *comparison between the two directories*, use the following construct: `diffDirs.bat/diffDirs.sh/diffDirsMac.sh [directory path 1] [directory path 2]`. If you pass only one argument, you are prompted to manually choose the second directory or archive.

For example, to start a comparison between two Windows directories, the command line would look like this:

```
diffDirs.bat "c:\documents new" "c:\documents old"
```

Tip: If there are spaces in the path names, surround the paths with quotes.

File Comparison Example

To start a *comparison between 2 or 3 files*, use the following construct: `diffFiles.bat/diffFiles.sh/diffFilesMac.sh [path to left file] [path to right file] [path to base file]`.

If three files are specified, the tool will start in the *3-way comparison mode*. If only two files are specified, the tool will start in the *2-way comparison mode*. The first specified file will be added to the left panel in the comparison tool, the second file to the right panel, and the optional third file will be the base (ancestor) file used for a 3-way comparison. If you pass only one argument, you are prompted to manually choose another file.

For example, to do a 3-way comparison on Windows, the command line would look like this:

```
diffFiles.bat "c:\docs\file 1" "c:\docs\file 2" c:\docs\basefile
```

Tip: If there are spaces in the path names, surround the paths with quotes.

Compare Files

The **Compare Files** tool can be used to compare files or XML file fragments. The tool provides a mechanism for comparing two files or fragments, as well as the mechanism for a three-way comparison. The utility is available from the **Tools** menu or can be opened as a stand-alone application from the Oxygen XML Editor installation folder (`diffFiles.exe`).

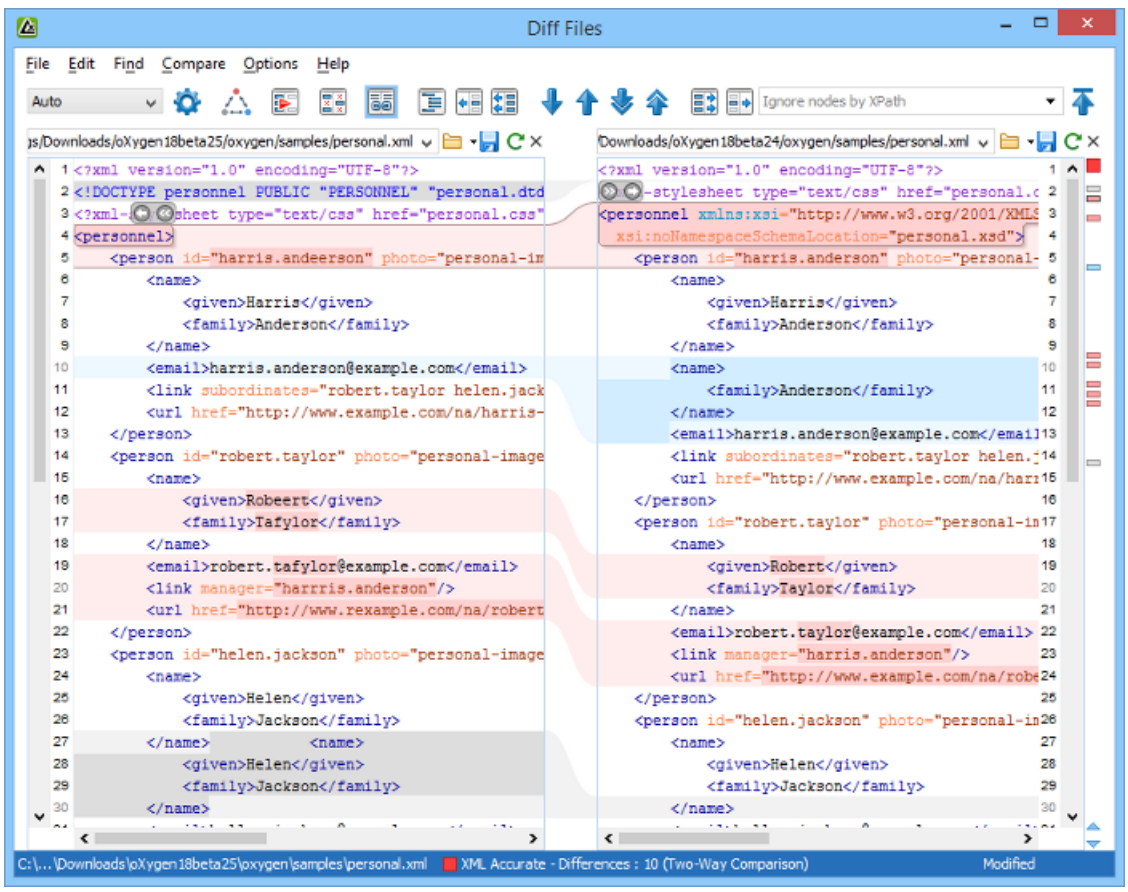


Figure 445: Compare Files Tool

Starting the Tool from a Command Line

The file comparison tool can also be started by using command-line arguments. In the installation folder there is an executable shell (`diffFiles.bat` on Windows, `diffFiles.sh` on Unix/Linux, `diffFilesMac.sh` on OS X). To specify the files to compare, you can pass command-line arguments using the following construct: `diffFiles.bat/diffFiles.sh/diffFilesMac.sh [path to left file] [path to right file] [path to 3-way base file]`.

If three files are specified, the tool will start in the *3-way comparison mode*. If only two files are specified, the tool will start in the *2-way comparison mode*. The first specified file will be added to the left panel in the comparison tool, the second file to the right panel, and the optional third file will be the base (ancestor) file used for a 3-way comparison. If you pass only one argument, you are prompted to manually choose another file.

If you want to launch the file comparison tool from an external application with specified files and you want the file browsing tools at the top of both panels to be hidden, you should use the `-ext` argument as the first command. There are some additional arguments that are allowed and to see all the details for the command line construct, type `diffFiles.bat --help` in the command line.

For example, to do a 3-way comparison on Windows, the command line might look like this:

```
diffFiles.bat "c:\docs\file 1" "c:\docs\file 2" c:\docs\basefile
```

Tip: If there are spaces in the path names, surround the paths with quotes.

Two-Way Comparisons

The **Compare Files** tool can be used to compare the differences between two files or XML fragments.

Compare Files

To perform a two-way comparison, follow these steps:

1. Open a file in the left panel and the file you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the browsing tools in the **Browse** drop-down menu.

   Step Result: The selected files are opened in the two side-by-side editors. A text editing mode is used to offer a better view of the differences.
2. To highlight the differences between the two files, click the **Perform File Differencing** button from the toolbar.
3. You can use the drop-down menu on the left side of the toolbar to change the *algorithm* for the operation.
4. You can also use the **Diff Options** button to access the **Files Comparison** preferences page where you can choose to ignore certain types of markup and configure various options.
5. If you are comparing XML documents using the **XML Fast** or **XML Accurate** algorithms, you can enter an XPath 2.0 expression in the **Ignore nodes by XPath** text field to ignore certain nodes from the comparison.

The resulting comparison will show you differences between the two files. The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes. This layout allows you to easily identify and focus on a group of related changes.

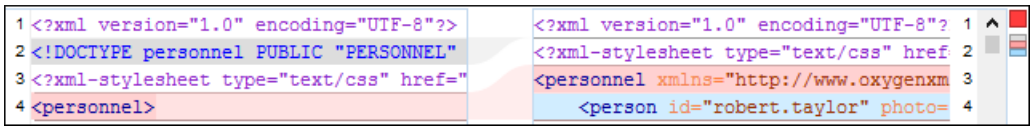


Figure 446: Two-Way Differences

Highlighting Colors

The differences are also highlighted in several colors, depending on the type of change, and dynamic lines connect the compared fragments in the middle section between the two panes. The highlighting colors can be customized in the ***Files Comparison / Appearance*** *preferences page*, but the default colors and their shades mean the following:

- **Pink** - Identifies modifications on either side.
- **Gray** - Identifies an addition of a node in the left side (your outgoing changes).
- **Blue** - Identifies an addition of a node in the right side (incoming changes).
- **Lighter Shade** - Identifies blocks of changes that can be merged in their entirety.
- **Darker Shade** - Identifies specific changes within the blocks that can be merged more precisely.

Compare Fragments

To compare XML file fragments, you need to copy and paste the fragments you want to compare into each side, without selecting a file. If a file is already selected, you need to close it using the **Close (Ctrl + W (Command + W on OS X))** button, before pasting the fragments. If you save modified fragments, a dialog box opens that allows you to save the changes as a new document.

Navigate Differences

To navigate through differences, do one of the following:

- Use the navigation buttons on the toolbar (or in the **Compare** menu).
- Select a block of differences by clicking its small colored marker in the overview ruler located in the right-most part of the window. At the top of the overview ruler there is a success indicator that turns green where there are no differences, or red if differences are found.
- Click a colored area in between the two text editors.

Editing Actions

You can edit the files directly in either editing pane. The two editors are constantly synchronized and the differences are refreshed when you save the modified document or when you click the **Perform File Differencing** button.

A variety of actions are available on the *toolbar* and in the *various menus* (these same actions are also available in the contextual menu in both editing panes). The tool also includes some inline actions to help you merge, copy, or remove changes. When you select a change, the following inline action widgets are available, depending on the type of change:

Append left change to right and Append right change to left

Copies the content of the selected change from one side and appends it on the other, according to the content of the corresponding change. As a result, the side where the arrow points to will contain the changes from both sides.

Copy change from left to right and Copy change from right to left

Replaces the content of a change from one side with the content of the corresponding change from the other side.

Remove change

Rejects the change on the particular side and preserves the particular content on the other side.

Two-Way Diff Algorithms

Oxygen XML Editor offers the following two-way diff algorithms to compare files or fragments:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Characters** - Computes the differences at character level, meaning that it compares two files or fragments looking for identical characters.
- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words.
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **Syntax Aware** - Computes differences for known file types or fragments. This algorithm splits the files or fragments into sequences of *tokens* and computes the differences between them. The meaning of a *token* depends on the type of compared files or fragments.

Known file types include those listed in the **New** dialog box, such as XML file types (XSLT files, XSL-FO files, XSD files, RNG files, NVDL files, etc.), XQuery file types (`.xquery`, `.xq`, `.xqy`, `.xqm` extensions), DTD file types (`.dtd`, `.ent`, `.mod` extensions), TEXT file type (`.txt` extension), or PHP file type (`.php` extension).

For example:

- When comparing XML files or fragments, a token can be one of the following:
  - The name of an XML tag
  - The < character
  - The /> sequence of characters
  - The name of an attribute inside an XML tag
  - The = sign
  - The " character
  - An attribute value
  - The text string between the start tag and the end tag (a text node that is a child of the XML element corresponding to the XML tag that encloses the text string)
- When comparing plain text, a token can be any continuous sequence of characters or any continuous sequence of whitespaces, including a new line character.

- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Three-Way Comparisons

Oxygen XML Editor also includes a three-way comparison feature to help you solve conflicts and merge changes between multiple modifications. It is especially helpful for teams who have multiple authors editing and committing the same documents. It provides a comparison between a local change, another change, and the original base revision. Some additional advantages include:

- Visualize and merge content that was modified by you and another member of your team.
- Marks differences correctly even when the document structure is rearranged.
- Allows you to merge XML-relevant modifications.

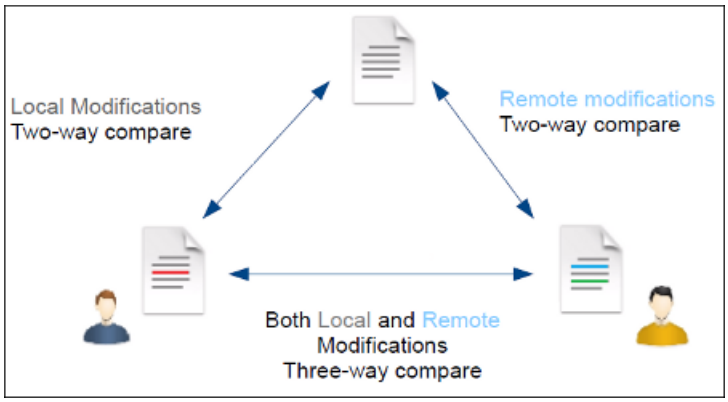


Figure 447: Three-Way Comparison

Compare Files

To perform a three-way comparison, follow these steps:

1. Open a file in the left panel and the file you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the browsing tools in the **Browse** drop-down menu.

   Step Result: The selected files are opened in the two side-by-side editors. A text editing mode is used to offer a better view of the differences.

2. Click the **Three-Way Comparison** button on the toolbar and select the base file in the **Base** field. You can specify the path by using the text field, the history drop-down, or the browsing tools in the **Browse** drop-down menu.
3. To highlight the differences, click the **Perform File Differencing** button on the toolbar.
4. You can use the drop-down menu on the left side of the toolbar to change the *algorithm* for the operation.
5. You can also use the **Diff Options** button to access the **Files Comparison** preferences page where you can choose to ignore certain types of markup and configure various options.

The resulting comparison will show you differences between the two files, as well as differences between either of them and the base (ancestor) file. The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes.

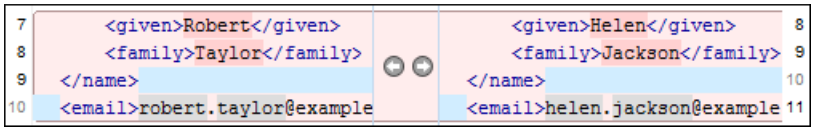


Figure 448: Three-Way Differences

Highlighting Colors

The differences are also highlighted in several colors, depending on the type of change, and dynamic lines connect the compared fragments in the middle section between the two panes. The highlighting colors can be customized in the ***Files Comparison / Appearance*** *preferences page*, but the default colors and their shades mean the following:

- **Pink** - Identifies blocks of changes that include conflicts.
- **Gray** - Identifies your outgoing changes that do not include conflicts.
- **Blue** - Identifies incoming changes that do not include conflicts.
- **Lighter Shade** - Identifies blocks of changes that can be merged in their entirety.
- **Darker Shade** - Identifies specific changes within the blocks that can be merged more precisely.

Navigate Differences

To navigate through differences, do one of the following:

- Use the navigation buttons on the toolbar (or in the **Compare** menu).
- Select a block of differences by clicking its small colored marker in the overview ruler located in the right-most part of the window. At the top of the overview ruler there is a success indicator that turns green where there are no differences, or red if differences are found.
- Click a colored area in between the two text editors.

Editing Actions

You can edit the files directly in either editing pane. The two editors are constantly synchronized and the differences are refreshed when you save the modified document or when you click the **Perform File Differencing** button.

A variety of actions are available on the *toolbar* and in the *various menus* (these same actions are also available in the contextual menu in both editing panes). The tool also includes some inline actions to help you merge, copy, or remove changes. When you select a change, the following inline action widgets are available, depending on the type of change:

Append left change to right and Append right change to left

Copies the content of the selected change from one side and appends it on the other, according to the content of the corresponding change. As a result, the side where the arrow points to will contain the changes from both sides.

Copy change from left to right and **Copy change from right to left**

Replaces the content of a change from one side with the content of the corresponding change from the other side.

Remove change

Rejects the change on the particular side and preserves the particular content on the other side.

Three-Way Diff Algorithms

Oxygen XML Editor offers the following three-way diff algorithms to compare files:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Second Level Comparisons

For both two-way and three-way comparisons, Oxygen XML Editor automatically performs a second level comparison for the **Lines**, **XML Fast**, and **XML Accurate** algorithms. After the first comparison is finished, the second level comparisons for the **Lines** algorithm is processed on text nodes using a word level comparison, meaning that it looks for identical words. For the **XML Fast** and **XML Accurate** algorithms, the second level comparison is processed using a *syntax-aware comparison*, meaning that it looks for identical *tokens*. This second level comparison makes it easier to spot precise differences and you can merge or reject the precise modifications.

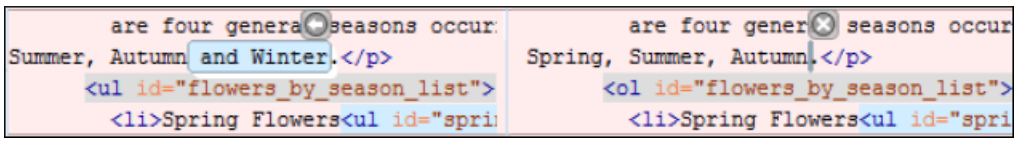


Figure 449: Second Level Diff Comparison

Note: If a modified text fragment contains XML markup (such as processing instructions, XML comments, CData, or elements), the second level comparison will not automatically be performed. In this case you can manually select a second level comparison by doing a word level or character level comparison.

To do a word level comparison, select **Show word level details** from the contextual menu or **Compare** menu.

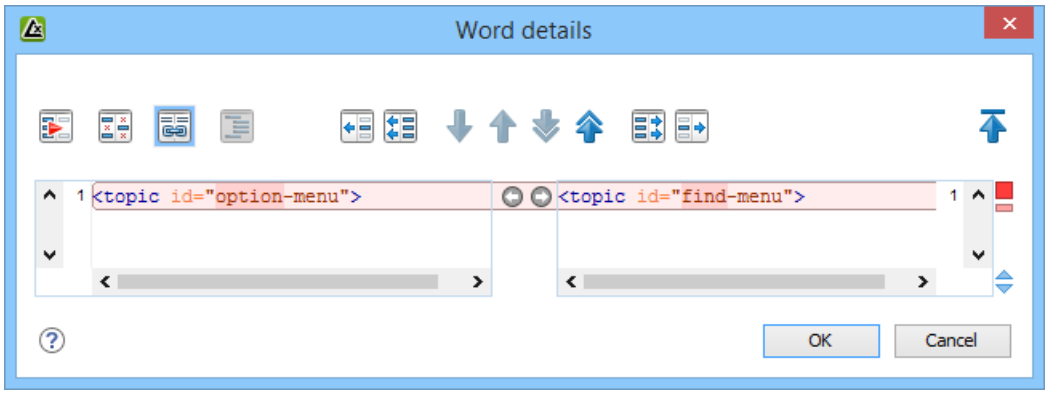


Figure 450: Word Level Comparison

To do a character level comparison, select **Show Character Level details** from the contextual menu or **Compare** menu.

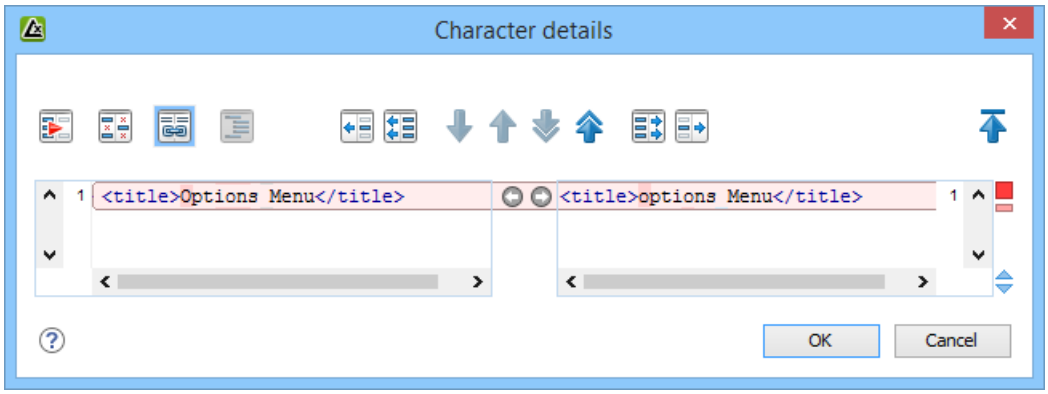


Figure 451: Character Level Comparison

Related Information:

Files Comparison Preferences Page on page 145
Compare Directories on page 753

Toolbar and Contextual Menu Actions of the Compare Files Tool

The toolbar of the **Compare Files** tool contains operations that can be performed on the source and target files or XML fragments. Many of the actions are also available in the contextual menu.

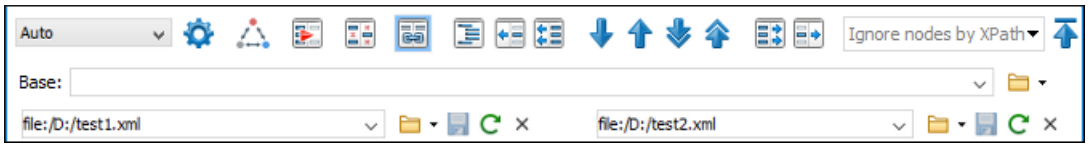


Figure 452: Compare Toolbar

The following actions are available:

Algorithm

This drop-down menu allows you to select one of the following diff algorithms (depending on whether it is a two-way or three-way comparison):

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Characters** - Computes the differences at character level, meaning that it compares two files or fragments looking for identical characters.
- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words.
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **Syntax Aware** - Computes differences for the file types or fragments known by Oxygen XML Editor, taking the syntax (the specific types of tokens) into consideration.
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Diff Options

Opens the ***Files Comparison*** *preferences page* where you can configure various options.

Three-Way Comparison

Toggle action that allows you to perform a three-way comparison between the two files displayed in the two editing panes and a base (ancestor) file.

Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.

Ignore Whitespaces

Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces.

Synchronized scrolling

Toggles synchronized scrolling on or off so that a selected difference can be seen on both sides of the application window. This option is on by default.

Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on OS X))

Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.

Note: When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.

Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.

Copy All Changes from Right to Left

Copies all changes from the file in the right panel to the file in the left panel.

Next Block of Changes (Ctrl + Period (Command + Period on OS X))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

Note: A change block groups one or more consecutive lines that contain at least one change.

Previous Block of Changes (Ctrl + Comma (Command + Comma on OS X))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

Next Change (Ctrl + Shift + Period (Command + Shift + Period on OS X))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

Previous Change (Ctrl + Shift + Comma (Command + Shift + M on OS X))

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

Copy All Changes from Left to Right

Copies all changes from the file in the left panel to the file in the right panel.

Copy Change from Left to Right

Copies the selected difference from the file in the left panel to the file in the right panel.

Ignore Nodes by XPath

You can use this text field to enter an *XPath expression* to ignore certain nodes from the comparison. It will be processed as XPath version 2.0. You can also enter the name of the node to ignore all nodes with the specified name (for example, if you want to ignore all ID attributes from the document, you could simply enter `@id`). This field is only available when comparing XML documents using the **XML Fast** or **XML Accurate** algorithms.

Note: If an XPath expression is specified in the ***Ignore nodes by XPath*** *option* in the **Diff / File Comparison** preferences page, that one is used as a default when the application is started. If you then enter an

expression in this field on the toolbar, this one will be used instead of the default. If you delete the expression from this field, neither will be used.

First Change (Ctrl + B (Command + B on OS X))

Jumps to the first change.

Base

Available for *three-way comparisons*. It is the base file that will be compared with the files opened in the left and right editors. You can specify the path to the file by using the text field, its history drop-down, or the browsing tools in the **Browse** drop-down menu.

Left-Side (Source) File

You can specify the path to the file to be compared on the left side (source) by using the text field, its history drop-down, or the browsing tools in the **Browse** drop-down menu.

Save

Saves the changes made in the source (left-side) file.

Reload

Reloads the source (left-side) file.

Close

Closes the source (left-side) file.

Right-Side (Target) File

You can specify the path to the file to be compared on the right side (target) by using the text field, its history drop-down, or the browsing tools in the **Browse** drop-down menu.

Save

Saves the target (right-side) file.

Reload

Reloads the target (right-side) file.

Close

Closes the target (right-side) file.

Compare Files Tool Menus

The menus in the **Compare Files** tool contain some of the same actions that are on the toolbar, as well as some common actions that are identical to the same actions in the Oxygen XML Editor menus. The menu actions include:

File Menu

Source > Open

Browses for a file that will be displayed in the left panel.

Source > Open URL

Browses for a remote file that will be displayed in the left panel.

Source > Open File from Archive

Browses an archive for a file that will be displayed in the left panel.

Source > Reload

Reloads the file in the left panel.

Source > Save

Saves the changes made to the file in the left panel.

Source > Save As

Allows you to choose a destination to save the file in the left panel.

Source > Close

Closes the file in the left panel.

Target > Open

Browses for a file that will be displayed in the right panel.

Target > Open URL

Browses for a remote file that will be displayed in the right panel.

Target > Open File from Archive

Browses an archive for a file that will be displayed in the right panel.

Target > Reload

Reloads the file in the right panel.

Target > Save

Saves the changes made to the file in the right panel.

Target > Save As

Allows you to choose a destination to save the file in the right panel.

Target > Close

Closes the file in the right panel.

Base > Open

Browses for a file that will be compared with both files in a *three-way comparison*.

Base > Open URL

Browses for a remote file that will be compared with both files in a *three-way comparison*.

Base > Open File from Archive

Browses an archive for a file that will be compared with both files in a *three-way comparison*.

Close (Ctrl + W (Command + W on OS X))

Closes the application.

Edit Menu

Cut

Cut the selection from the currently focused editor panel to the clipboard.

Copy

Copy the selection from the currently focused editor panel to the clipboard.

Paste

Paste content from the clipboard into the currently focused editor panel.

Select all

Selects all content in the currently focused editor panel.

Undo

Undo changes in the currently focused editor panel.

Redo

Redo changes in the currently focused editor panel.

Find Menu

Find/Replace

Perform *find/replace* operations in the currently focused editor panel.

Find Next

Go to the next match using the same options as the last *find* operation. This action runs in both editor panels.

Find Previous

Go to the previous match using the same options as the last *find* operation. This action runs in both editor panels.

Compare Menu

Three-Way Comparison

Toggle action that allows you to perform a three-way comparison between the two files displayed in the two editing panes and a base (ancestor) file.

Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.

Next Block of Changes (Ctrl + Period (Command + Period on OS X))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

Note: A change block groups one or more consecutive lines that contain at least one change.

Previous Block of Changes (Ctrl + Comma (Command + Comma on OS X))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

Next Change (Ctrl + Shift + Period (Command + Shift + Period on OS X))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

Previous Change (Ctrl + Shift + Comma (Command + Shift + M on OS X))

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

Last Change (Ctrl + E (Command + E on OS X))

Jumps to the last change.

First Change (Ctrl + B (Command + B on OS X))

Jumps to the first change.

Copy All Changes from Right to Left

Copies all changes from the file in the right panel to the file in the left panel.

Copy All Changes from Left to Right

Copies all changes from the file in the left panel to the file in the right panel.

Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.

Copy Change from Left to Right

Copies the selected difference from the file in the left panel to the file in the right panel.

Show Word Level Details

Provides a word-level comparison of the selected change.

Show Character Level Details

Provides a character-level comparison of the selected change.

Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on OS X))

Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.

Note: When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.

Options Menu

Preferences

Opens the preferences dialog box that includes numerous pages of options that can be configured.

Menu Shortcut Keys

Opens the **Menu Shortcut Keys** option page where you can configure keyboard shortcuts available for menu items.

Reset Global Options

Resets options to their default values. Note that this option appears only when the tool is executed as a stand-alone application.

Import Global Options

Allows you to import an options set that you have previously exported.

Export Global Options

Allows you to export the current options set to a file.

Help Menu

Help (F1)

Opens a **Help** dialog box that displays the User Manual at a section that is appropriate for the context of the current cursor position.

Use Online Help

If this option is selected, when you select Help or press F1 while hovering over any part of the interface, Oxygen XML Editor attempts to open the help documentation in online mode. If this option is not selected or an internet connection fails, the help documentation is opened in offline mode.

Report problem

Opens a dialog box that allows the user to write the description of a problem that was encountered while using the application. You can change the URL where the reported problem is sent by using the `com.oxygenxml.report.problems.url` system property. The report is sent in XML format through the `report` parameter of the POST HTTP method.

Support Center

Opens the Oxygen XML Editor Support Center web page in a browser.

Compare Directories

The **Compare Directories** tool can be used to compare and manage changes to files and folders within the structure of your directories. The utility is available from the **Tools** menu or can be opened as a stand-alone application from the Oxygen XML Editor installation folder (`diffDirs.exe`).

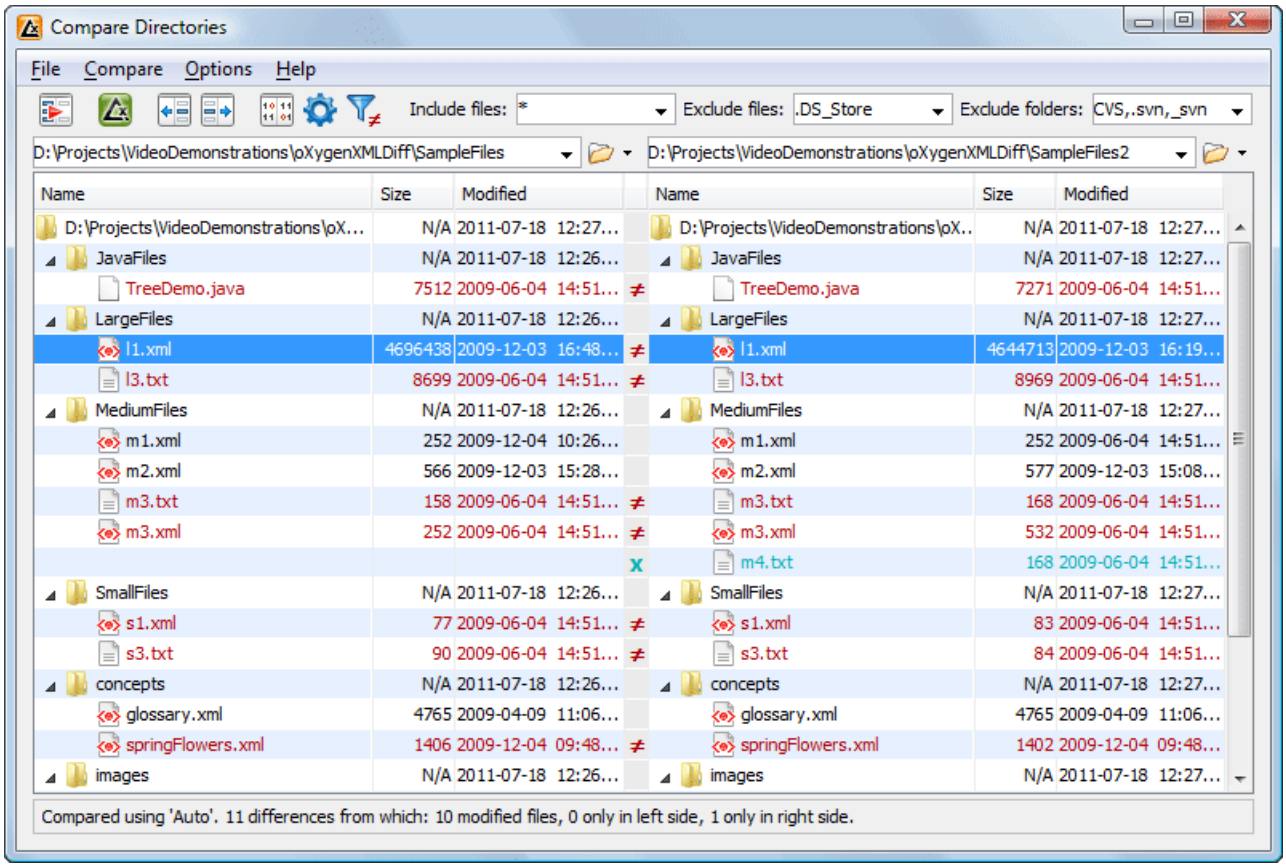


Figure 453: Compare Directories Tool

Starting the Tool from a Command Line

The directory comparison tool can also be started by using command-line arguments. In the installation folder there is an executable shell (`diffDirs.bat` on Windows, `diffDirs.sh` on Unix/Linux, `diffDirsMac.sh` on OS X). To specify the directories to compare, you can pass command-line arguments using the following construct: `diffDirs.bat/diffDirs.sh/diffDirsMac.sh [directory path 1] [directory path 2]`.

If you pass only one argument, you are prompted to manually choose the second directory or archive.

For example, to start a comparison between two Windows directories, the command line would look like this:

```
diffDirs.bat "c:\documents new" "c:\documents old"
```

Tip: If there are spaces in the path names, surround the paths with quotes.

Directory Comparisons

To perform a directory comparison, follow these steps:

1. Select a folder in the left panel and the folder you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the **Browse for local directory** action in the **Browse** drop-down menu.

   Step Result: The selected directory structures are opened in the two side-by-side panels.
2. To highlight the differences between the two folders, click the **Perform Directories Differencing** button from the toolbar.
3. You can also use the **Diff Options** button to access the ***Directories Comparison*** *preferences page* where you can configure various options.

To compare the content of two archives, follow these steps:

1. Use the **Browse for archive file** action in the **Browse** drop-down menu to select the archives in the left and right panels.
2. By default, the supported archives are not treated as directories and the comparison is not performed on the files inside them. To make Oxygen XML Editor treat supported archives as directories, select the ***Look in archives** option* in the **Directories Comparison** preferences page.
3. To highlight the differences, click the **Perform Directories Differencing** button from the toolbar.

The directory comparison results are presented using two tree-like structures showing the files and folders, including their name, size, and modification date. A column that contains graphic symbols separates the two tree-like structures. The graphic symbols can be one of the following:

- An **X** symbol, when a file or a folder exists in only one of the compared directories.
- A ≠symbol, when a file exists in both directories but the content differs. The same sign appears when a collapsed folder contains differing files.

The color used for the symbol and the directory or file name can be customized in the ***Directories Comparison / Appearance** preferences page*. You can double-click lines marked with the ≠ symbol to open a **Compare Files** window, which shows the differences between the two files.

The directories that contain files that differ are expanded automatically so that you can focus directly on the differences. You can merge the contents of the directories by using the copy actions. If you double-click (or press **Enter**) on a line with a pair of files, Oxygen XML Editor starts a *file comparison* between the two files, using the **Compare Files** tool.

Related Information:

Compare Files on page 742

Toolbar and Contextual Menu Actions of the Compare Directories Tool

The toolbar of the **Compare Directories** tool contains operations that can be performed on the compared directory structure. Some of the toolbar actions are also available in the contextual menu.

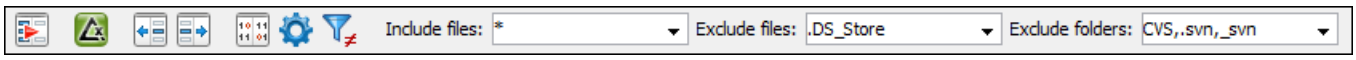


Figure 454: Compare toolbar

Toolbar Actions

Perform Directories Differencing

Looks for differences between the two directories displayed in the left and right side of the application window.

Perform Files Differencing

Opens the ***Compare Files** tool* that allows you to compare the currently selected files.

Copy Change from Right to Left

Copies the selected change from the right side to the left side (if there is no file/folder in the right side, the left file/folder is deleted).

Copy Change from Left to Right

Copies the selected change from the left side to the right side (if there is no file/folder in the left side, the right file/folder is deleted).

Binary Compare

Performs a byte-level comparison on the selected files.

Diff Options

Opens the ***Directory Comparison** preferences page* where you can configure various options.

Show Only Modifications

Displays a more uncluttered file structure by hiding all identical files.

File and folder filters

Differences can be filtered using three combo boxes: **Include files**, **Exclude files**, and **Exclude folders**. They come with predefined values and are editable to allow custom values. All of them accept multiple comma-separated values and the ***** and **?** wildcards. For example, to filter out all JPEG and GIF image files, edit the **Exclude files** filter box to read **`*.jpeg, *.png`**. Each filter includes a drop-down menu with the latest 15 filters applied.

Contextual Menu Actions

Perform Files Differencing

Opens the ***Compare Files*** *tool* that allows you to compare the currently selected files.

Binary Compare

Performs a byte-level comparison on the selected files.

Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.

Copy Change from Left to Right

Copies the selected difference from the file in the left panel to the file in the right panel.

Open

If the action is invoked on a file, the selected file is opened in Oxygen XML Editor. If the action is invoked on a directory, the selected directory is opened in the default file browser for your particular operating system.

Open in System Application

Opens the selected file in the system application that is associated with that type of file. The action is available when launching the **Compare Directories** tool from the **Tools** menu in Oxygen XML Editor.

Show in Explorer

Opens the default file browser for your particular operating system with the selected file highlighted.

Compare Directories Tool Menus

The menus in the **Compare Directories** tool contain some of the same actions that are on the toolbar, as well as some common actions that are identical to the same actions in the Oxygen XML Editor menus. The menu actions include:

File Menu

Close (Ctrl + W (Command + W on OS X))

Closes the application.

Compare Menu

Perform Directories Differencing

Looks for differences between the two directories displayed in the left and right side of the application window.

Perform Files Differencing

Opens the ***Compare Files*** *tool* that allows you to compare the currently selected files.

Copy Change from Right to Left

Copies the selected change from the right side to the left side (if there is no file/folder in the right side, the left file/folder is deleted).

Copy Change from Left to Right

Copies the selected change from the left side to the right side (if there is no file/folder in the left side, the right file/folder is deleted).

Options Menu

Preferences

Opens the preferences dialog box that includes numerous pages of options that can be configured.

Menu Shortcut Keys

Opens the **Menu Shortcut Keys** option page where you can configure keyboard shortcuts available for menu items.

Reset Global Options

Resets options to their default values. Note that this option appears only when the tool is executed as a stand-alone application.

Import Global Options

Allows you to import an options set that you have previously exported.

Export Global Options

Allows you to export the current options set to a file.

Help Menu

Help (F1)

Opens a **Help** dialog box that displays the User Manual at a section that is appropriate for the context of the current cursor position.

Use Online Help

If this option is selected, when you select Help or press F1 while hovering over any part of the interface, Oxygen XML Editor attempts to open the help documentation in online mode. If this option is not selected or an internet connection fails, the help documentation is opened in offline mode.

Report problem

Opens a dialog box that allows the user to write the description of a problem that was encountered while using the application. You can change the URL where the reported problem is sent by using the `com.oxygenxml.report.problems.url` system property. The report is sent in XML format through the `report` parameter of the POST HTTP method.

Support Center

Opens the Oxygen XML Editor Support Center web page in a browser.

Compare Images

You can use the **Compare Directories** tool to compare images. If you double-click a line that contains two different images, the **Compare images** window is displayed. This dialog box presents the images in the left and right sides, scaled to fit the available view area. You can use the contextual menu actions to scale the images to their original size or scale them down to fit in the view area.

The supported image types are: *GIF*, *JPG*, *JPEG*, *PNG*, and *BMP*.

Compare Directories Against a Base (3-Way)

The **Compare Directories Against a Base (3-way)** tool allows you to perform three-way comparisons on directories to help you identify and merge changes between multiple modifications of the same directory structure. It is especially helpful for teams that have multiple authors contributing documents to the same directory system. It offers information about conflicts and changes, and includes actions to easily merge, accept, overwrite, or ignore changes to the directory system.

How to Perform 3-Way Directory Comparisons

To perform a 3-way directories comparison, follow these steps:

1. Select **Compare Directories Against a Base (3-way)** from the **Tools** menu.

   Step Result: This opens a dialog box that allows you to select the 3 file sets that will be used for the comparison.

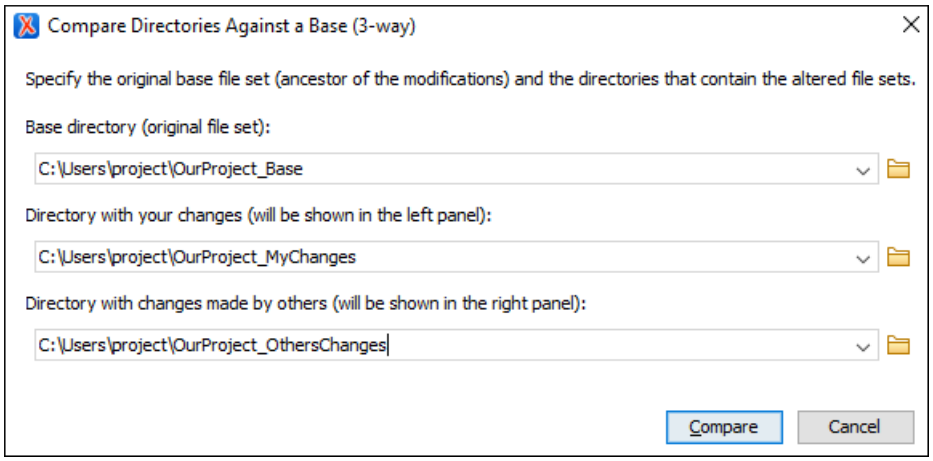


   Figure 455: Compare Directories Against a Base File Set Chooser

2. Select the file sets to be compared:
   - **Base directory** - This is the original (base) file set before any modifications were made by your or others.
   - **Directory with your changes** - This is the file set with changes that you have made. This file set will be displayed in the left panel in the comparison tool.
   - **Directory with changes made by others** - This is the file set with changes made by others that you want to merge with your changes. This file set will be displayed in the right panel in the comparison tool.
3. Click the **Compare** button to compare the file sets and open the comparison and merge tool.
4. Use the features and actions described in the next section to identify and merge the changes.

3-Way Directory Comparison and Merge Tool

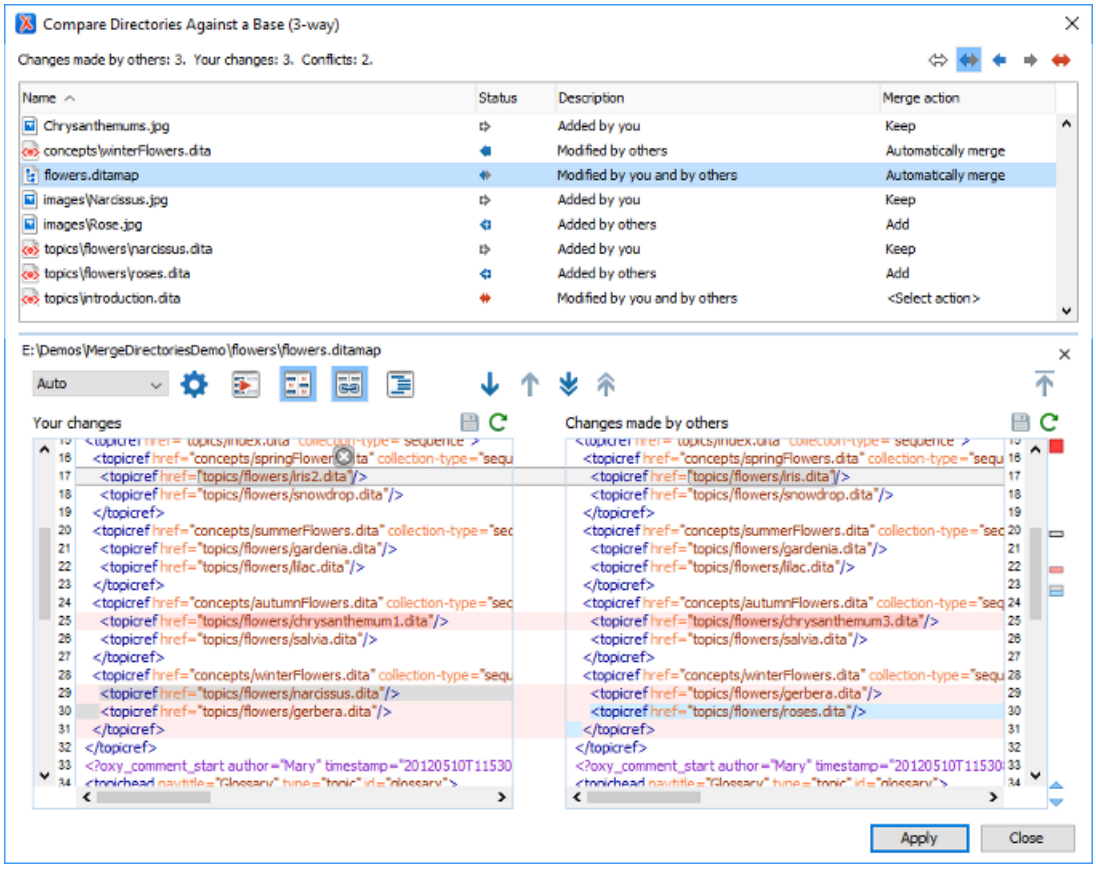


Figure 456: Comparison and Merge Tool

The 3-way directory comparison and merge tool includes the following information, features, and actions:

Number of Changes and Conflicts

The first thing you see in top-left corner of the tool is grand total of all the changes made by others, changes made by you, and the number of conflicts.

Filter Buttons

In the top-right corner you can use the toggle buttons to filter the list of modifications:

Show all files

Use this button to show all modified and unmodified files, as well as conflicts.

Show only files modified by you and others

Filters the list to show all files that have been modified, including conflicts.

Show only files modified by others

Filters the list to only show the files that were modified by others.

Show only files modified by you

Filters the list to only show the files that were modified by you.

Show only conflicting files

Filters the list to only show files that contain conflicts.

List of Files Panel

This panel shows the list of files in the compared file sets based upon the filter button that is selected. This panel includes the following sortable columns:

- **Name** - The file names.

- **Status** - An icon that represents the file status. Red icons indicate some sort of conflict. Gray icons indicate modifications made by you. Blue icons indicate modifications made by others.
- **Description** - A description of the file status.
- **Merge Action** - This column provides a drop-down menu for each file that allows you to choose some merge actions depending upon its status. A default action is always set to automatically merge the changes made by others with your changes. If there is a conflict, the default is **<Select action>** and you are required to make a selection. Click this column to access the drop-down menu where you can make a selection. The same actions are available in the contextual menu.

You can double-click any non-binary file (or select **Show modifications** from the contextual menu) to open it in the file comparison panel (the file from your file set is shown in the left panel while the file from the file set with changes made by others is shown in the right panel).

File Comparison Panels

If you double-click any non-binary file in the top panel, the file is opened in this file comparison section. The file from your file set is shown in the left panel and the file from the other file set is shown in the right panel.

Note: If Oxygen XML Editor does not recognize the file type, a dialog box will be displayed that allows you to select the type of editor you want it to be associated with for this comparison (if you want Oxygen XML Editor to remember this association, you can select the **Associate file type with editor** option at the bottom of the dialog box).

This panel includes the following information and toolbar actions:

File Path

The first thing you see in this panel is the file path where merge actions will be applied if you make changes.

Close

Closes the file comparison panel.

Algorithm Drop-Down Menu

This drop-down menu allows you to select one of the following diff algorithms to be used for file comparisons:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Diff Options

Opens the ***Files Comparison*** *preferences page* where you can configure various options.

Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.

Ignore Whitespaces

Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces.

Synchronized scrolling

Toggles synchronized scrolling. When toggled on, a selected difference can be seen in both panels.

Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on OS X))

Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.

Note: When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.

Next Block of Changes (Ctrl + Period (Command + Period on OS X))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

Note: A change block groups one or more consecutive lines that contain at least one change.

Previous Block of Changes (Ctrl + Comma (Command + Comma on OS X))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

Next Change (Ctrl + Shift + Period (Command + Shift + Period on OS X))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

Previous Change (Ctrl + Shift + Comma (Command + Shift + M on OS X))

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

First Change (Ctrl + B (Command + B on OS X))

Jumps to the first change.

Left-Side File (Your changes)

Above the panel you can see the file path and the following two buttons:

Save

Saves changes made to the file.

Reload

Reloads the file.

Right-Side File (Changes made by others)

Above the panel you can see the file path and the following two buttons:

Reload

Reloads the file.

Displaying Changes in the File Comparison Panels

The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes.

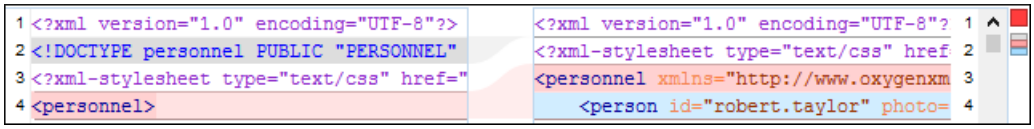


Figure 457: File Comparison Panels

The differences are also highlighted in several colors, depending on the type of change, and dynamic lines connect the compared fragments in the middle section between the two panes. The highlighting colors can be customized in the ***Files Comparison / Appearance*** *preferences page*, but the default colors and their shades mean the following:

- **Pink** - Identifies modifications on either side.
- **Gray** - Identifies an addition of a node in the left side (your outgoing changes).
- **Blue** - Identifies an addition of a node in the right side (incoming changes).
- **Lighter Shade** - Identifies blocks of changes that can be merged in their entirety.

- **Darker Shade** - Identifies specific changes within the blocks that can be merged more precisely.

Direct Editing Actions in the File Comparison Panels

In addition to selecting merge actions from the drop-down menus in the **Merge Action** column in the top panel, you can also edit the files directly in the left pane (your local changes). The two editors are constantly synchronized and the differences are refreshed when you save the modified document (**Save** button or **Ctrl +S**) or when you click the **Perform File Differencing** button.

A variety of actions are available in the contextual menu in both editing panes. The tool also includes some inline actions to help you merge, copy, or remove changes. When you select a change, the following inline action widgets are available, depending on the type of change:

Append right change to left

Copies the content of the selected change from the right side and appends it on the left side.

Copy change from right to left

Replaces the content of a change in the left side with the content of the change in the right side.

Remove change

Removes the change from the left side.

Any time you save manual changes (**Save** button or **Ctrl+S**), the selection in the **Merge Action** column in the top panel automatically changes to **Use merged** and a copy of the original file is kept so that you can revert to the original file if necessary. To discard your manual changes and revert to your original changes, select a different action in the **Merge Action** drop-down menu.

Applying Changes

When you click the **Apply** button, all the merge actions you have selected and the changes you have made will be processed.

If there are unresolved conflicts (conflicts where no merge action is selected in the **Merge Action** drop-down menu), a dialog box will be displayed that allows you to choose how to solve the conflicts. You can choose between the following:

- **Keep your changes** - If you select this option and then click **Apply**, your local changes will be preserved for the unresolved conflicts.
- **Overwrite your changes** - If you select this option and then click **Apply**, your local changes will be overwritten with the changes made by others, for the unresolved conflicts.
- **Cancel** - You can click the **Cancel** button to go back to the merge tool to resolve the conflicts individually.

Cancelling Changes

If you click the **Cancel** button at the bottom of the merge tool, all the changes you made in the tool will be lost.

Related Information:

Compare Directories on page 753
Compare Files on page 742

Document Types and Frameworks

8

Topics:

- *Predefined Document Types (Frameworks)*
- *Other Supported Document Types*

A *framework* is associated to an XML document type according to a set of rules. It also includes a variety of settings that improve editing capabilities in the **Author** mode for its particular file type. These settings include:

- A default grammar used for validation and content completion in both **Author** mode and **Text** mode.
- CSS stylesheets for rendering XML documents in **Author** mode.
- User actions invoked from toolbars or menus in **Author** mode.
- Predefined scenarios used for transformations for the class of XML documents defined by the document type.
- *XML Catalogs*.
- Directories with file templates.
- User-defined extensions for customizing the interaction with the content author in **Author** mode.

Oxygen XML Editor includes built-in support for many common document types. Each *document type is defined in a framework*. You can *create new frameworks or make changes to existing frameworks* to suit your individual requirements.

To see a video on configuring a *framework* in Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/FrameworkConfiguration.html*.

Related Information:

Advanced Framework Customization on page 1169

Document Type Configuration Dialog Box on page 59

Predefined Document Types (Frameworks)

Oxygen XML Editor includes a variety of specialized editors, views, and features that are dynamic according to the type of document that you open or create. The type of documents that are supported include the most popular predefined XML *frameworks* that include a full set of features as well as other document types that include more generic or common features.

Predefined Frameworks

The following predefined document types (*frameworks*) are fully supported in Oxygen XML Editor and each of these document types include built-in transformation scenarios, validation, content completion, file templates , default CSS files for rendering content in **Author** mode, and default actions for editing in **Author** mode:

- *DocBook 4* - A document type standard for books, articles, and other prose documents (particularly technical documentation).
- *DocBook 5* - An enhanced (version 5) document type standard designed for a variety of documents (particularly technical documentation).
- *DITA* - An XML-based architecture designed for authoring, producing, and delivering technical information.
- *DITA Map* - A document type that collects and organizes references to DITA topics or other maps.
- *XHTML* - Extensible HyperText Markup Language includes the same depth of expression as HTML, but also conforms to XML syntax.
- *TEI ODD* - Text Encoding Initiative One Document Does it all is an XML-conformant specification that allows you to create TEI P5 schema in a literate programming style.

- *TEI P5* - The Text Encoding Initiative guidelines is a standard for the academic community that collectively define an XML format for text that is primarily semantic rather than presentational.
- *JATS* - The NISO Journal Article Tag Suite is a technical standard that defines an XML format for scientific literature.

Related Information:

Advanced Framework Customization on page 1169

Document Type Association Preferences on page 57

DocBook 4 Document Type (Framework)

DocBook is a very popular set of tags for describing books, articles, and other prose documents, particularly technical documentation. DocBook provides a vast number of semantic element tags, divided into three broad categories: structural, *block-level*, and *inline*. DocBook content can then be published in a variety of formats, including HTML, PDF, WebHelp, and EPUB.

File Definition

A file is considered to be a *DocBook 4* document when one of the following conditions are true:

- The root element name is `book` or `article`.
- The PUBLIC ID of the document contains the string `-//OASIS//DTD DocBook XML`.

Default Document Templates

There are a variety of default *DocBook 4* templates available when creating *new documents from templates* and they can be found in: **Framework Templates** > **DocBook 4**.

The default templates for DocBook 4 documents are located in the `[OXYGEN_INSTALL_DIR]/frameworks/docbook/templates/Docbook 4` folder.

Default Schema for Validation and Content Completion

The default schema that is used if one is not detected in the DocBook 4 file is `docbookxi.dtd` and it is stored in `[OXYGEN_INSTALL_DIR]/frameworks/docbook/4.5/dtd/`.

Default CSS

The default CSS files used for rendering DocBook content in **Author** mode are stored in `[OXYGEN_INSTALL_DIR]/frameworks/docbook/css/`.

By default, these default CSS files are merged with any that are specified in the document. For more information, see *Configuring and Managing Multiple CSS Styles* on page 1256.

Default XML Catalog

The default *XML Catalog*, `catalog.xml`, is stored in `[OXYGEN_INSTALL_DIR]/frameworks/docbook/`.

Transformation Scenarios

Oxygen XML Editor includes numerous built-in DocBook transformation scenarios that allow you to transform DocBook 4 documents to a variety of outputs, such as WebHelp, PDF, HTML, HTML Chunk, XHTML, XHTML Chunk, and EPUB. For more information, see the *DocBook 4 Transformation Scenarios* on page 771 section.

Resources

- *Oxygen Video Tutorial: Editing DocBook Documents in Author Mode*
- *DocBook Specifications*

Related Information:

Editing XML Documents in Author Mode on page 329

Editing XML Documents in Text Mode on page 290

Adding Tables in DocBook on page 387

DocBook 4 Author Mode Actions

A variety of actions are available in the DocBook 4 *framework* that can be added to the **DocBook4** menu, the **Author Custom Actions** toolbar, the contextual menu, and the *Content Completion Assistant*.

DocBook 4 Toolbar Actions

The following default actions are readily available on the **DocBook (Author Custom Actions)** toolbar when editing in **Author** mode (by default, most of them are also available in the **DocBook4** menu and in various submenus of the contextual menu):

Bold

Emphasizes the selected text by surrounding it with `<emphasis role="bold">` tag. You can use this action on multiple non-contiguous selections.

Italic

Emphasizes the selected text by surrounding it with `<emphasis role="italic">` tag. You can use this action on multiple non-contiguous selections.

Underline

Emphasizes the selected text by surrounding it with `<emphasis role="underline">` tag. You can use this action on multiple non-contiguous selections.

Link Actions Drop-Down Menu

The following link actions are available from this menu:

Cross reference (link)

Opens a dialog box that allows you to select a target to insert as a hypertext link.

Cross reference (xref)

Inserts a cross reference to other parts of the document.

Web Link (ulink)

Inserts a link that addresses its target with a URL (Universal Resource Locator).

Insert OLink

Opens an **OLink** dialog box that allows you to insert a link that addresses its target indirectly, using the `targetdoc` and `targetptr` values that are present in a ***Targetset*** file.

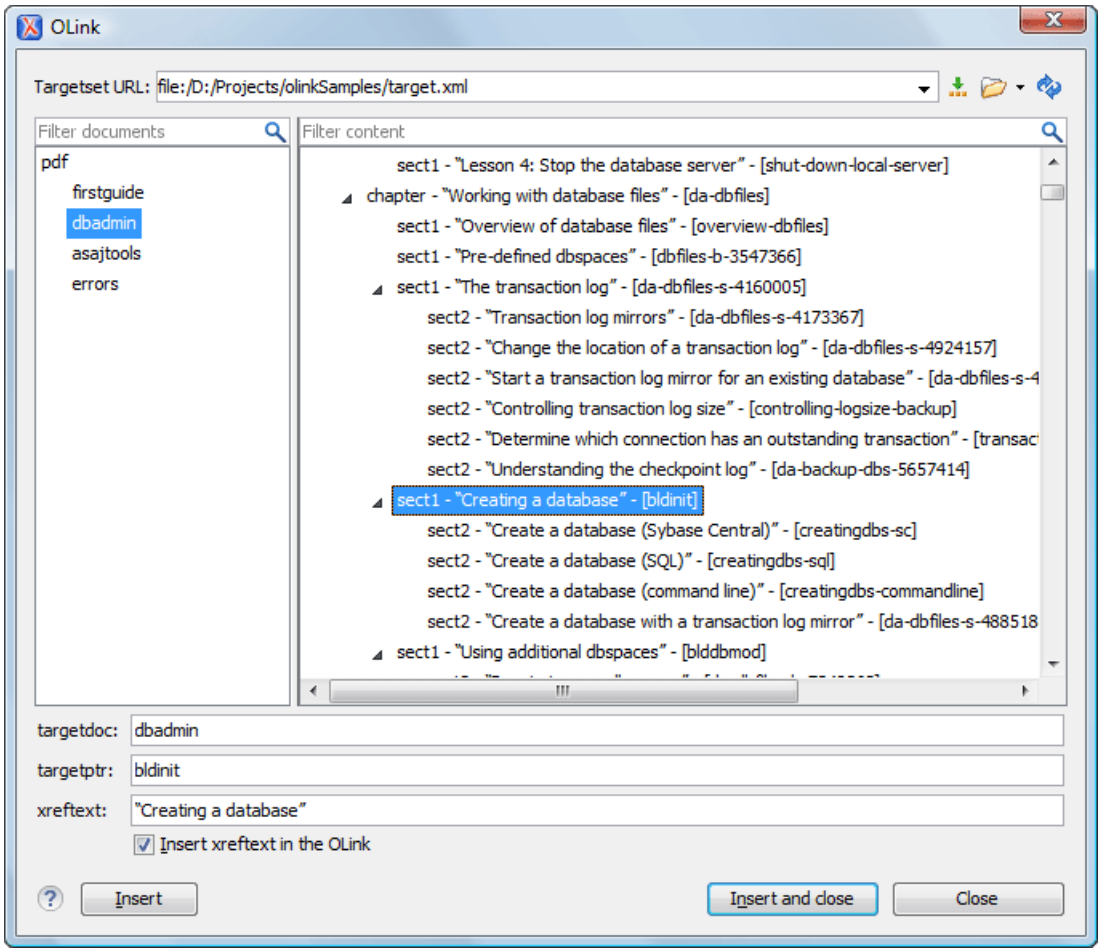


Figure 458: Insert OLink Dialog Box

After you choose the **Targetset URL**, the structure of the target documents is presented. For each target document (`targetdoc`), its content is displayed allowing you to easily identify the `targetptr` for the `olink` element that will be inserted. You can also use the search fields to quickly identify a target. If you already know the values for **targetdoc** and **targetptr**, you can insert them directly in the corresponding fields. You can also edit an `olink` using the **Edit OLink** action that is available on the contextual menu. The last used **Targetset** URL will be used to identify the edited target.

To insert XREF text into the `olink`, enter the text in the **xreftext** field and make sure the **Insert xreftext in the OLink** option is selected.

Insert URI

Inserts a URI element. The URI identifies a Uniform Resource Identifier (URI) in content.

Insert Image

Inserts an image reference at the cursor position. Depending on the current location, an image-type element is inserted.

Insert Media Resource

Opens a ***Choose Media*** *dialog box* that allows you to select the URL of a media object to be inserted into a document at the cursor position. The result will be that a reference to the specified video, audio, or embedded HTML frame is inserted and rendered in **Author** mode so that it can be played directly from there.

Insert XInclude

Opens a dialog box that allows you to browse and select content to be included and automatically generates the corresponding XInclude instruction.

Section Drop-Down Menu

The following actions are available from this menu:

Insert Section

Inserts a new section or subsection in the document, depending on the current context. For example, if the current context is `sect1`, then a `sect2` is inserted. By default, this action also inserts a `para` element as a child node. The `para` element can be deleted if it is not needed.

Promote Section (Ctrl + Alt + LeftArrow (Command + Alt + LeftArrow on OS X))

Promotes the current node as a sibling of the parent node.

Demote Section (Ctrl + Alt + RightArrow (Command + Alt + RightArrow on OS X))

Demotes the current node a child of the previous node.

Insert Paragraph

Insert a new paragraph element at current cursor position.

Insert Equation

Opens the **XML Fragment Editor** that allows you to insert and *edit MathML notations*.

Insert List Item

Inserts a list item in the current list type.

Insert Ordered List

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

Insert Itemized List

Inserts an itemized list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an itemized list.

Insert Variable List

Inserts a DocBook variable list. A child list item is also inserted automatically by default. You can also use this action to convert selected paragraphs or other types of lists to a variable list.

Insert Procedure List

Inserts a DocBook `procedure` element. A step `child` item is also inserted automatically. You can also use this action to convert selected paragraphs or other types of lists to a procedure list.

Sort

Sorts cells or list items in a table.

Insert Table

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content — text + markup — that is rendered inside a *block element*) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

Note: If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables**.

Insert Row Below

Inserts a new table row with empty cells below the current row. This action is available when the cursor is positioned inside a table.

Delete Row(s)

Deletes the table row located at cursor position or multiple rows in a selection.

Insert Column After

Inserts a new table column with empty cells after the current column. This action is available when the cursor is positioned inside a table.

Delete Column(s)

Deletes the table column located at cursor position or multiple columns in a selection.

Table Properties

Opens the **Table properties** dialog box that allows you to configure properties of a table (such as frame borders).

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

DocBook4 Menu Actions

In addition, the following default actions are available in the **DocBook4** menu when editing in **Author** mode:

Paste special submenu

This submenu includes the following special paste actions that are specific to the DocBook 4 *framework*:

Paste As XInclude

Allows you to create an `xi:include` element that references a DocBook element copied from **Author** mode. The operation fails if the copied element does not have a declared ID.

Paste as link

Allows you to create a `link` element that references a DocBook element copied from **Author** mode. The operation fails if the copied element does not have a declared ID.

Paste as xref

Allows you to create an `xref` element that references a DocBook element copied from **Author** mode. The operation fails if the copied element does not have a declared ID.

Table submenu

In addition to the table actions available on the toolbar, the following actions are available in this submenu:

Insert Row Above

Inserts a new table row with empty cells above the current row. This action is available when the cursor is positioned inside a table.

Insert Rows

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (**Above** or **Below** the current row).

Insert Column Before

Inserts a column before the current one.

Insert Columns

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (**Above** or **Below** the current column).

Insert Cell

Inserts a new empty cell depending on the current context. If the cursor is positioned between two cells, Oxygen XML Editor a new cell at cursor position. If the cursor is inside a cell, the new cell is created after the current cell.

ID Options

Opens the **ID Options** dialog box that allows you to configure options for generating IDs in **Author** mode. The dialog box includes the following:

ID Pattern

The pattern for the ID values that will be generated. This text field can be customized using constant strings or any of the Oxygen XML Editor *Editor Variables*.

Element name or class value to generate ID for

The elements for which ID values will be generated, specified using class attribute values. To customize the list, use the **Add**, **Edit**, or **Remove** buttons.

Auto generate IDs for elements

If selected, Oxygen XML Editor will automatically generate unique IDs for the elements listed in this dialog box when they are created in **Author** mode.

Remove IDs when copying content in the same document

This option allows you to control whether or not pasted elements that are listed in this dialog box should retain their existing IDs. If this option is not selected, IDs are always retained when you copy or cut content and paste it in the same document or other documents. If this option is selected, IDs are never retained for copied content, but if you cut the content, they are preserved for the first paste action (and not retained for any subsequent paste actions).

Generate IDs

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements from the list in the **ID Options** dialog box that are found in the current selection.

Note: The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `id` attribute.

Refresh References

You can use this action to manually trigger a refresh and update of all referenced resources.

Tags display mode Submenu

Full Tags with Attributes

Displays full tag names with attributes for both *block* and *inline elements*.

Full Tags

Displays full tag names without attributes for both *block* and *inline elements*.

Block Tags

Displays full tag names for *block elements* and simple tags without names for *inline elements*.

Inline Tags

Displays full tag names for *inline elements*, while *block elements* are not displayed.

Partial Tags

Displays simple tags without names for *inline elements*, while *block elements* are not displayed.

No Tags

No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Configure Tags Display Mode

Use this option to go to the ***Author*** *preferences page* where you can configure the **Tags Display Mode** options.

Profiling/Conditional Text Submenu

Edit Profiling Attributes

Allows you to configure the *profiling attributes* and their values.

Show Profiling Colors and Styles

Select this option to turn on conditional styling.

Show Profiling Attributes

Select this option to turn on conditional text markers. They are displayed at the end of conditional text blocks, as a list of attribute name and their currently set values.

Show Excluded Content

When this option is selected, the content filtered by the currently applied condition set is grayed-out. To show only the content that matches the currently applied condition set, deselect this option.

List of all profiling condition sets that match the current document type

You can click a listed *condition set* to activate it.

Profiling Settings

Opens the *profiling options preferences page*, where you can manage profiling attributes and profiling conditions sets. You can also configure the profiling styles and colors options from the *colors/styles preferences page* and the *attributes rendering preferences page*.

DocBook 4 Contextual Menu Actions

In addition to many of the *DocBook 4 toolbar actions* and the *general **Author** mode contextual menu actions*, the following DocBook 4 *framework*-specific actions are also available in the contextual menu when editing in **Author** mode:

Paste special submenu

This submenu includes the following special paste actions that are specific to the DocBook 4 *framework*:

Paste As XInclude

Allows you to create an `xi:include` element that references a DocBook element copied from **Author** mode. The operation fails if the copied element does not have a declared ID.

Paste as link

Allows you to create a `link` element that references a DocBook element copied from **Author** mode. The operation fails if the copied element does not have a declared ID.

Paste as xref

Allows you to create an `xref` element that references a DocBook element copied from **Author** mode. The operation fails if the copied element does not have a declared ID.

Image Map Editor

This action is available in the contextual menu when it is invoked on an image. This action applies an *image map* to the current image (if one does not already exist) and opens the **Image Map Editor** dialog box. This feature allows you to create hyperlinks in specific areas of an image that will link to various destinations.

Table Actions

The following table editing actions are available in the contextual menu when it is invoked on a table:

Insert Rows

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (**Above** or **Below** the current row).

Delete Row(s)

Deletes the table row located at cursor position or multiple rows in a selection.

Insert Columns

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (**Above** or **Below** the current column).

Delete Column(s)

Deletes the table column located at cursor position or multiple columns in a selection.

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

Sort

Sorts cells or list items in a table.

Table Properties

Opens the **Table properties** dialog box that allows you to configure properties of a table (such as frame borders).

Other Actions submenu

This submenu give you access to all the usual contextual menu actions.

Link > Edit OLink

Opens a dialog box that allows you edit an existing OLink. See the **Insert OLink** action for more information.

Generate IDs

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements from the list in the **ID Options** dialog box that are found in the current selection.

Note: The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `id` attribute.

DocBook 4 Drag/Drop (or Copy/Paste) Actions

Dragging a file from the ***Project*** *view* or ***DITA Maps Manager*** *view* and dropping it into a DocBook 4 document that is edited in **Author** mode, creates a link to the dragged file (the `ulink` DocBook element) at the drop location. Copy and paste actions work the same.

You can also drag images or media files from your system explorer or the ***Project*** *view* and drop them into a DocBook 4 document (or copy and paste). This will insert the appropriate element at the drop or paste location (for example, dropping/pasting an image will insert the `inlinegraphic` DocBook element with a `fileref` attribute).

DocBook 4 Transformation Scenarios

Default transformation scenarios allow you to convert DocBook 4 to DocBook 5 documents and transform DocBook documents to a variety of outputs, such as WebHelp, PDF, HTML, HTML Chunk, XHTML, XHTML Chunk, and EPUB. All of them are listed in the **DocBook 4** section in the ***Configure Transformation Scenario(s)*** *dialog box*.

Related Information:

Editing a Transformation Scenario on page 925
Configure Transformation Scenario(s) Dialog Box on page 926

DocBook 4 to WebHelp Output

DocBook 4 documents can be transformed into several types of WebHelp systems.

WebHelp Classic Output

To publish a DocBook 4 document as a **WebHelp Classic** system, follow these steps:

1. Click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
2. Select the **DocBook WebHelp Classic** scenario from the **DocBook 4** section.
3. Click **Apply associated**.

   When the **DocBook WebHelp Classic** transformation is complete, the output is automatically opened in your default browser.

WebHelp Classic with Feedback Output

To publish a DocBook 4 document as a **WebHelp Classic with Feedback** system, follow these steps:

1. Click **Configure Transformation Scenarios**.
2. Select the **DocBook WebHelp Classic with Feedback** scenario from the **DocBook 4** section.
3. Click **Apply associated**.
4. Enter the documentation product ID and the documentation version.

   When the **DocBook WebHelp Classic with Feedback** transformation is complete, your default browser opens the `installation.html` file. This file contains information about the output location, system requirements, installation instructions, and deployment of the output. Follow the instructions to complete the system deployment. For more information, see *Deploying a Feedback-Enabled WebHelp System* on page 947.

To watch our video demonstration about the feedback-enabled WebHelp system, go to *https://www.oxygenxml.com/demo/Feedback_Enabled_WebHelp.html*.

WebHelp Classic Mobile Output

To publish a DocBook 4 document as a **WebHelp Classic Mobile** system, follow these steps:

1. Click **Configure Transformation Scenarios**.
2. Select the **DocBook WebHelp Classic Mobile** scenario from the **DocBook 4** section.
3. Click **Apply associated**.

When the **DocBook WebHelp Classic Mobile** transformation is complete, the output is automatically opened in your default browser.

Customizing WebHelp Transformation Scenarios

To customize a DocBook WebHelp transformation scenario, you can edit various parameters, including the following most commonly used ones:

args.default.language

This parameter is used if the language is not detected in the *DITA map*. The default value is `en-us`.

clean.output

Deletes all files from the output folder before the transformation is performed (only no and `yes` values are valid and the default value is no).

l10n.gentext.default.language

This parameter is used to identify the correct stemmer that differs from language to language. For example, for English the value of this parameter is en or for French it is `fr`, and so on.

use.stemming

Controls whether or not you want to include stemming search algorithms into the published output (default setting is `false`).

webhelp.copyright

Adds a small copyright text that appears at the end of the **Table of Contents** pane.

webhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

webhelp.favicon

The file path that points to an image to be used as a *favicon* in the WebHelp output.

webhelp.footer.file

Path to an XML file that includes the footer content for your WebHelp output pages. You can use this parameter to integrate social media features (such as widgets for Facebook™, Twitter™, Google Analytics, or Google+™). The file must be well-formed, each widget must be in separate `div` or `span` element, and the code for each `script` element is included in an XML comment (also, the start and end tags for the XML comment must be on a separate line). The following code exert is an example for adding a Facebook™ widget:

```
<div id="facebook">
  <div id="fb-root"/>
  <script>
    <!-- (function(d, s, id) { var js, fjs = d.getElementsByTagName(s)[0];
      if (d.getElementById(id)) return;
      js = d.createElement(s); js.id = id;
      js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
      fjs.parentNode.insertBefore(js, fjs); }
       (document, 'script', 'facebook-jssdk')); -->
    </script>
    <div data-share="true" data-show-faces="true" data-action="like"
       data-layout="standard" class="fb-like"/>
</div>
```

webhelp.footer.include

Specifies whether or not to include footer in each WebHelp page. Possible values: `yes`, no. If set to no, no footer is added to the WebHelp pages. If set to `yes` and the `webhelp.footer.file` parameter has a value, then the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default Oxygen XML Editor footer is inserted in each WebHelp page.

webhelp.logo.image.target.url

Specifies a target URL that is set on the logo image. When you click the logo image, you will be redirected to this address.

webhelp.logo.image

Specifies a path to an image displayed as a logo in the left side of the output header.

webhelp.product.id (available only for Feedback-enabled systems)

This parameter specifies a short name for the documentation target, or product (for example, `mobile-phone-user-guide`, `hvac-installation-guide`).

Note: You can deploy documentation for multiple products on the same server.

Restriction: The following characters are not allowed in the value of this parameter: `< > / \ ' ( ) { } = ; * % + &`.

webhelp.product.version (available only for Feedback-enabled systems)

Specifies the documentation version number (for example, 1.0, 2.5, etc.). New user comments are bound to this version.

Note: Multiple documentation versions can be deployed on the same server.

Restriction: The following characters are not allowed in the value of this parameter: `< > / \ ' ( ) { } = ; * % + &`.

webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Editor uses for indexing Japanese content in the WebHelp pages.

webhelp.search.ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the search results that are displayed on the **Search** tab (default setting is `true`).

webhelp.skin.css

Path to a CSS file that sets the style theme in the output WebHelp pages. It can be one of the predefined skin CSS from the `OXYGEN_INSTALL_DIR\frameworks\docbook\xsl\com.oxygenxml.webhelp`

`\predefined-skins` directory, or it can be a custom skin CSS generated with the *Oxygen Skin Builder* web application.

For more information about all the DocBook transformation parameters, go to *http://docbook.sourceforge.net/release/xsl/current/doc/fo/index.html*.

Related Information:

WebHelp Classic Output on page 815
WebHelp Classic With Feedback Output on page 818
WebHelp Classic Mobile System (Deprecated) on page 1023
Customizing WebHelp Classic Systems on page 999
Customizing WebHelp Classic Mobile Systems on page 1024

DocBook to PDF Output Customization

When the default layout and output of the DocBook to PDF transformation needs to be customized, follow these steps:

1. Create a custom version of the DocBook title spec file.

   You should start from a copy of the file *[OXYGEN_INSTALL_DIR]*`/frameworks/docbook/xsl/fo/titlepage.templates.xml` and customize it. The instructions for the spec file can be found *here.*

   An example of spec file:

```
<t:titlepage-content t:side="recto">
    <mediaobject/>
    <title
        t:named-template="book.verso.title"
        font-size="&hsize2;"
        font-weight="bold"
        font-family="{$title.font.family}"/>
    <corpauthor/>
      ...
</t:titlepage-content>
```

2. Generate a new XSLT stylesheet from the title spec file from the previous step.

   Apply *[OXYGEN_INSTALL_DIR]*`/frameworks/docbook/xsl/template/titlepage.xsl` to the title spec file. The result is an XSLT stylesheet (for example, `mytitlepages.xsl`).

3. Import `mytitlepages.xsl` in a *DocBook customization layer*.

   The customization layer is the stylesheet that will be applied to the XML document. The `mytitlepages.xsl` should be imported with an element like this:

```
<xsl:import href="dir-name/mytitlepages.xsl"/>
```

4. Insert a logo image in the XML document.

   The path to the logo image must be inserted in the `book/info/mediaobject` element of the XML document.

5. Apply the customization layer to the XML document.

   A quick way is to duplicate the transformation scenario **DocBook PDF** that is included with Oxygen XML Editor and set the customization layer in *the **XSL URL** property of the scenario*.

Related Information:

*The book **DocBook XSL: The Complete Guide** by Bob Stayton contains more details about customizing the PDF output.*
Video demonstration for creating a DocBook customization layer in Oxygen XML Editor.

DocBook to EPUB Transformation

The EPUB specification recommends the use of *OpenType* fonts (recognized by their `.otf` file extension) when possible. To use a specific font, follow these steps:

1. Declare it in your CSS file, as in the following example:

```
@font-face {
font-family: "MyFont";
font-weight: bold;
```

```
font-style: normal;
src: url(fonts/MyFont.otf);
}
```

2. In the CSS, specify where this font is used. To set it as default for h1 elements, use the `font-family` rule, as in the following example:

```
h1 {
font-size:20pt;
margin-bottom:20px;
font-weight: bold;
font-family: "MyFont";
text-align: center;
}
```

3. Open the ***Configure Transformation Scenario(s)*** *dialog box*, select the **DocBook EPUB** transformation scenario, and click **Edit**.
4. In the **Parameters** tab, set the `epub.embedded.fonts` parameter to `fonts/MyFont.otf`. If you need to provide more files, use commas to separate their file paths.

   Note: The `html.stylesheet` parameter allows you to include a custom CSS in the output EPUB.
5. Run the transformation scenario.

DocBook to DITA Transformation

Oxygen XML Editor includes a built-in transformation scenario that is designed to convert DocBook content to DITA. This transformation scenario is based upon a DITA Open Toolkit plugin that is available at *sourceforge.net*.

To convert a DocBook document to DITA, follow these steps:

1. Use one of the following two methods to begin the transformation process:
   - To apply the transformation scenario to a newly opened file, use the **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on OS X))** action from the toolbar or the **Document** > **Transformation** menu.
   - To customize the transformation or change the scenario that is associated with the document, use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu.
2. Select the **DocBook to DITA** transformation scenario in the **DocBook 4** or **DocBook 5** section.
3. Click the **Apply associated** button to run the transformation.

   Step Result: The transformation will convert as many of the DocBook elements into equivalent DITA elements as it can recognize in its mapping process. For elements that cannot be mapped, the transformation will insert XML comments so that you can see which elements could not be converted.
4. Adjust the resulting DITA composite to suit your needs. You may have to remove comments, fix validation errors, adjust certain attributes, or split the content into individual topics.

Related Information:

Editing a Transformation Scenario on page 925

Configure Transformation Scenario(s) Dialog Box on page 926

Inserting an `olink` in DocBook Documents

The `olink` element is used for linking to resources outside the current DocBook document. The `targetdoc` attribute is used for the document ID that contains the target element and the `targetptr` attribute for the ID of the target element (the value of an `id` or `xml:id` attribute). The combination of those two attributes provides a unique identifier to locate cross references.

For example, a *Mail Administrator Guide* with the document ID `MailAdminGuide` might contain a chapter about user accounts, like this:

```
<chapter id="user_accounts">
<title>Administering User Accounts</title>
<para>blah blah</para>
```

You can form a cross reference to that chapter by adding an `olink`, as in the following example:

```
You may need to update your
```

```
<olink targetdoc="MailAdminGuide" targetptr="user_accounts">user accounts
</olink>
when you get a new machine.
```

To use an `olink` to create links between documents, follow these steps:

1. Decide which documents are to be included in the domain for cross referencing.

   A unique ID must be assigned to each document that will be referenced with an `olink`. It is usually added as an `id` (or `xml:id` for DocBook5) attribute to the root element of the document.

2. Decide on your output hierarchy.

   For creating links between documents, the relative locations of the output documents must be known. Before going further you must decide the names and locations of the output directories for all the documents from the domain. Each directory will be represented by an element: `<dir name="directory_name">`, in the target database document.

3. Create the target database document.

   Each collection of documents has a master target database document that is used to resolve all `olinks` from that collection. The target database document is an XML file that is created once. It provides a means for pulling in the target data for each document. The database document is static and all the document data is pulled in dynamically.

   Example: The following is an example of a target database document. It structures a collection of documents in a `sitemap` element that provides the relative locations of the outputs (HTML in this example). Then it pulls in the individual target data using system entity references to target data files that will be created in the next step.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE targetset [
<!ENTITY ugtargets SYSTEM "file:///doc/userguide/target.db">
<!ENTITY agtargets SYSTEM "file:///doc/adminguide/target.db">
<!ENTITY reftargets SYSTEM "file:///doc/man/target.db">
]>
<targetset>
  <targetsetinfo>
    Description of this target database document,
    which is for the examples in olink doc.
  </targetsetinfo>

  <!-- Site map for generating relative paths between documents -->
  <sitemap>
    <dir name="documentation">
      <dir name="guides">
        <dir name="mailuser">
          <document targetdoc="MailUserGuide"
                    baseuri="userguide.html">
            &ugtargets;
          </document>
        </dir>
        <dir name="mailadmin">
          <document targetdoc="MailAdminGuide">
            &agtargets;
          </document>
        </dir>
      </dir>
      <dir name="reference">
        <dir name="mailref">
          <document targetdoc="MailReference">
            &reftargets;
          </document>
        </dir>
      </dir>
    </dir>
  </sitemap>
</targetset>
```

4. Generate the target data files by executing a DocBook transformation scenario.

   Before applying the transformation, you need to edit the transformation scenario, go to the **Parameters** tab, and make sure the value of the `collect.xref.targets` parameter is set to `yes`. The default name of a target data file is `target.db`, but it can be changed by setting an absolute file path in the `targets.filename` parameter.

   Example: An example of a `target.db` file:

```
<div element="book" href="#MailAdminGuide" number="1" targetptr="user_accounts">
 <ttl>Administering User Accounts</ttl>
 <xreftext>How to administer user accounts</xreftext>
 <div element="part" href="#d5e4" number="I">
  <ttl>First Part</ttl>
```

```
  <xreftext>Part I, "First Part"</xreftext>
  <div element="chapter" href="#d5e6" number="1">
    <ttl>Chapter Title</ttl>
    <xreftext>Chapter 1, Chapter Title</xreftext>
    <div element="sect1" href="#src_chapter" number="1" targetptr="src_chapter">
      <ttl>Section1 Title</ttl>
      <xreftext>xreflabel_here</xreftext>
    </div>
  </div>
 </div>
</div>
```

5. Insert `olink` elements in the DocBook documents.

   When editing a DocBook XML document in **Author** mode, the **Insert OLink** action is available in the **Link** drop-down menu from the toolbar. This action opens the **Insert OLink** dialog box that allows you to select the target of an `olink` from the list of all possible targets from a specified target database document (specified in the **Targetset URL** field). Once a **Targetset URL** is selected, the structure of the target documents is presented. For each target document (`targetdoc`), its content is displayed, allowing you to easily identify the appropriate `targetptr`. You can also use the search fields to quickly identify a target. If you already know the values for the `targetdoc` and `targetptr` attributes, you can insert them directly in the corresponding fields.

   Example: In the following image, the target database document is called `target.xml`, *dbadmin* is selected for the target document (`targetdoc`), and *bldinit* is selected as the value for the `targetptr` attribute. Notice that you can also add XREF text into the `olink` by using the `xreftext` field.

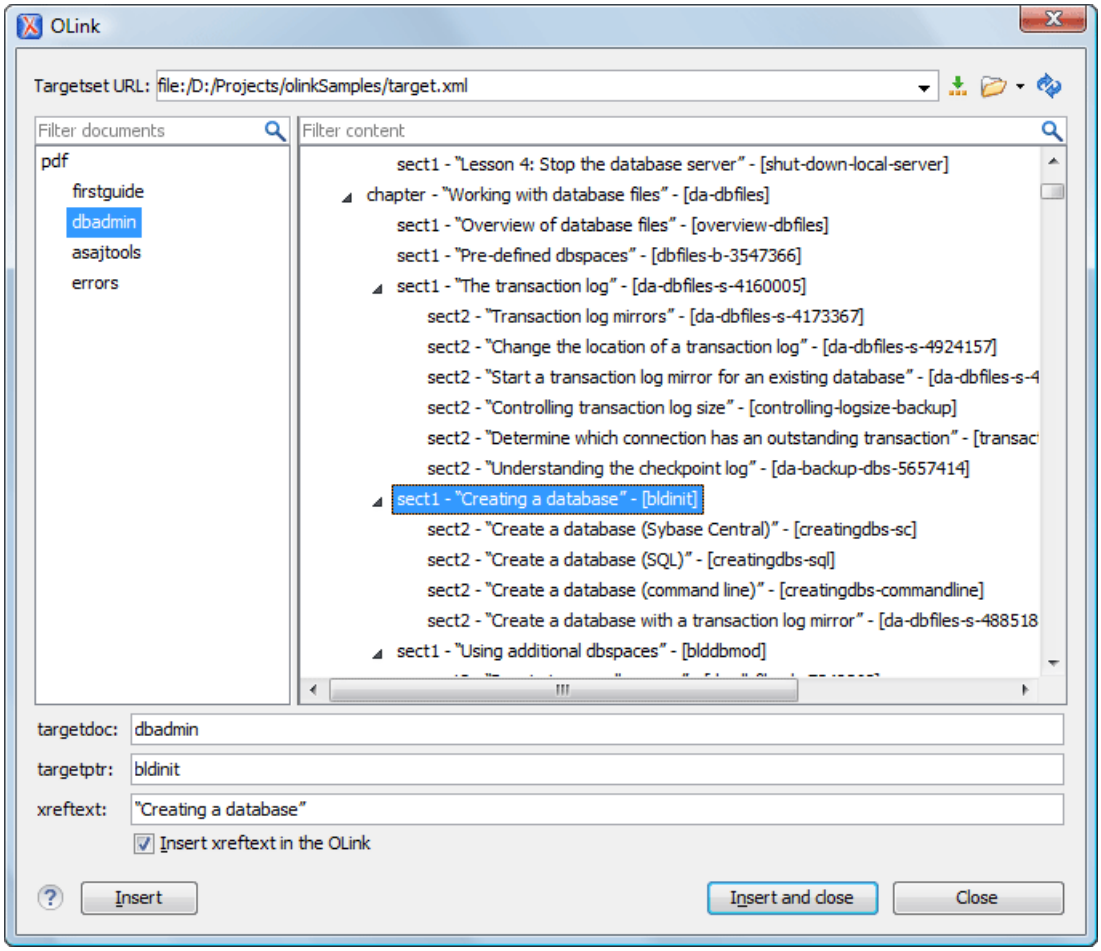


Figure 459: Insert OLink Dialog Box

6. Process a DocBook transformation for each document to generate the output.
   a) Edit the transformation scenario and set the value of the `target.database.document` parameter to be the URL of the target database document.
   b) Apply the transformation scenario.

DocBook 5 Document Type (Framework)

DocBook is a very popular set of tags for describing books, articles, and other prose documents, particularly technical documentation. DocBook provides a vast number of semantic element tags, divided into three broad categories: structural, *block-level*, and *inline*. DocBook content can then be published in a variety of formats, including HTML, PDF, WebHelp, and EPUB.

File Definition

A file is considered to be a DocBook 5 document when the namespace is *http://docbook.org/ns/docbook*.

Default Document Templates

There are a variety of default *DocBook 5* templates available when creating *new documents from templates* and they can be found in: **Framework Templates** > **DocBook 5** > **DocBook 5.0** and **Framework Templates** > **DocBook 5** > **DocBook 5.1**.

New file templates for both DocBook 5 documents are located in the `[OXYGEN_INSTALL_DIR]/frameworks/docbook/templates/Docbook5.0` folder.

New file templates for both DocBook 5.1 documents are located in the `[OXYGEN_INSTALL_DIR]/frameworks/docbook/templates/Docbook5.1` folder.

Default Schema for Validation and Content Completion

The default schema that is used if one is not detected is `docbookxi.rng` and it is stored in `[OXYGEN_INSTALL_DIR]/frameworks/docbook/5.0/rng/` (or for DocBook 5.1 in `[OXYGEN_INSTALL_DIR]/frameworks/docbook/5.1/rng/`). Other types of schemas for various DocBook versions are also located in various folders inside the `[OXYGEN_INSTALL_DIR]/frameworks/docbook/` directory.

Default CSS

The default CSS files used for rendering DocBook content in **Author** mode is stored in `[OXYGEN_INSTALL_DIR]/frameworks/docbook/css/`.

By default, these default CSS files are merged with any that are specified in the document. For more information, see *Configuring and Managing Multiple CSS Styles* on page 1256.

Transformation Scenarios

Oxygen XML Editor includes numerous built-in DocBook transformation scenarios that allow you to transform DocBook 5 documents to a variety of outputs, such as WebHelp, PDF, HTML, HTML Chunk, XHTML, XHTML Chunk, EPUB, and Assembly (5.1). For more information, see the *DocBook 5 Transformation Scenarios* on page 785 section.

Resources

- *Oxygen Video Tutorial: Editing DocBook Documents in Author Mode*
- *DocBook 5.0 (and older) Specifications*
- *Docbook 5.1 Specifications*
- *DocBook 5.1: The Definitive Guide*

Related Information:

Editing XML Documents in Author Mode on page 329
Editing XML Documents in Text Mode on page 290
Adding Tables in DocBook on page 387
DocBook 5.1 Assembly on page 792
DocBook 5.1 Topic on page 793

DocBook 5 Author Mode Actions

A variety of actions are available in the DocBook 5 *framework* that can be added to the **DocBook5** menu, the **Author Custom Actions** toolbar, the contextual menu, and the *Content Completion Assistant*.

DocBook 5 Toolbar Actions

The following default actions are readily available on the **DocBook (Author Custom Actions)** toolbar when editing in **Author** mode (by default, most of them are also available in the **DocBook5** menu and in various submenus of the contextual menu):

Bold

Emphasizes the selected text by surrounding it with `<emphasis role="bold">` tag. You can use this action on multiple non-contiguous selections.

Italic

Emphasizes the selected text by surrounding it with `<emphasis role="italic">` tag. You can use this action on multiple non-contiguous selections.

Underline

Emphasizes the selected text by surrounding it with `<emphasis role="underline">` tag. You can use this action on multiple non-contiguous selections.

Link Actions Drop-Down Menu

The following link actions are available from this menu:

Cross reference (link)

Opens a dialog box that allows you to select a target to insert as a hypertext link.

Cross reference (xref)

Inserts a cross reference to other parts of the document.

Web Link (ulink)

Inserts a link that addresses its target with a URL (Universal Resource Locator).

Insert OLink

Opens an **OLink** dialog box that allows you to insert a link that addresses its target indirectly, using the `targetdoc` and `targetptr` values that are present in a ***Targetset*** file.

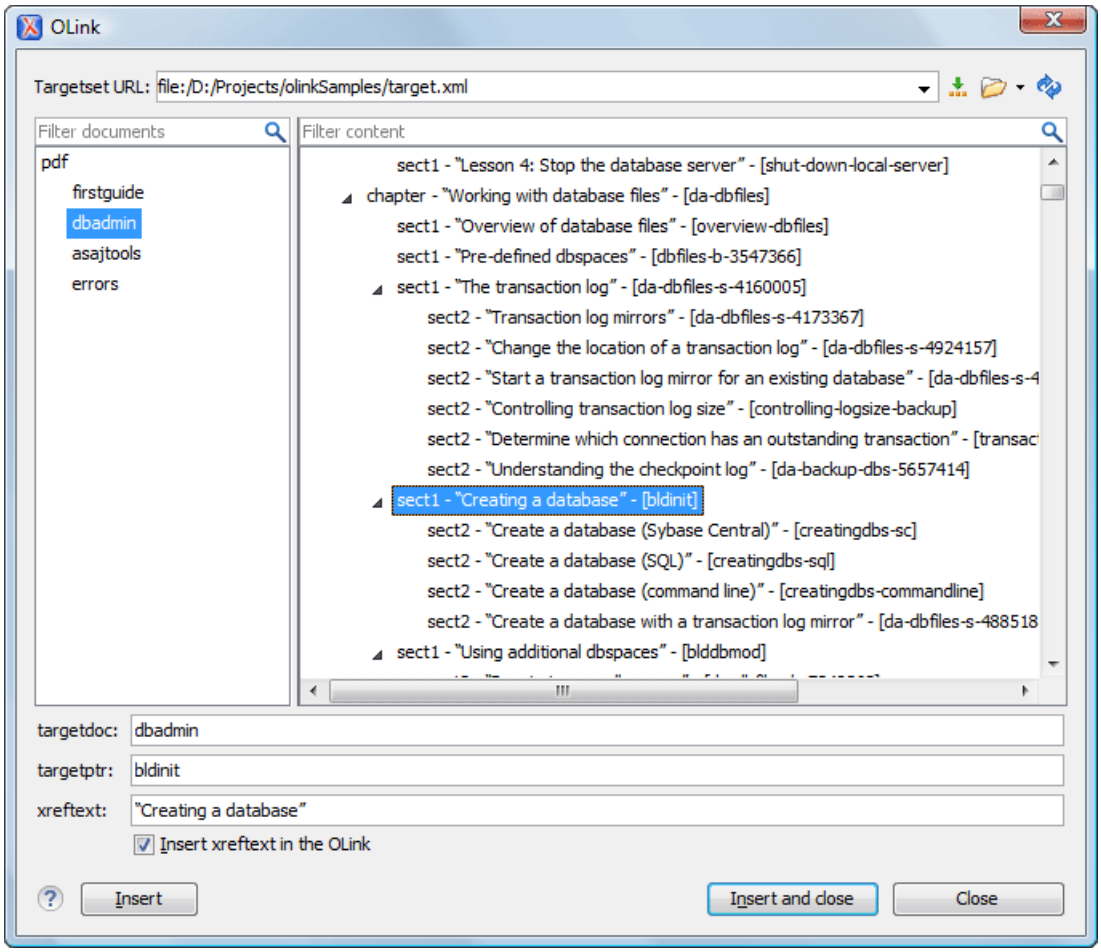


Figure 460: Insert OLink Dialog Box

After you choose the **Targetset URL**, the structure of the target documents is presented. For each target document (`targetdoc`), its content is displayed allowing you to easily identify the `targetptr` for the `olink` element that will be inserted. You can also use the search fields to quickly identify a target. If you already know the values for **targetdoc** and **targetptr**, you can insert them directly in the corresponding fields. You can also edit an `olink` using the **Edit OLink** action that is available on the contextual menu. The last used **Targetset** URL will be used to identify the edited target.

To insert XREF text into the `olink`, enter the text in the **xreftext** field and make sure the **Insert xreftext in the OLink** option is selected.

Insert URI

Inserts a URI element. The URI identifies a Uniform Resource Identifier (URI) in content.

Insert Image

Inserts an image reference at the cursor position. Depending on the current location, an image-type element is inserted.

Insert Media Resource

Opens a ***Choose Media*** *dialog box* that allows you to select the URL of a media object to be inserted into a document at the cursor position. The result will be that a reference to the specified video, audio, or embedded HTML frame is inserted and rendered in **Author** mode so that it can be played directly from there.

Insert XInclude

Opens a dialog box that allows you to browse and select content to be included and automatically generates the corresponding XInclude instruction.

Section Drop-Down Menu

The following actions are available from this menu:

Insert Section

Inserts a new section or subsection in the document, depending on the current context. For example, if the current context is `sect1`, then a `sect2` is inserted. By default, this action also inserts a `para` element as a child node. The `para` element can be deleted if it is not needed.

Promote Section (Ctrl + Alt + LeftArrow (Command + Alt + LeftArrow on OS X))

Promotes the current node as a sibling of the parent node.

Demote Section (Ctrl + Alt + RightArrow (Command + Alt + RightArrow on OS X))

Demotes the current node a child of the previous node.

Insert Paragraph

Insert a new paragraph element at current cursor position.

Insert Equation

Opens the **XML Fragment Editor** that allows you to insert and *edit MathML notations*.

Insert List Item

Inserts a list item in the current list type.

Insert Ordered List

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

Insert Itemized List

Inserts an itemized list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an itemized list.

Insert Variable List

Inserts a DocBook variable list. A child list item is also inserted automatically by default. You can also use this action to convert selected paragraphs or other types of lists to a variable list.

Insert Procedure List

Inserts a DocBook `procedure` element. A step `child` item is also inserted automatically. You can also use this action to convert selected paragraphs or other types of lists to a procedure list.

Sort

Sorts cells or list items in a table.

Insert Table

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content — text + markup — that is rendered inside a *block element*) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

Note: If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables**.

Insert Row Below

Inserts a new table row with empty cells below the current row. This action is available when the cursor is positioned inside a table.

Delete Row(s)

Deletes the table row located at cursor position or multiple rows in a selection.

Insert Column After

Inserts a new table column with empty cells after the current column. This action is available when the cursor is positioned inside a table.

Delete Column(s)

Deletes the table column located at cursor position or multiple columns in a selection.

Table Properties

Opens the **Table properties** dialog box that allows you to configure properties of a table (such as frame borders).

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

DocBook5 Menu Actions

In addition, the following default actions are available in the **DocBook5** menu when editing in **Author** mode:

Paste special submenu

This submenu includes the following special paste actions that are specific to the DocBook 5 *framework*:

Paste As XInclude

Allows you to create an `xi:include` element that references a DocBook element copied from **Author** mode. The operation fails if the copied element does not have a declared ID.

Paste as link

Allows you to create a `link` element that references a DocBook element copied from **Author** mode. The operation fails if the copied element does not have a declared ID.

Paste as xref

Allows you to create an `xref` element that references a DocBook element copied from **Author** mode. The operation fails if the copied element does not have a declared ID.

Table submenu

In addition to the table actions available on the toolbar, the following actions are available in this submenu:

Insert Row Above

Inserts a new table row with empty cells above the current row. This action is available when the cursor is positioned inside a table.

Insert Rows

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (**Above** or **Below** the current row).

Insert Column Before

Inserts a column before the current one.

Insert Columns

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (**Above** or **Below** the current column).

Insert Cell

Inserts a new empty cell depending on the current context. If the cursor is positioned between two cells, Oxygen XML Editor a new cell at cursor position. If the cursor is inside a cell, the new cell is created after the current cell.

ID Options

Opens the **ID Options** dialog box that allows you to configure options for generating IDs in **Author** mode. The dialog box includes the following:

ID Pattern

The pattern for the ID values that will be generated. This text field can be customized using constant strings or any of the Oxygen XML Editor *Editor Variables*.

Element name or class value to generate ID for

The elements for which ID values will be generated, specified using class attribute values. To customize the list, use the **Add**, **Edit**, or **Remove** buttons.

Auto generate IDs for elements

If selected, Oxygen XML Editor will automatically generate unique IDs for the elements listed in this dialog box when they are created in **Author** mode.

Remove IDs when copying content in the same document

This option allows you to control whether or not pasted elements that are listed in this dialog box should retain their existing IDs. If this option is not selected, IDs are always retained when you copy or cut content and paste it in the same document or other documents. If this option is selected, IDs are never retained for copied content, but if you cut the content, they are preserved for the first paste action (and not retained for any subsequent paste actions).

Generate IDs

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements from the list in the **ID Options** dialog box that are found in the current selection.

Note: The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `id` attribute.

Refresh References

You can use this action to manually trigger a refresh and update of all referenced resources.

Tags display mode Submenu

Full Tags with Attributes

Displays full tag names with attributes for both *block* and *inline elements*.

Full Tags

Displays full tag names without attributes for both *block* and *inline elements*.

Block Tags

Displays full tag names for *block elements* and simple tags without names for *inline elements*.

Inline Tags

Displays full tag names for *inline elements*, while *block elements* are not displayed.

Partial Tags

Displays simple tags without names for *inline elements*, while *block elements* are not displayed.

No Tags

No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Configure Tags Display Mode

Use this option to go to the ***Author*** *preferences page* where you can configure the **Tags Display Mode** options.

Profiling/Conditional Text Submenu

Edit Profiling Attributes

Allows you to configure the *profiling attributes* and their values.

Show Profiling Colors and Styles

Select this option to turn on conditional styling.

Show Profiling Attributes

Select this option to turn on conditional text markers. They are displayed at the end of conditional text blocks, as a list of attribute name and their currently set values.

Show Excluded Content

When this option is selected, the content filtered by the currently applied condition set is grayed-out. To show only the content that matches the currently applied condition set, deselect this option.

List of all profiling condition sets that match the current document type

You can click a listed *condition set* to activate it.

Profiling Settings

Opens the *profiling options preferences page*, where you can manage profiling attributes and profiling conditions sets. You can also configure the profiling styles and colors options from the *colors/styles preferences page* and the *attributes rendering preferences page*.

DocBook 5 Contextual Menu Actions

In addition to many of the *DocBook 5 toolbar actions* and the *general **Author** mode contextual menu actions*, the following DocBook 5 *framework*-specific actions are also available in the contextual menu when editing in **Author** mode:

Paste special submenu

This submenu includes the following special paste actions that are specific to the DocBook 5 *framework*:

Paste As XInclude

Allows you to create an `xi:include` element that references a DocBook element copied from **Author** mode. The operation fails if the copied element does not have a declared ID.

Paste as link

Allows you to create a `link` element that references a DocBook element copied from **Author** mode. The operation fails if the copied element does not have a declared ID.

Paste as xref

Allows you to create an `xref` element that references a DocBook element copied from **Author** mode. The operation fails if the copied element does not have a declared ID.

Image Map Editor

This action is available in the contextual menu when it is invoked on an image. This action applies an *image map* to the current image (if one does not already exist) and opens the **Image Map Editor** dialog box. This feature allows you to create hyperlinks in specific areas of an image that will link to various destinations.

Table Actions

The following table editing actions are available in the contextual menu when it is invoked on a table:

Insert Rows

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (**Above** or **Below** the current row).

Delete Row(s)

Deletes the table row located at cursor position or multiple rows in a selection.

Insert Columns

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (**Above** or **Below** the current column).

Delete Column(s)

Deletes the table column located at cursor position or multiple columns in a selection.

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

Sort

Sorts cells or list items in a table.

Table Properties

Opens the **Table properties** dialog box that allows you to configure properties of a table (such as frame borders).

Other Actions submenu

This submenu give you access to all the usual contextual menu actions.

Link > Edit OLink

Opens a dialog box that allows you edit an existing OLink. See the **Insert OLink** action for more information.

Generate IDs

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements from the list in the **ID Options** dialog box that are found in the current selection.

Note: The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `id` attribute.

DocBook 5 Drag/Drop (or Copy/Paste) Actions

Dragging a file from the ***Project*** *view* or ***DITA Maps Manager*** *view* and dropping it into a DocBook 5 document that is edited in **Author** mode, creates a link to the dragged file (the `link` DocBook element) at the drop location. Copy and paste actions work the same.

You can also drag images or media files from your system explorer or the ***Project*** *view* and drop them into a DocBook 5 document (or copy and paste). This will insert the appropriate element at the drop or paste location (for example, dropping/pasting an image will insert the `imageobject` DocBook element with an `imagedata` child element and a `fileref` attribute).

DocBook 5 Transformation Scenarios

Built-in transformation scenarios allow you to transform DocBook 5 documents to a variety of outputs, such as WebHelp, PDF, HTML, HTML Chunk, XHTML, XHTML Chunk, and EPUB. Oxygen XML Editor also includes a DocBook 5.1 transformation scenario for *Assembly* documents. All of them are listed in the **DocBook 5** section in the ***Configure Transformation Scenario(s)*** *dialog box*.

Related Information:

Editing a Transformation Scenario on page 925
Configure Transformation Scenario(s) Dialog Box on page 926

DocBook 5 to WebHelp Output

DocBook 5 documents can be transformed into several types of WebHelp systems.

WebHelp Classic Output

To publish a DocBook 5 document as a **WebHelp Classic** system, follow these steps:

1. Click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
2. Select the **DocBook WebHelp Classic** scenario from the **DocBook 5** section.
3. Click **Apply associated**.

   When the **DocBook WebHelp Classic** transformation is complete, the output is automatically opened in your default browser.

WebHelp Classic with Feedback Output

To publish a DocBook 5 document as a **WebHelp Classic with Feedback** system, follow these steps:

1. Click **Configure Transformation Scenarios**.
2. Select the **DocBook WebHelp Classic with Feedback** scenario from the **DocBook 5** section.
3. Click **Apply associated**.
4. Enter the documentation product ID and the documentation version.

   When the **DocBook WebHelp Classic with Feedback** transformation is complete, your default browser opens the `installation.html` file. This file contains information about the output location, system requirements, installation instructions, and deployment of the output. Follow the instructions to complete the system deployment. For more information, see *Deploying a Feedback-Enabled WebHelp System*.

To watch our video demonstration about the feedback-enabled WebHelp system, go to *https://www.oxygenxml.com/demo/Feedback_Enabled_WebHelp.html*.

WebHelp Classic Mobile Output

To publish a DocBook 5 document as a **WebHelp Classic Mobile** system, follow these steps:

1. Click **Configure Transformation Scenarios**.
2. Select the **DocBook WebHelp Classic Mobile** scenario from the **DocBook 5** section.
3. Click **Apply associated**.

When the **DocBook WebHelp Classic Mobile** transformation is complete, the output is automatically opened in your default browser.

Customizing WebHelp Transformation Scenarios

To customize a DocBook WebHelp transformation scenario, you can edit various parameters, including the following most commonly used ones:

args.default.language

This parameter is used if the language is not detected in the *DITA map*. The default value is `en-us`.

clean.output

Deletes all files from the output folder before the transformation is performed (only no and `yes` values are valid and the default value is no).

l10n.gentext.default.language

This parameter is used to identify the correct stemmer that differs from language to language. For example, for English the value of this parameter is `en` or for French it is `fr`, and so on.

use.stemming

Controls whether or not you want to include stemming search algorithms into the published output (default setting is `false`).

webhelp.copyright

Adds a small copyright text that appears at the end of the **Table of Contents** pane.

webhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

webhelp.favicon

The file path that points to an image to be used as a *favicon* in the WebHelp output.

webhelp.footer.file

Path to an XML file that includes the footer content for your WebHelp output pages. You can use this parameter to integrate social media features (such as widgets for Facebook™, Twitter™, Google Analytics, or Google+™). The file must be well-formed, each widget must be in separate `div` or `span` element, and the code for each `script` element is included in an XML comment (also, the start and end tags for the XML comment must be on a separate line). The following code exert is an example for adding a Facebook™ widget:

```
<div id="facebook">
  <div id="fb-root"/>
  <script>
    <!-- (function(d, s, id) { var js, fjs = d.getElementsByTagName(s)[0];
      if (d.getElementById(id)) return;
      js = d.createElement(s); js.id = id;
      js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
      fjs.parentNode.insertBefore(js, fjs); }
       (document, 'script', 'facebook-jssdk')); -->
    </script>
    <div data-share="true" data-show-faces="true" data-action="like"
       data-layout="standard" class="fb-like"/>
</div>
```

webhelp.footer.include

Specifies whether or not to include footer in each WebHelp page. Possible values: `yes`, no. If set to no, no footer is added to the WebHelp pages. If set to `yes` and the `webhelp.footer.file` parameter has a value, then the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default Oxygen XML Editor footer is inserted in each WebHelp page.

webhelp.logo.image.target.url

Specifies a target URL that is set on the logo image. When you click the logo image, you will be redirected to this address.

webhelp.logo.image

Specifies a path to an image displayed as a logo in the left side of the output header.

webhelp.product.id (available only for Feedback-enabled systems)

This parameter specifies a short name for the documentation target, or product (for example, `mobile-phone-user-guide`, `hvac-installation-guide`).

Note: You can deploy documentation for multiple products on the same server.

Restriction: The following characters are not allowed in the value of this parameter: `< > / \ ' ( ) { } = ; * % + &`.

webhelp.product.version (available only for Feedback-enabled systems)

Specifies the documentation version number (for example, 1.0, 2.5, etc.). New user comments are bound to this version.

Note: Multiple documentation versions can be deployed on the same server.

Restriction: The following characters are not allowed in the value of this parameter: `< > / \ ' ( ) { } = ; * % + &`.

webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Editor uses for indexing Japanese content in the WebHelp pages.

webhelp.search.ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the search results that are displayed on the **Search** tab (default setting is `true`).

webhelp.skin.css

Path to a CSS file that sets the style theme in the output WebHelp pages. It can be one of the predefined skin CSS from the `OXYGEN_INSTALL_DIR\frameworks\docbook\xsl\com.oxygenxml.webhelp\predefined-skins` directory, or it can be a custom skin CSS generated with the *Oxygen Skin Builder* web application.

For more information about all the DocBook transformation parameters, go to *http://docbook.sourceforge.net/release/xsl/current/doc/fo/index.html*.

Related Information:

WebHelp Classic Output on page 815
WebHelp Classic With Feedback Output on page 818
WebHelp Classic Mobile System (Deprecated) on page 1023
Customizing WebHelp Classic Systems on page 999
Customizing WebHelp Classic Mobile Systems on page 1024

DocBook to PDF Output Customization

When the default layout and output of the DocBook to PDF transformation needs to be customized, follow these steps:

1. Create a custom version of the DocBook title spec file.

   You should start from a copy of the file *[OXYGEN_INSTALL_DIR]*`/frameworks/docbook/xsl/fo/titlepage.templates.xml` and customize it. The instructions for the spec file can be found *here.*

   An example of spec file:

```
<t:titlepage-content t:side="recto">
    <mediaobject/>
    <title
        t:named-template="book.verso.title"
        font-size="&hsize2;"
        font-weight="bold"
        font-family="{$title.font.family}"/>
    <corpauthor/>
      ...
</t:titlepage-content>
```

2. Generate a new XSLT stylesheet from the title spec file from the previous step.

   Apply *[OXYGEN_INSTALL_DIR]*`/frameworks/docbook/xsl/template/titlepage.xsl` to the title spec file. The result is an XSLT stylesheet (for example, `mytitlepages.xsl`).

3. Import `mytitlepages.xsl` in a *DocBook customization layer*.

   The customization layer is the stylesheet that will be applied to the XML document. The `mytitlepages.xsl` should be imported with an element like this:

```
<xsl:import href="dir-name/mytitlepages.xsl"/>
```

4. Insert a logo image in the XML document.

   The path to the logo image must be inserted in the `book/info/mediaobject` element of the XML document.

5. Apply the customization layer to the XML document.

   A quick way is to duplicate the transformation scenario **DocBook PDF** that is included with Oxygen XML Editor and set the customization layer in *the **XSL URL** property of the scenario*.

Related Information:

*The book **DocBook XSL: The Complete Guide** by Bob Stayton contains more details about customizing the PDF output.*
Video demonstration for creating a DocBook customization layer in Oxygen XML Editor.

Show Comments and Tracked Changes in DocBook 5 PDF Output

To include comments and *tracked changes* (stored within your DocBook 5 documents) in the PDF output, follow these steps:

1. Click the **Configure Transformation Scenario(s)** button.
2. Select **DocBook PDF (Show Change Tracking and Comments)** in the **DocBook 5** section.
3. If you need to configure the transformation, click the ***Edit*** or ***Duplicate*** button, make your changes to the scenario, and click **OK**.
4. Click the **Apply Associated** button to run the transformation scenario.

   Result: Comment threads and tracked changes will now appear in the PDF output. Details about each comment or change will be available in the footer section for each page.

DocBook to EPUB Transformation

The EPUB specification recommends the use of *OpenType* fonts (recognized by their `.otf` file extension) when possible. To use a specific font, follow these steps:

1. Declare it in your CSS file, as in the following example:

```
@font-face {
font-family: "MyFont";
font-weight: bold;
font-style: normal;
src: url(fonts/MyFont.otf);
}
```

2. In the CSS, specify where this font is used. To set it as default for `h1` elements, use the `font-family` rule, as in the following example:

```
h1 {
font-size:20pt;
margin-bottom:20px;
font-weight: bold;
font-family: "MyFont";
text-align: center;
}
```

3. Open the ***Configure Transformation Scenario(s)*** *dialog box*, select the **DocBook EPUB** transformation scenario, and click **Edit**.
4. In the **Parameters** tab, set the `epub.embedded.fonts` parameter to `fonts/MyFont.otf`. If you need to provide more files, use commas to separate their file paths.

   Note: The `html.stylesheet` parameter allows you to include a custom CSS in the output EPUB.
5. Run the transformation scenario.

DocBook to DITA Transformation

Oxygen XML Editor includes a built-in transformation scenario that is designed to convert DocBook content to DITA. This transformation scenario is based upon a DITA Open Toolkit plugin that is available at *sourceforge.net*.

To convert a DocBook document to DITA, follow these steps:

1. Use one of the following two methods to begin the transformation process:
   - To apply the transformation scenario to a newly opened file, use the **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on OS X))** action from the toolbar or the **Document** > **Transformation** menu.
   - To customize the transformation or change the scenario that is associated with the document, use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu.
2. Select the **DocBook to DITA** transformation scenario in the **DocBook 4** or **DocBook 5** section.
3. Click the **Apply associated** button to run the transformation.

   Step Result: The transformation will convert as many of the DocBook elements into equivalent DITA elements as it can recognize in its mapping process. For elements that cannot be mapped, the transformation will insert XML comments so that you can see which elements could not be converted.
4. Adjust the resulting DITA composite to suit your needs. You may have to remove comments, fix validation errors, adjust certain attributes, or split the content into individual topics.

Related Information:

Editing a Transformation Scenario on page 925

Configure Transformation Scenario(s) Dialog Box on page 926

Inserting an `olink` in DocBook Documents

The `olink` element is used for linking to resources outside the current DocBook document. The `targetdoc` attribute is used for the document ID that contains the target element and the `targetptr` attribute for the ID of the target element (the value of an `id` or `xml:id` attribute). The combination of those two attributes provides a unique identifier to locate cross references.

For example, a *Mail Administrator Guide* with the document ID `MailAdminGuide` might contain a chapter about user accounts, like this:

```
<chapter id="user_accounts">
<title>Administering User Accounts</title>
<para>blah blah</para>
```

You can form a cross reference to that chapter by adding an `olink`, as in the following example:

```
You may need to update your
<olink targetdoc="MailAdminGuide" targetptr="user_accounts">user accounts
</olink>
when you get a new machine.
```

To use an `olink` to create links between documents, follow these steps:

1. Decide which documents are to be included in the domain for cross referencing.

   A unique ID must be assigned to each document that will be referenced with an `olink`. It is usually added as an `id` (or `xml:id` for DocBook5) attribute to the root element of the document.
2. Decide on your output hierarchy.

   For creating links between documents, the relative locations of the output documents must be known. Before going further you must decide the names and locations of the output directories for all the documents from the domain. Each directory will be represented by an element: `<dir name="directory_name">`, in the target database document.
3. Create the target database document.

   Each collection of documents has a master target database document that is used to resolve all `olinks` from that collection. The target database document is an XML file that is created once. It provides a means for pulling in the target data for each document. The database document is static and all the document data is pulled in dynamically.

   Example: The following is an example of a target database document. It structures a collection of documents in a `sitemap` element that provides the relative locations of the outputs (HTML in this example). Then it pulls in the individual target data using system entity references to target data files that will be created in the next step.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE targetset [
<!ENTITY ugtargets SYSTEM "file:///doc/userguide/target.db">
<!ENTITY agtargets SYSTEM "file:///doc/adminguide/target.db">
<!ENTITY reftargets SYSTEM "file:///doc/man/target.db">
]>
<targetset>
  <targetsetinfo>
    Description of this target database document,
    which is for the examples in olink doc.
  </targetsetinfo>

  <!-- Site map for generating relative paths between documents -->
  <sitemap>
    <dir name="documentation">
      <dir name="guides">
        <dir name="mailuser">
          <document targetdoc="MailUserGuide"
                    baseuri="userguide.html">
            &ugtargets;
          </document>
        </dir>
        <dir name="mailadmin">
          <document targetdoc="MailAdminGuide">
            &agtargets;
          </document>
        </dir>
      </dir>
      <dir name="reference">
        <dir name="mailref">
```

```
          <document targetdoc="MailReference">
            &reftargets;
          </document>
        </dir>
      </dir>
    </dir>
  </sitemap>
</targetset>
```

4. Generate the target data files by executing a DocBook transformation scenario.

   Before applying the transformation, you need to edit the transformation scenario, go to the **Parameters** tab, and make sure the value of the `collect.xref.targets` parameter is set to `yes`. The default name of a target data file is `target.db`, but it can be changed by setting an absolute file path in the `targets.filename` parameter.

   Example: An example of a `target.db` file:

```
<div element="book" href="#MailAdminGuide" number="1" targetptr="user_accounts">
 <ttl>Administering User Accounts</ttl>
 <xreftext>How to administer user accounts</xreftext>
 <div element="part" href="#d5e4" number="I">
  <ttl>First Part</ttl>
  <xreftext>Part I, “First Part”</xreftext>
  <div element="chapter" href="#d5e6" number="1">
    <ttl>Chapter Title</ttl>
    <xreftext>Chapter 1, Chapter Title</xreftext>
    <div element="sect1" href="#src_chapter" number="1" targetptr="src_chapter">
      <ttl>Section1 Title</ttl>
      <xreftext>xreflabel_here</xreftext>
    </div>
  </div>
 </div>
</div>
```

5. Insert `olink` elements in the DocBook documents.

   When editing a DocBook XML document in **Author** mode, the **Insert OLink** action is available in the **Link** drop-down menu from the toolbar. This action opens the **Insert OLink** dialog box that allows you to select the target of an `olink` from the list of all possible targets from a specified target database document (specified in the **Targetset URL** field). Once a **Targetset URL** is selected, the structure of the target documents is presented. For each target document (`targetdoc`), its content is displayed, allowing you to easily identify the appropriate `targetptr`. You can also use the search fields to quickly identify a target. If you already know the values for the `targetdoc` and `targetptr` attributes, you can insert them directly in the corresponding fields.

   Example: In the following image, the target database document is called `target.xml`, *dbadmin* is selected for the target document (`targetdoc`), and *bldinit* is selected as the value for the `targetptr` attribute. Notice that you can also add XREF text into the `olink` by using the `xreftext` field.

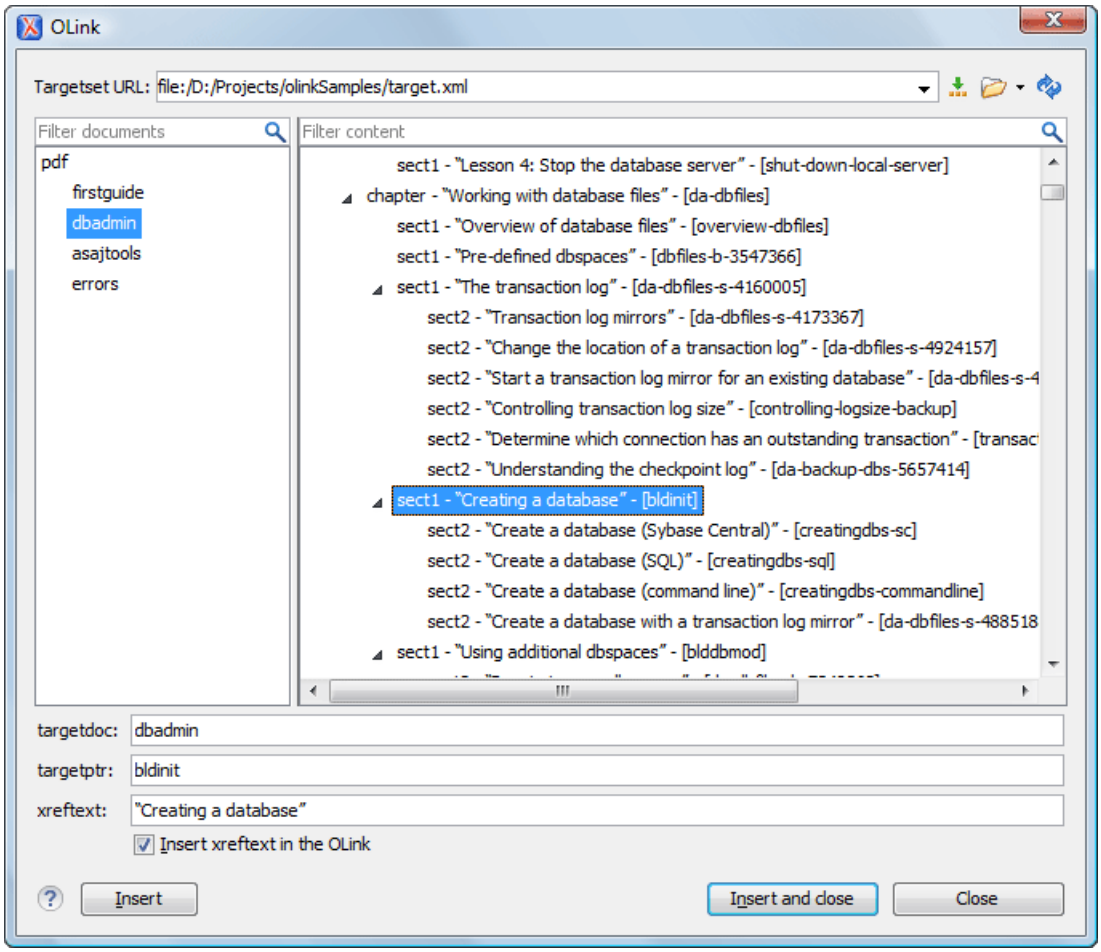


Figure 461: Insert OLink Dialog Box

6. Process a DocBook transformation for each document to generate the output.
   a) Edit the transformation scenario and set the value of the `target.database.document` parameter to be the URL of the target database document.
   b) Apply the transformation scenario.

DocBook 5.1 Assembly

The DocBook *Assembly* document type was introduced with DocBook 5.1 and it is used to define the hierarchy and relationships for a collection of resources. It is especially helpful for topic-oriented authoring scenarios since it assembles a set of resources (such as *DocBook 5.1 topics*) to form a hierarchical structure for a larger publication.

An *Assembly* document usually has four major parts:

- **Resources** - Identifies a collection of resources (such as topics). An *Assembly* may identify one or more collections.
- **Structure** - Identifies an artifact to be assembled. A document in this case is the particular collection of resources (such as topics) that forms the documentation. Within the `structure` element, an `output` element can be used to identify the type of output to be generated and `module` elements can be used to identify the resources to be included. An *Assembly* may identify one or more *structures*.
- **Relationships** - Identifies relationships between resources. These relationships may be manifested in any number of *structures* during assembly. An *Assembly* may identify any number of relationships.
- **Transformations** - Identifies transformations that can be applied during assembly. An *Assembly* may identify any number of transformations.

For detailed information about the DocBook *Assembly* document type, see *DocBook 5.1: The Definitive Guide - DocBook Assemblies*.

File Definition

A file is considered to be an *Assembly* when the root name is `assembly`.

Default Document Templates

A default **Assembly** document template is available when creating *new documents from templates* and it can be found in: **Framework Templates** > **DocBook 5** > **DocBook 5.1**.

The default template for DocBook Assembly documents is located in the *[OXYGEN_INSTALL_DIR]*/`frameworks/docbook/templates/Docbook5.1` folder.

Default Schema for Validation and Content Completion

The default schema that is used if one is not detected is `docbookxi.rng` and it is stored in *[OXYGEN_INSTALL_DIR]*/`frameworks/docbook/5.1/rng/`.

Transformation Scenarios

Oxygen XML Editor includes a built-in transformation scenario that can be applied on an *Assembly* file to generate an *assembled* (merged) DocBook file. The scenarios is called **DocBook Assembly** and is found in the **DocBook 5** section in the ***Configure Transformation Scenario(s)*** *dialog box*.

Resources

- *DocBook 5.1: The Definitive Guide - DocBook Assemblies*
- *Docbook 5.1 Specifications*
- Sample files: *[OXYGEN_INSTALL_DIR]*/`samples/docbook/v5/assembly/`

DocBook 5.1 Topic

The DocBook *Topic* document type was introduced with DocBook 5.1 and it is used as a modular unit of documentation. It is similar to the concept of the DITA *Topic* and can be used as modular resources in conjunction with *DocBook 5.1 Assembly* documents.

For detailed information about the DocBook *Topic* document type, see *DocBook 5.1: The Definitive Guide - Topic*.

File Definition

A DocBook file is considered to be a *Topic* when the root name is `topic`.

Default Document Templates

A default **Topic** document template is available when creating *new documents from templates* and it can be found in: **Framework Templates** > **DocBook 5** > **DocBook 5.1**.

The default template for DocBook Assembly documents is located in the *[OXYGEN_INSTALL_DIR]*/`frameworks/docbook/templates/Docbook5.1` folder.

Default Schema for Validation and Content Completion

The default schema that is used if one is not detected is `docbookxi.rng` and it is stored in *[OXYGEN_INSTALL_DIR]*/`frameworks/docbook/5.1/rng/`.

Transformation Scenarios

Since DocBook *Topics* are modular resources, they are *assembled* and transformed in the ***DocBook Assembly*** *transformation process*. You can also use any of the built-in DocBook transformation scenarios to transform individual DocBook Topics to a variety of outputs, such as PDF, HTML, EPUB, and more. They are found in the **DocBook 5** section in the ***Configure Transformation Scenario(s)*** *dialog box*.

Resources

- *DocBook 5.1: The Definitive Guide - Topic*
- *Docbook 5.1 Specifications*
- Sample files: `[OXYGEN_INSTALL_DIR]/samples/docbook/v5/assembly/`

Related Information:
DocBook 5.1 Assembly on page 792

DocBook Targetset Document Type (Framework)

DocBook *Targetset* documents are used to resolve cross references with the DocBook `olink`.

File Definition

A file is considered to be a *Targetset* when the root name is `targetset`.

Default Document Templates

A default **Map** document template is available when creating *new documents from templates* and it can be found in: **Framework Templates** > **DocBook Targetset**.

The default template for DocBook Targetset documents is located in the `[OXYGEN_INSTALL_DIR]/frameworks/docbook/templates/Targetset` folder.

Default Schema for Validation and Content Completion

The default schema, `targetdatabase.dtd`, for this type of document is stored in `[OXYGEN_INSTALL_DIR]/frameworks/docbook/xsl/common/`.

Resources

- *DocBook Specifications*

DITA Topics Document Type (Framework)

The Darwin Information Typing Architecture (DITA) is an XML-based architecture for authoring, producing, and delivering technical information. It divides content into small, self-contained topics that you can reuse in various deliverables. The extensibility of DITA permits organizations to define specific information structures while still using standard tools to work with them. DITA content is created as topics, each an individual XML file. Typically, each topic has a defined primary objective and structure, and DITA also includes several specialized topic types (*task*, *concept*, *reference*, *glossary entry*).

For much more detailed information, resources, and instructions, see the *DITA Authoring and Publishing* on page 1580 chapter.

File Definition

A file is considered to be a DITA topic document when one of the following conditions are true:

- The root element name is one of the following: `concept`, `task`, `reference`, `dita`, or `topic`.
- The PUBLIC ID of the document is a PUBLIC ID for the elements listed above.
- The root element of the file has an attribute named `DITAArchVersion` for the “http://dita.oasis-open.org/architecture/2005/” namespace. This enhanced case of matching is only applied when the ***Enable DTD/XML Schema processing in document type detection*** *option* is selected from the ***Document Type Association*** *preferences page*.

Default Document Templates

There are a variety of default *DITA topic* templates available when creating *new documents from templates* and they can be found in various folders inside: **Framework Templates** > **DITA**.

The default templates for DITA topic documents are located in the *[OXYGEN_INSTALL_DIR]*/frameworks/dita/templates/topic folder.

Default Schema for Validation and Content Completion

Default schemas that are used if one is not detected in the DITA documents are stored in the various folders inside *DITA-OT-DIR*/dtd/ or *DITA-OT-DIR*/schema/.

Default CSS

The default CSS files used for rendering DITA content in **Author** mode are stored in the various folders inside: *[OXYGEN_INSTALL_DIR]*/frameworks/dita/css/.

By default, these default CSS files are merged with any that are specified in the document. For more information, see *Configuring and Managing Multiple CSS Styles* on page 1256.

Default XML Catalogs

The default *XML Catalogs* for the DITA topic document type are as follows:

- *DITA-OT-DIR*/catalog-dita.xml
- *[OXYGEN_INSTALL_DIR]*/frameworks/dita/catalog.xml
- *[OXYGEN_INSTALL_DIR]*/frameworks/dita/plugin/catalog.xml
- *[OXYGEN_INSTALL_DIR]*/frameworks/dita/styleguide/catalog.xml

Transformation Scenarios

Oxygen XML Editor includes built-in transformation scenarios that allow you to transform individual DITA Topics to a XHTML or PDF output. They can be found in the **DITA** section in the ***Configure Transformation Scenario(s)*** *dialog box*.

Resources

- *DITA Specifications*
- *DITA Style Guide Best Practices for Authors*
- *Oxygen Video Tutorial: DITA Editing*

Related Information:

DITA Authoring and Publishing on page 1580
Your First DITA Topic on page 9
Editing XML Documents in Author Mode on page 329
Editing XML Documents in Text Mode on page 290

DITA Author Mode Actions

A variety of actions are available in the DITA *framework* that can be added to the **DITA** menu, the **Author Custom Actions** toolbar, the contextual menu, and the *Content Completion Assistant*.

DITA Toolbar Actions

The following default actions are readily available on the **DITA (Author Custom Actions)** toolbar when editing in **Author** mode (by default, most of them are also available in the **DITA** menu and in various submenus of the contextual menu):

B Bold

Surrounds the selected text with a b tag. You can use this action on multiple non-contiguous selections.

I Italic

Surrounds the selected text with an i tag. You can use this action on multiple non-contiguous selections.

Underline

Surrounds the selected text with a u tag. You can use this action on multiple non-contiguous selections.

Link Actions Drop-Down Menu

The following link actions are available from this menu:

Cross Reference

Opens the ***Cross Reference (xref)*** *dialog box* that allows you to insert a link to a target resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your *DITA map* structure. Once the target resource has been selected, you can also target specific elements within that resource. For more information, see *Linking in DITA Topics* on page 1674.

File Reference

Opens the ***File Reference*** *dialog box* that allows you to insert a link to a target file resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your *DITA map* structure. For more information, see *Linking in DITA Topics* on page 1674.

Web Link

Opens the ***Web Link*** *dialog box* that allows you to insert a link to a target web-related resource at the current location within a document. The target resource can be a URL or a key that is already defined in your *DITA map* structure. For more information, see *Linking in DITA Topics* on page 1674.

Related Link to Topic

Opens the ***Cross Reference (xref)*** *dialog box* that allows you to insert a link to a target resource in a related links section at the bottom of the current document. The target resource can be the location of a file or a key that is already defined in your *DITA map* structure. Once the target resource has been selected, you can also target specific elements within that resource. If a related links section does not already exist, this action creates one. For more information, see *Linking in DITA Topics* on page 1674.

Related Link to File

Opens the ***File Reference*** *dialog box* that allows you to insert a link to a target file resource in a related links section at the bottom of the current document. The target resource can be the location of a file or a key that is already defined in your *DITA map* structure. If a related links section does not already exist, this action creates one. For more information, see *Linking in DITA Topics* on page 1674.

Related Link to Web Page

Opens the ***Web Link*** *dialog box* that allows you to insert a link to a target web-related resource in a related links section at the bottom of the current document. The target resource can be a URL or a key that is already defined in your *DITA map* structure. If a related links section does not already exist, this action creates one. For more information, see *Linking in DITA Topics* on page 1674.

Insert Image

Opens the ***Insert Image*** *dialog box* that allows you to configure the properties of an image to be inserted into a DITA document at the cursor position.

Insert Media Resource

Opens the ***Insert Media*** *dialog box* that allows you to select and configure the properties of a media object to be inserted into a DITA document at the cursor position. The result will be that a reference to the specified video, audio, or embedded HTML frame is inserted in an `object` element and it is rendered in **Author** mode so that it can be played directly from there.

Insert Section Drop-Down Menu

The following insert actions are available from this menu:

Insert Section

Inserts a new `section` element in the document, depending on the current context.

Insert Concept

Inserts a new `concept` element, depending on the current context. Concepts provide background information that users must know before they can successfully work with a product or interface.

Insert Task

Inserts a new `task` element, depending on the current context. Tasks are the main building blocks for task-oriented user assistance. They generally provide step-by-step instructions that will help a user to perform a task.

Insert Topic

Inserts a new `topic` element, depending on the current context. Topics are the basic units of DITA content and are usually organized around a single subject.

Insert Reference

Inserts a new `reference` element, depending on the current context. A reference is a top-level container for a reference topic.

Insert Paragraph

Inserts a new paragraph at current cursor position.

Reuse Content

This action provides a mechanism for reusing content fragments. It opens the ***Reuse Content*** *dialog box* that allows you to insert several types of references to reusable content at the cursor position. The types of references that you can insert using this dialog box include *content references (`conref`)*, *content key references (`conkeyref`)*, or *key references to metadata (`keyref`)*.

Insert step or list item

Inserts a new list or step item in the current list type.

Insert Unordered List

Inserts an unordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an unordered list.

Insert Ordered List

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

Sort

Sorts cells or list items in a table.

Insert Table

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content — text + markup — that is rendered inside a *block element*) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

Note: If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables**.

Insert Row Below

Inserts a new table row with empty cells below the current row. This action is available when the cursor is positioned inside a table.

Delete Row(s)

Deletes the table row located at cursor position or multiple rows in a selection.

Insert Column After

Inserts a new table column with empty cells after the current column. This action is available when the cursor is positioned inside a table.

Delete Column(s)

Deletes the table column located at cursor position or multiple columns in a selection.

Table Properties

Opens the **Table properties** dialog box that allows you to configure properties of a table (such as frame borders).

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

DITA Menu Actions

In addition to the *DITA toolbar actions*, the following default actions are available in the **DITA** menu when editing in **Author** mode:

Reuse Content

This action provides a mechanism for reusing content fragments. It opens the ***Reuse Content*** *dialog box* that allows you to insert several types of references to reusable content at the cursor position. The types of references that you can insert using this dialog box include *content references (`conref`)*, *content key references (`conkeyref`)*, or *key references to metadata (`keyref`)*.

Push Current Element

Opens the ***Push current element*** *dialog box* that allows content from a source topic to be inserted into another topic without any special coding in the topic where the content will be re-used.

Edit Content Reference

This action is available for elements with a `conref` or `conkeyref` attribute. it opens the **Edit Content Reference** dialog box that allows you to edit the source location (or key) and source element of a content reference (or content key reference), and the reference details (`conref`/`conkeyref` and `conrefend` attributes). For more information, see *Reuse Content Dialog Box* on page 1659.

Replace Reference with content

Replaces the referenced fragment (`conref` or `conkeyref`) at the cursor position with its content. This action is useful if you want to make changes to the content in the currently edited document without changing the referenced fragment in its source location.

Remove Content Reference

Removes the content reference (`conref` or `conkeyref`) inside the element at the cursor position.

Create Reusable Component

Creates a reusable component from the selected fragment of text. For more information, see *Creating a Reusable Content Component* on page 1667.

Insert Reusable Component

Inserts a reusable component at cursor location. For more information, see *Inserting a Reusable Content Component* on page 1668.

Paste special submenu

This submenu includes the following special paste actions that are specific to the DITA *framework*:

Paste as content reference

Inserts a content reference (a DITA element with a `conref` attribute) to the DITA XML element from the clipboard. An entire DITA XML element with an ID attribute must be present in the clipboard when the action is invoked. The `conref` attribute will point to this ID value.

Paste as content key reference

Allows you to indirectly reference content using the `conkeyref` attribute. When the DITA content is processed, the key references are resolved using key definitions from *DITA maps*. To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.

2. In the **DITA Maps Manager** view, make sure that the ***Root map*** *combo box* points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the **DITA Maps Manager**, select **Edit Properties**, and enter a value in the **Keys** field.

Paste as link

Inserts a `link` element or an `xref` (depending on the location of the paste operation) that points to the DITA XML element from the clipboard. An entire DITA XML element with an ID attribute must be present in the clipboard when the action is invoked. The `href` attribute of `link/href` will point to this ID value.

Paste as link (keyref)

Inserts a link to the element that you want to reference. To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.
2. In the **DITA Maps Manager** view, make sure that the ***Root map*** *combo box* points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the **DITA Maps Manager**, select **Edit Properties**, and enter a value in the **Keys** field.

Table submenu

In addition to the table actions available on the toolbar, the following actions are available in this submenu:

Insert Row Above

Inserts a new table row with empty cells above the current row. This action is available when the cursor is positioned inside a table.

Insert Rows

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (**Above** or **Below** the current row).

Insert Column Before

Inserts a column before the current one.

Insert Columns

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (**Above** or **Below** the current column).

Insert Cell

Inserts a new empty cell depending on the current context. If the cursor is positioned between two cells, Oxygen XML Editor a new cell at cursor position. If the cursor is inside a cell, the new cell is created after the current cell.

Insert > Σ Insert Equation

Opens the **XML Fragment Editor** that allows you to insert and *edit MathML notations*.

ID Options

Opens the **ID Options** dialog box that allows you to configure options for generating IDs in **Author** mode. The dialog box includes the following:

ID Pattern

The pattern for the ID values that will be generated. This text field can be customized using constant strings or any of the Oxygen XML Editor *Editor Variables*.

Element name or class value to generate ID for

The elements for which ID values will be generated, specified using class attribute values. To customize the list, use the **Add**, **Edit**, or **Remove** buttons.

Auto generate IDs for elements

If selected, Oxygen XML Editor will automatically generate unique IDs for the elements listed in this dialog box when they are created in **Author** mode.

Remove IDs when copying content in the same document

When copying and pasting content in the same document, this option allows you to control whether or not pasted elements that are listed in this dialog box should retain their existing IDs. To retain the element IDs, deselect this option.

Note: This option does not have an effect on content that is *cut* and *pasted*.

Generate IDs

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements from the list in the **ID Options** dialog box that are found in the current selection.

Note: The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `id` attribute.

Browse DITA Style Guide

Opens the **DITA Style Guide Best Practices for Authors** in your browser and displays a topic that is relevant to the element at the cursor position. When editing DITA documents, this action is available in the contextual menu of the editing area (under the **About Element** sub-menu), in the **DITA** menu, and in some of the documentation tips that are displayed by the *Content Completion Assistant*.

Refresh References

You can use this action to manually trigger a refresh and update of all referenced resources.

Tags display mode Submenu

Full Tags with Attributes

Displays full tag names with attributes for both *block* and *inline elements*.

Full Tags

Displays full tag names without attributes for both *block* and *inline elements*.

Block Tags

Displays full tag names for *block elements* and simple tags without names for *inline elements*.

Inline Tags

Displays full tag names for *inline elements*, while *block elements* are not displayed.

Partial Tags

Displays simple tags without names for *inline elements*, while *block elements* are not displayed.

No Tags

No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Configure Tags Display Mode

Use this option to go to the ***Author*** *preferences page* where you can configure the **Tags Display Mode** options.

Profiling/Conditional Text Submenu

Edit Profiling Attributes

Allows you to configure the *profiling attributes* and their values.

Show Profiling Colors and Styles

Select this option to turn on conditional styling.

Show Profiling Attributes

Select this option to turn on conditional text markers. They are displayed at the end of conditional text blocks, as a list of attribute name and their currently set values.

Show Excluded Content

When this option is selected, the content filtered by the currently applied condition set is grayed-out. To show only the content that matches the currently applied condition set, deselect this option.

List of all profiling condition sets that match the current document type

You can click a listed *condition set* to activate it.

Profiling Settings

Opens the *profiling options preferences page*, where you can manage profiling attributes and profiling conditions sets. You can also configure the profiling styles and colors options from the *colors/styles preferences page* and the *attributes rendering preferences page*.

DITA Contextual Menu Actions

In addition to many of the *DITA toolbar actions* and the *general **Author** mode contextual menu actions*, the following DITA *framework*-specific actions are also available in the contextual menu when editing in **Author** mode:

Paste special submenu

This submenu includes the following special paste actions that are specific to the DITA *framework*:

Paste as content reference

Inserts a content reference (a DITA element with a `conref` attribute) to the DITA XML element from the clipboard. An entire DITA XML element with an ID attribute must be present in the clipboard when the action is invoked. The `conref` attribute will point to this ID value.

Paste as content key reference

Allows you to indirectly reference content using the `conkeyref` attribute. When the DITA content is processed, the key references are resolved using key definitions from *DITA maps*. To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.
2. In the **DITA Maps Manager** view, make sure that the ***Root map** combo box* points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the **DITA Maps Manager**, select **Edit Properties**, and enter a value in the **Keys** field.

Paste as link

Inserts a `link` element or an `xref` (depending on the location of the paste operation) that points to the DITA XML element from the clipboard. An entire DITA XML element with an ID attribute must be present in the clipboard when the action is invoked. The `href` attribute of `link/href` will point to this ID value.

Paste as link (keyref)

Inserts a link to the element that you want to reference. To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.
2. In the **DITA Maps Manager** view, make sure that the ***Root map** combo box* points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the **DITA Maps Manager**, select **Edit Properties**, and enter a value in the **Keys** field.

Image Map Editor

This action is available in the contextual menu when it is invoked on an image. This action applies an *image map* to the current image (if one does not already exist) and opens the **Image Map Editor** dialog box. This feature allows you to create hyperlinks in specific areas of an image that will link to various destinations.

Table Actions

The following table editing actions are available in the contextual menu when it is invoked on a tabl

e:

Insert Rows

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (**Above** or **Below** the current row).

Delete Row(s)

Deletes the table row located at cursor position or multiple rows in a selection.

Insert Columns

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (**Above** or **Below** the current column).

Delete Column(s)

Deletes the table column located at cursor position or multiple columns in a selection.

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

Sort

Sorts cells or list items in a table.

Table Properties

Opens the **Table properties** dialog box that allows you to configure properties of a table (such as frame borders).

Other Actions submenu

This submenu give you access to all the usual contextual menu actions.

Insert > Σ Insert Equation

Opens the **XML Fragment Editor** that allows you to insert and *edit MathML notations*.

Generate IDs

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements from the list in the **ID Options** dialog box that are found in the current selection.

Note: The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `id` attribute.

Reuse submenu

This submenu includes the following actions in regards to reusing content in DITA:

Push Current Element

Opens the ***Push current element*** *dialog box* that allows content from a source topic to be inserted into another topic without any special coding in the topic where the content will be re-used.

Edit Content Reference

This action is available for elements with a `conref` or `conkeyref` attribute. it opens the **Edit Content Reference** dialog box that allows you to edit the source location (or key) and source element of a content reference (or content key reference), and the reference details (`conref`/`conkeyref` and `conrefend` attributes). For more information, see *Reuse Content Dialog Box* on page 1659.

Replace Reference with content

Replaces the referenced fragment (`conref` or `conkeyref`) at the cursor position with its content. This action is useful if you want to make changes to the content in the currently edited document without changing the referenced fragment in its source location.

Remove Content Reference

Removes the content reference (`conref` or `conkeyref`) inside the element at the cursor position.

Create Reusable Component

Creates a reusable component from the selected fragment of text. For more information, see *Creating a Reusable Content Component* on page 1667.

Insert Reusable Component

Inserts a reusable component at cursor location. For more information, see *Inserting a Reusable Content Component* on page 1668.

Search References (Ctrl + Shift + G)

Finds the references to the `id` attribute value for the element at the current cursor position, in all the topics contained in the current *DITA map* (opened in the ***DITA Maps Manager** view*). If no references are found for the current element, a dialog box will be displayed that offers you the option of searching for references to its ancestor elements.

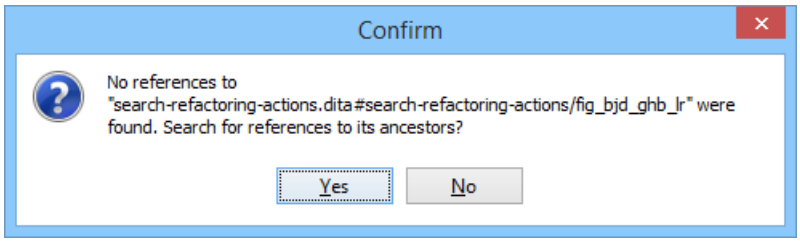


Figure 462: Search References to Ancestors Dialog Box

Show Key Definition

Available for elements that have a `conkeyref` or `keyref` attribute set (or elements with an ancestor element that has a `conkeyref` or `keyref` attribute). It computes the key name and opens the *DITA map* that contains the definition of the key with the element that defines that key selected.

About Element submenu

This submenu includes the following actions:

Style Guide

Opens the **DITA Style Guide Best Practices for Authors** in your browser and displays a topic that is relevant to the element at the cursor position. When editing DITA documents, this action is available in the contextual menu of the editing area (under the **About Element** sub-menu), in the **DITA** menu, and in some of the documentation tips that are displayed by the *Content Completion Assistant*.

Browse reference manual

Opens a reference to the documentation of the XML element closest to the cursor position in a web browser.

Show Definition

Moves the cursor to the definition of the current element.

DITA Drag/Drop (or Copy/Paste) Actions

Dragging a file from the ***Project** view* or ***DITA Maps Manager** view* and dropping it into a DITA document that is edited in **Author** mode, creates a link to the dragged file (the `xref` DITA element with the `href` attribute) at the drop location. Copy and paste actions work the same.

You can also drag images or media files from your system explorer or the ***Project** view* and drop them into a DITA document (or copy and paste). This will insert the appropriate element at the drop or paste location (for example, dropping/pasting an image will insert the `image` DITA element with the `href` attribute).

DITA Map Document Type (Framework)

DITA maps are documents that collect and organize references to DITA topics to indicate the relationships between the topics. They can be used as a container for topics used to transform a collection of content into a publication and they offer a sequence and structure to the topics. They can also serve as outlines or tables of contents for DITA deliverables and as build manifests for DITA projects. *DITA maps* allow scalable reuse of content across multiple contexts. Maps can reference topics or other maps, and can contain a variety of content types and metadata.

For much more detailed information, resources, and instructions, see the *DITA Authoring and Publishing* on page 1580 chapter.

File Definition

A file is considered to be a *DITA map* document when one of the following conditions are true:

- The root element name is one of the following: `map`, `bookmap`.
- The public ID of the document is `-//OASIS//DTD DITA Map` or `-//OASIS//DTD DITA BookMap`.
- The root element of the file has an attribute named `class` that contains the value `map/map` and a `DITAArchVersion` attribute from the *http://dita.oasis-open.org/architecture/2005/* namespace. This enhanced case of matching is only applied when the ***Enable DTD/XML Schema processing in document type detection*** *option* from the ***Document Type Association*** *preferences page* is selected.

Default Document Templates

There are a variety of default *DITA map* templates available when creating *new documents from templates* and they can be found in various folders inside: **Framework Templates** > **DITA Map**.

The default templates for *DITA map* documents are located in the *`[OXYGEN_INSTALL_DIR]`*`/frameworks/sita/templates/map` folder.

Default Schema for Validation and Content Completion

Default schemas that are used if one is not detected in the *DITA map* document are stored in the various folders inside *`DITA-OT-DIR`*`/dtd/` or *`DITA-OT-DIR`*`/schema/`.

Default CSS

The default CSS files used for rendering DITA content in **Author** mode are stored in the various folders inside: *`[OXYGEN_INSTALL_DIR]`*`/frameworks/dita/css/`.

By default, these default CSS files are merged with any that are specified in the document. For more information, see *Configuring and Managing Multiple CSS Styles* on page 1256.

Default XML Catalogs

The default *XML Catalogs* for the *DITA map* document type are as follows:

- *`[OXYGEN_INSTALL_DIR]`*`/frameworks/dita/catalog.xml`
- *`DITA-OT-DIR`*`/catalog-dita.xml`

Transformation Scenarios

Oxygen XML Editor includes numerous built-in transformation scenarios that allow you to transform *DITA maps* to a variety of outputs, such as WebHelp, PDF, ODF, XHTML, EPUB, and CHM. For more information, see the *DITA Map Transformation Scenarios* on page 808 section.

Resources

- *DITA Specifications*
- *DITA Style Guide Best Practices for Authors*
- *Oxygen Video Tutorial: DITA Maps Manager*

Related Information:

Selecting a Root Map on page 1591
DITA Authoring and Publishing on page 1580
Your First DITA Topic on page 9
Editing XML Documents in Author Mode on page 329
Editing XML Documents in Text Mode on page 290

DITA Map Author Mode Actions

A variety of actions are available in the *DITA map framework* that can be added to the **DITA** menu, the **Author Custom Actions** toolbar, the contextual menu, and the *Content Completion Assistant*.

DITA Map Toolbar and Menu Actions

When a *DITA map* is opened in **Author** mode, the following default actions are available on the **DITA Map Author Custom Actions** toolbar (by default, they are also available in the **DITA** menu and in various submenus of the contextual menu):

Insert New Topic

Opens a ***New*** *DITA topic dialog box* that allows you to create a new topic and inserts a reference to it at the cursor position.

Insert Topic Reference

Opens the ***Insert Reference*** *dialog box* that allows you to insert and configure a reference to a topic at the cursor position.

Reuse Content

Opens the ***Reuse Content*** *dialog box* that allows you to insert and configure a content reference (`conref`), or a content key reference (`conkeyref`) at the cursor position.

Insert Topic Heading

Opens the ***Insert Reference*** *dialog box* that allows you to insert a topic heading at the cursor position.

Insert Topic Group

Opens the ***Insert Reference*** *dialog box* that allows you to insert a topic group at the cursor position.

Insert Relationship Table

Opens a dialog box that allows you to configure the relationship table to be inserted. The dialog box allows you to configure the number of rows and columns of the relationship table, if the header will be generated and if the title will be added.

Relationship Table Properties

Allows you to change the properties of rows in relationship tables.

Insert Relationship Row

Inserts a new table row with empty cells. The action is available when the cursor position is inside a table.

Insert Relationship Column

Inserts a new table column with empty cells after the current column. The action is available when the cursor position is inside a table.

Delete Relationship Column

Deletes the table column where the cursor is located.

Delete Relationship Row

Deletes the table row where the cursor is located.

Move Up

Moves the selected node up one position on its same level.

Move Down

Moves the selected node down one position on its same level.

Promote(Alt + LeftArrow)

Moves the selected node up one level to the level of its parent node.

Demote(Alt + RightArrow)

Moves the selected node down one level to the level of its child nodes.

DITA Menu Actions

In addition, the following default actions are available in the **DITA** menu when editing a *DITA map* in **Author** mode:

Refresh References

You can use this action to manually trigger a refresh and update of all referenced resources.

Tags display mode Submenu

Full Tags with Attributes

Displays full tag names with attributes for both *block* and *inline elements*.

Full Tags

Displays full tag names without attributes for both *block* and *inline elements*.

Block Tags

Displays full tag names for *block elements* and simple tags without names for *inline elements*.

Inline Tags

Displays full tag names for *inline elements*, while *block elements* are not displayed.

Partial Tags

Displays simple tags without names for *inline elements*, while *block elements* are not displayed.

No Tags

No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Configure Tags Display Mode

Use this option to go to the ***Author*** *preferences page* where you can configure the **Tags Display Mode** options.

Profiling/Conditional Text Submenu

Edit Profiling Attributes

Allows you to configure the *profiling attributes* and their values.

Show Profiling Colors and Styles

Select this option to turn on conditional styling.

Show Profiling Attributes

Select this option to turn on conditional text markers. They are displayed at the end of conditional text blocks, as a list of attribute name and their currently set values.

Show Excluded Content

When this option is selected, the content filtered by the currently applied condition set is grayed-out. To show only the content that matches the currently applied condition set, deselect this option.

List of all profiling condition sets that match the current document type

You can click a listed *condition set* to activate it.

Profiling Settings

Opens the *profiling options preferences page*, where you can manage profiling attributes and profiling conditions sets. You can also configure the profiling styles and colors options from the *colors/styles preferences page* and the *attributes rendering preferences page*.

ID Options

Opens the **ID Options** dialog box that allows you to configure options for generating IDs in **Author** mode. The dialog box includes the following:

ID Pattern

The pattern for the ID values that will be generated. This text field can be customized using constant strings or any of the Oxygen XML Editor *Editor Variables*.

Element name or class value to generate ID for

The elements for which ID values will be generated, specified using class attribute values. To customize the list, use the **Add**, **Edit**, or **Remove** buttons.

Auto generate IDs for elements

If selected, Oxygen XML Editor will automatically generate unique IDs for the elements listed in this dialog box when they are created in **Author** mode.

Remove IDs when copying content in the same document

When copying and pasting content in the same document, this option allows you to control whether or not pasted elements that are listed in this dialog box should retain their existing IDs. To retain the element IDs, deselect this option.

Note: This option does not have an effect on content that is *cut* and *pasted*.

Generate IDs

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements from the list in the **ID Options** dialog box that are found in the current selection.

Note: The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `id` attribute.

DITA Map Contextual Menu Actions

In addition to many of the ***DITA Map*** *toolbar actions* and the *general* ***Author*** *mode contextual menu actions*, the following *DITA map framework*-specific actions are also available in the contextual menu when editing in **Author** mode:

Edit Properties

Opens the **Edit Properties** dialog box that allows you to configure the properties of a selected node. For more details about this dialog box, see *Edit Properties Dialog Box* on page 1603.

Generate IDs

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements from the list in the **ID Options** dialog box that are found in the current selection.

Note: The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `id` attribute.

Search References

Finds the references to the `href` or `keys` attribute value of the topic/map reference element at the current cursor position, in all the topics from the current *DITA map* (opened in the ***DITA Maps Manager*** *view*). The current topic/map reference element must have an `href` or `keys` attribute defined to complete the search.

Show Key Definition

Available for elements that have a `conkeyref` or `keyref` attribute set (or elements with an ancestor element that has a `conkeyref` or `keyref` attribute). It computes the key name and opens the *DITA map* that contains the definition of the key with the element that defines that key selected.

DITA Map Drag/Drop Actions

Dragging a file from the ***Project*** *view* or ***DITA Maps Manager*** *view* and dropping it into a *DITA map* document that is edited in **Author** mode creates a link to the dragged file (a `topicref` element, `chapter`, `part`, etc.) at the drop location.

Open Topic from DITA Map in Author Mode

When a *DITA map* is opened in **Author** mode, you can click the icon to the left of a topic to open that particular topic in the editor.

DITA Map Transformation Scenarios

The following types of transformations scenarios are available for *DITA maps*:

- Predefined transformation scenarios allow you to transform a *DITA map* to a variety of outputs, such as WebHelp, PDF, ODF, XHTML, EPUB, CHM, Kindle, and MS Word.
- **Run DITA-OT Integrator** - Use this transformation scenario if you want to *integrate a DITA-OT plugin*. This scenario runs an Ant task that integrates all the plugins from the DITA-OT/plugins directory.
- **DITA Map Metrics Report** - Use this type of transformation scenario if you want to generate a *DITA map* statistics report. They contain information such as:
  - The number of processed maps and topics.
  - Content reuse percentage.
  - Number of elements, attributes, words, and characters used in the entire *DITA map* structure.
  - DITA conditional processing attributes used in the *DITA maps*.
  - Words count.
  - Information types such as number of containing *maps*, *bookmaps*, or topics.

Related Information:

Editing a Transformation Scenario on page 925
Configure Transformation Scenario(s) Dialog Box on page 926

DITA Map to WebHelp Output

DITA maps can be transformed into a variety of WebHelp systems designed to suit your specific needs. This section contains the procedures for obtaining the output for the variants of the WebHelp system.

Related Information:

WebHelp System Output on page 939

WebHelp Responsive Output

To publish a *DITA map* as a **WebHelp Responsive** system, follow these steps:

1. Select the **Configure Transformation Scenario(s)** action from the ***DITA Maps Manager*** toolbar.
2. Select the **DITA Map WebHelp Responsive** scenario from the **DITA Map** section.
3. If you want to configure the transformation, click the **Edit** button.

   Step Result: This opens an **Edit scenario** configuration dialog box that allows you to configure various options in the following tabs:

- *Templates Tab* - This tab contains a set of predefined skins that you can use for the layout of your WebHelp system output.
- *Parameters Tab* - This tab includes numerous parameters that can be set to customize your WebHelp system output. See the *Parameters section* below for details about the most commonly used parameters for WebHelp Responsive transformations.
- *Filters Tab* - This tab allows you to filter certain content elements from the generated output.
- *Advanced Tab* - This tab allows you to specify some advanced options for the transformation scenario.
- *Output Tab* - This tab allows you to configure options that are related to the location where the output is generated.

4. Click **Apply associated** to process the transformation.

   When the **DITA Map WebHelp Responsive** transformation is complete, the output is automatically opened in your default browser.

Parameters for Customizing WebHelp Responsive Output

To customize a transformation scenario, you can edit various parameters, including the following most commonly used ones:

args.default.language

This parameter is used if the language is not detected in the *DITA map*. The default value is `en-us`.

clean.output

Deletes all files from the output folder before the transformation is performed (only no and `yes` values are valid and the default value is no).

editlink.web.author.url

Use this parameter in conjunction with `editlink.web.author.url` to add an *Edit* link next to the topic title in the WebHelp output. When a user clicks the link, the topic is opened in Oxygen XML Web Author where they can make changes that can be saved to a file server. The value should be set as the custom URL of the *main DITA map*. For example, a GitHub custom URL might look like this: `https://getFileContent/oxyengxml/userguide/master/UserGuide.ditamap`. For more details about custom URLs, see *this section*.

editlink.web.author.url

This parameter needs to be used in conjunction with `editlink.remote.ditamap.url` to add an *Edit* link next to the topic title in the WebHelp output. When a user clicks the link, the topic is opened in Oxygen XML Web Author where they can make changes that can be saved to a file server. The value should be set as the URL of the Web Author installation. For example: `https://www.oxygenxml.com/webapp-demo-aws/app/oxygen.html`.

fix.external.refs.com.oxygenxml (Only supported when the DITA OT transformation process is started from Oxygen XML Editor)

The DITA Open Toolkit usually has problems processing references that point to locations outside of the directory of the processed *DITA map*. This parameter is used to specify whether or not the application should try to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references. The fix has no impact on your edited DITA content. Allowed values: `true` or `false` (default).

use.stemming

Controls whether or not you want to include stemming search algorithms into the published output (default setting is `false`).

webhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

webhelp.favicon

The file path that points to an image to be used as a *favicon* in the WebHelp output.

webhelp.reload.stylesheet

Set this parameter to `true` if you have out of memory problems when generating WebHelp. It will increase processing time but decrease the memory footprint. The default value is `false`.

webhelp.search.custom.excludes.file

The path of the file that contains name patterns for HTML files that should not be indexed by the WebHelp search engine. Each exclude pattern must be on a new line. The patterns are considered to be relative to the output directory, and they accept wildcards such as `'*'` (matches zero or more characters) or `'?'` (matches one character). For more information about the patterns, see *https://ant.apache.org/manual/dirtasks.html#patterns*.

webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Editor uses for indexing Japanese content in the WebHelp pages.

webhelp.search.ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the search results that are displayed on the **Search** tab (default setting is `true`).

webhelp.show.changes.and.comments

When set to `yes`, user comments, replies to comments, and tracked changes are published in the WebHelp output. The default value is no.

webhelp.sitemap.base.url

Base URL for all the `loc` elements in the generated `sitemap.xml` file. The value of a `loc` element is computed as the relative file path from the `href` attribute of a `topicref` element from the *DITA map*, appended to this base URL value. The `loc` element is mandatory in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `sitemap.xml` file is not generated.

webhelp.sitemap.change.frequency

The value of the `changefreq` element in the generated `sitemap.xml` file. The `changefreq` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `changefreq` element is not added in `sitemap.xml`. Allowed values: <empty string> (default), `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, `never`.

webhelp.sitemap.priority

The value of the `priority` element in the generated `sitemap.xml` file. It can be set to any fractional number between 0.0 (least important priority) and 1.0 (most important priority). For example, 0.3, 0.5, or 0.8. The `priority` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `priority` element is not added in `sitemap.xml`.

Parameters Specific to WebHelp Responsive Output

webhelp.enable.search.autocomplete

Specifies if the *Autocomplete* feature is enabled in the WebHelp search text field. The default value is `yes`.

webhelp.fragment.after.body

In the generated output it displays a given XHTML fragment after the page body. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.logo_and_title

In the generated output it displays a given XHTML fragment after the logo and title. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.main.page.search

In the generated output it displays a given XHTML fragment after the search field. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.toc_or_tiles

In the generated output it displays a given XHTML fragment after the table of contents or tiles in the main page. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.top_menu

In the generated output it displays a given XHTML fragment after the top menu. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.body

In the generated output it displays a given XHTML fragment before the page body. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.logo_and_title

In the generated output it displays a given XHTML fragment before the logo and title. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.main.page.search

In the generated output it displays a given XHTML fragment before the search field. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.toc_or_tiles

In the generated output it displays a given XHTML fragment before the table of contents or tiles in the main page. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.top_menu

In the generated output it displays a given XHTML fragment before the top menu. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.footer

In the generated output it displays a given XHTML fragment as the page footer. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.head

In the generated output it displays a given XHTML fragment as a page header. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.welcome

In the generated output it displays a given XHTML fragment as a welcome message (or title). The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.merge.nested.topics.related.links

Specifies if the related links from nested topics will be merged with the links in the parent topic. Thus the links will be moved from the topic content to the related links component and all of the links from the same group (for example, *Related Tasks*, *Related References*, *Related Information*) are merged into a single group. The default value is yes.

webhelp.show.breadcrumb

Specifies if the breadcrumb component will be presented in the output. The default value is yes.

webhelp.show.child.links

Specifies if child links will be generated in the output for all topics that have subtopics. The default value is no.

webhelp.show.indexterms.link

Specifies if an icon that links to the index will be presented in the output. The default value is yes.

webhelp.show.main.page.tiles

Specifies if the tiles component will be presented in the main page of the output. For a *tree* style layout, this parameter should be set to no.

webhelp.show.main.page.toc

Specifies if the table of contents will be presented in the main page of the output. The default value is yes.

webhelp.show.navigation.links

Specifies if navigation links will be presented in the output. The default value is yes.

webhelp.show.print.link

Specifies if a print link or icon will be presented within each topic in the output. The default value is yes.

webhelp.show.related.links

Specifies if the related links component will be presented in the WebHelp Responsive output. The default value is yes. The webhelp.merge.nested.topics.related.links parameter can used in conjunction with this one to merge the related links from nested topics into the links in the parent topic.

webhelp.show.side.toc

Specifies if a side table of contents will be presented on the right side of each topic in the output. The default value is `yes`.

webhelp.show.top.menu

Specifies if a menu will be presented at the topic of the main page in the output. The default value is `yes`.

webhelp.top.menu.depth

Specifies the maximum depth level of the topics that will be included in the top menu. The default value is 2.

Related Information:

Customizing the WebHelp Responsive Output on page 963
WebHelp Responsive System on page 940

WebHelp Responsive with Feedback Output

To publish a *DITA map* as a **WebHelp Responsive with Feedback** system, follow these steps:

1. Select the **Configure Transformation Scenario(s)** action from the ***DITA Maps Manager*** toolbar.
2. Select the **DITA Map WebHelp Responsive with Feedback** scenario from the **DITA Map** section.
3. If you want to configure the transformation, click the **Edit** button.

   Step Result: This opens an **Edit scenario** configuration dialog box that allows you to configure various options in the following tabs:

   - *Templates Tab* - This tab contains a set of predefined skins that you can use for the layout of your WebHelp system output.
   - *Parameters Tab* - This tab includes numerous parameters that can be set to customize your WebHelp system output. See the *Parameters section* below for details about the most commonly used parameters for WebHelp Responsive transformations.
   - *Filters Tab* - This tab allows you to filter certain content elements from the generated output.
   - *Advanced Tab* - This tab allows you to specify some advanced options for the transformation scenario.
   - *Output Tab* - This tab allows you to configure options that are related to the location where the output is generated.
4. Click **Apply associated** to begin the transformation.
5. Enter the documentation product ID (value for the `webhelp.product.id` parameter) and the documentation version (value for the `webhelp.product.version` parameter).

   When the **DITA Map WebHelp Responsive with Feedback** transformation is complete, your default browser opens the `installation.html` file. This file contains information about the output location, system requirements, installation instructions, and deployment of the output. Follow the *instructions to complete the system deployment*.

Parameters for Customizing WebHelp Responsive with Feedback Output

To customize a transformation scenario, you can edit various parameters, including the following most commonly used ones:

args.default.language

This parameter is used if the language is not detected in the *DITA map*. The default value is `en-us`.

clean.output

Deletes all files from the output folder before the transformation is performed (only no and `yes` values are valid and the default value is no).

editlink.web.author.url

Use this parameter in conjunction with `editlink.web.author.url` to add an *Edit* link next to the topic title in the WebHelp output. When a user clicks the link, the topic is opened in Oxygen XML Web Author where they can make changes that can be saved to a file server. The value should be set as the custom URL of the *main DITA map*. For example, a GitHub custom URL might look like this: `https://getFileContent/oxyengxml/userguide/master/UserGuide.ditamap`. For more details about custom URLs, see *this section*.

editlink.web.author.url

This parameter needs to be used in conjunction with `editlink.remote.ditamap.url` to add an *Edit* link next to the topic title in the WebHelp output. When a user clicks the link, the topic is opened in Oxygen XML Web Author where they can make changes that can be saved to a file server. The value should be set as the URL of the Web Author installation. For example: `https://www.oxygenxml.com/webapp-demo-aws/app/oxygen.html`.

fix.external.refs.com.oxygenxml (Only supported when the DITA OT transformation process is started from Oxygen XML Editor)

The DITA Open Toolkit usually has problems processing references that point to locations outside of the directory of the processed *DITA map*. This parameter is used to specify whether or not the application should try to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references. The fix has no impact on your edited DITA content. Allowed values: `true` or `false` (default).

use.stemming

Controls whether or not you want to include stemming search algorithms into the published output (default setting is `false`).

webhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

webhelp.favicon

The file path that points to an image to be used as a *favicon* in the WebHelp output.

webhelp.reload.stylesheet

Set this parameter to `true` if you have out of memory problems when generating WebHelp. It will increase processing time but decrease the memory footprint. The default value is `false`.

webhelp.search.custom.excludes.file

The path of the file that contains name patterns for HTML files that should not be indexed by the WebHelp search engine. Each exclude pattern must be on a new line. The patterns are considered to be relative to the output directory, and they accept wildcards such as `'*'` (matches zero or more characters) or `'?'` (matches one character). For more information about the patterns, see *https://ant.apache.org/manual/dirtasks.html#patterns*.

webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Editor uses for indexing Japanese content in the WebHelp pages.

webhelp.search.ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the search results that are displayed on the **Search** tab (default setting is `true`).

webhelp.show.changes.and.comments

When set to `yes`, user comments, replies to comments, and tracked changes are published in the WebHelp output. The default value is no.

webhelp.sitemap.base.url

Base URL for all the `loc` elements in the generated `sitemap.xml` file. The value of a `loc` element is computed as the relative file path from the `href` attribute of a `topicref` element from the *DITA map*, appended to this base URL value. The `loc` element is mandatory in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `sitemap.xml` file is not generated.

webhelp.sitemap.change.frequency

The value of the `changefreq` element in the generated `sitemap.xml` file. The `changefreq` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `changefreq` element is not added in `sitemap.xml`. Allowed values: <empty string> (default), `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, `never`.

webhelp.sitemap.priority

The value of the `priority` element in the generated `sitemap.xml` file. It can be set to any fractional number between 0.0 (least important priority) and 1.0 (most important priority). For example, 0.3, 0.5, or 0.8.

The `priority` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `priority` element is not added in `sitemap.xml`.

Parameters Specific to WebHelp Responsive with Feedback Output

webhelp.enable.search.autocomplete

Specifies if the *Autocomplete* feature is enabled in the WebHelp search text field. The default value is `yes`.

webhelp.fragment.after.body

In the generated output it displays a given XHTML fragment after the page body. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.logo_and_title

In the generated output it displays a given XHTML fragment after the logo and title. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.main.page.search

In the generated output it displays a given XHTML fragment after the search field. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.toc_or_tiles

In the generated output it displays a given XHTML fragment after the table of contents or tiles in the main page. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.top_menu

In the generated output it displays a given XHTML fragment after the top menu. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.body

In the generated output it displays a given XHTML fragment before the page body. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.logo_and_title

In the generated output it displays a given XHTML fragment before the logo and title. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.main.page.search

In the generated output it displays a given XHTML fragment before the search field. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.toc_or_tiles

In the generated output it displays a given XHTML fragment before the table of contents or tiles in the main page. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.top_menu

In the generated output it displays a given XHTML fragment before the top menu. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.footer

In the generated output it displays a given XHTML fragment as the page footer. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.head

In the generated output it displays a given XHTML fragment as a page header. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.welcome

In the generated output it displays a given XHTML fragment as a welcome message (or title). The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.merge.nested.topics.related.links

Specifies if the related links from nested topics will be merged with the links in the parent topic. Thus the links will be moved from the topic content to the related links component and all of the links from the same group

(for example, *Related Tasks*, *Related References*, *Related Information*) are merged into a single group. The default value is yes.

webhelp.show.breadcrumb

Specifies if the breadcrumb component will be presented in the output. The default value is yes.

webhelp.show.child.links

Specifies if child links will be generated in the output for all topics that have subtopics. The default value is no.

webhelp.show.indexterms.link

Specifies if an icon that links to the index will be presented in the output. The default value is yes.

webhelp.show.main.page.tiles

Specifies if the tiles component will be presented in the main page of the output. For a *tree* style layout, this parameter should be set to no.

webhelp.show.main.page.toc

Specifies if the table of contents will be presented in the main page of the output. The default value is yes.

webhelp.show.navigation.links

Specifies if navigation links will be presented in the output. The default value is yes.

webhelp.show.print.link

Specifies if a print link or icon will be presented within each topic in the output. The default value is yes.

webhelp.show.related.links

Specifies if the related links component will be presented in the WebHelp Responsive output. The default value is yes. The webhelp.merge.nested.topics.related.links parameter can used in conjunction with this one to merge the related links from nested topics into the links in the parent topic.

webhelp.show.side.toc

Specifies if a side table of contents will be presented on the right side of each topic in the output. The default value is yes.

webhelp.show.top.menu

Specifies if a menu will be presented at the topic of the main page in the output. The default value is yes.

webhelp.top.menu.depth

Specifies the maximum depth level of the topics that will be included in the top menu. The default value is 2.

Related Information:

Customizing the WebHelp Responsive Output on page 963
WebHelp Responsive with Feedback System on page 944

WebHelp Classic Output

To publish a *DITA map* as a **WebHelp Classic** system, follow these steps:

1. Select the **Configure Transformation Scenario(s)** action from the ***DITA Maps Manager*** toolbar.
2. Select the **DITA Map WebHelp Classic** scenario from the **DITA Map** section.
3. If you want to configure the transformation, click the **Edit** button.

   Step Result: This opens an **Edit scenario** configuration dialog box that allows you to configure various options in the following tabs:

   - *Skins Tab* - This tab contains a set of predefined skins that you can use for the layout of your WebHelp system output.
   - *Parameters Tab* - This tab includes numerous parameters that can be set to customize your WebHelp system output. See the *Parameters section* below for details about the most commonly used parameters for WebHelp Responsive transformations.
   - *Filters Tab* - This tab allows you to filter certain content elements from the generated output.
   - *Advanced Tab* - This tab allows you to specify some advanced options for the transformation scenario.
   - *Output Tab* - This tab allows you to configure options that are related to the location where the output is generated.

4. Click **Apply associated** to process the transformation.

   When the **DITA Map WebHelp Classic** transformation is complete, the output is automatically opened in your default browser.

To further customize this transformation, you can edit various parameters, including the following most commonly used ones:

Parameters for Customizing WebHelp Classic Output

To customize a transformation scenario, you can edit various parameters, including the following most commonly used ones:

args.default.language

This parameter is used if the language is not detected in the *DITA map*. The default value is `en-us`.

clean.output

Deletes all files from the output folder before the transformation is performed (only no and `yes` values are valid and the default value is no).

fix.external.refs.com.oxygenxml (Only supported when the DITA OT transformation process is started from Oxygen XML Editor)

The DITA Open Toolkit usually has problems processing references that point to locations outside of the directory of the processed *DITA map*. This parameter is used to specify whether or not the application should try to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references. The fix has no impact on your edited DITA content. Allowed values: `true` or `false` (default).

use.stemming

Controls whether or not you want to include stemming search algorithms into the published output (default setting is `false`).

webhelp.body.script

URL value that specifies the location of a well-formed XHTML file containing the custom script that will be copied in the `<body>` section of every WebHelp page.

webhelp.copyright

Adds a small copyright text that appears at the end of the **Table of Contents** pane.

webhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

webhelp.favicon

The file path that points to an image to be used as a *favicon* in the WebHelp output.

webhelp.footer.file

Path to an XML file that includes the footer content for your WebHelp output pages. You can use this parameter to integrate social media features (such as widgets for Facebook™, Twitter™, Google Analytics, or Google+™). The file must be well-formed, each widget must be in separate `div` or `span` element, and the code for each `script` element is included in an XML comment (also, the start and end tags for the XML comment must be on a separate line). The following code exert is an example for adding a Facebook™ widget:

```
<div id="facebook">
  <div id="fb-root"/>
  <script>
    <!-- (function(d, s, id) { var js, fjs = d.getElementsByTagName(s)[0];
      if (d.getElementById(id)) return;
      js = d.createElement(s); js.id = id;
      js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
      fjs.parentNode.insertBefore(js, fjs); }
       (document, 'script', 'facebook-jssdk')); -->
    </script>
    <div data-share="true" data-show-faces="true" data-action="like"
       data-layout="standard" class="fb-like"/>
</div>
```

webhelp.footer.include

Specifies whether or not to include footer in each WebHelp page. Possible values: `yes`, no. If set to no, no footer is added to the WebHelp pages. If set to `yes` and the `webhelp.footer.file` parameter has a value,

then the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default Oxygen XML Editor footer is inserted in each WebHelp page.

webhelp.google.search.results

A file path that specifies the location of a well-formed XHTML file containing the Google Custom Search Engine element `gcse:searchresults-only`. You can use all supported attributes for this element. It is recommend to set the `linkTarget` attribute to `frm` for frameless (*iframe*) version of WebHelp or to `contentWin` for the frameset version of WebHelp. The default value for this attribute is `_blank` and the search results will be loaded in a new window. If this parameter is not specified, the following code will be used `<gcse:searchresults-only linkTarget="frm"></gcse:searchresults-only>`

webhelp.google.search.script

A file path that specifies the location of a well-formed XHTML file containing the Custom Search Engine script from Google.

webhelp.head.script

URL value that specifies the location of a well-formed XHTML file containing the custom script that will be copied in the `<head>` section of every WebHelp page.

webhelp.logo.image.target.url

Specifies a target URL that is set on the logo image. When you click the logo image, you will be redirected to this address.

webhelp.logo.image

Specifies a path to an image displayed as a logo in the left side of the output header.

webhelp.reload.stylesheet

Set this parameter to `true` if you have out of memory problems when generating WebHelp. It will increase processing time but decrease the memory footprint. The default value is `false`.

webhelp.search.custom.excludes.file

The path of the file that contains name patterns for HTML files that should not be indexed by the WebHelp search engine. Each exclude pattern must be on a new line. The patterns are considered to be relative to the output directory, and they accept wildcards such as `'*'` (matches zero or more characters) or `'?'` (matches one character). For more information about the patterns, see *https://ant.apache.org/manual/dirtasks.html#patterns*.

webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Editor uses for indexing Japanese content in the WebHelp pages.

webhelp.search.ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the search results that are displayed on the **Search** tab (default setting is `true`).

webhelp.show.changes.and.comments

When set to `yes`, user comments, replies to comments, and tracked changes are published in the WebHelp output. The default value is no.

webhelp.sitemap.base.url

Base URL for all the `loc` elements in the generated `sitemap.xml` file. The value of a `loc` element is computed as the relative file path from the `href` attribute of a `topicref` element from the *DITA map*, appended to this base URL value. The `loc` element is mandatory in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `sitemap.xml` file is not generated.

webhelp.sitemap.change.frequency

The value of the `changefreq` element in the generated `sitemap.xml` file. The `changefreq` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `changefreq` element is not added in `sitemap.xml`. Allowed values: <empty string> (default), `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, `never`.

webhelp.sitemap.priority

The value of the `priority` element in the generated `sitemap.xml` file. It can be set to any fractional number between 0.0 (least important priority) and 1.0 (most important priority). For example, 0.3, 0.5, or 0.8.

The `priority` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `priority` element is not added in `sitemap.xml`.

Related Information:

Customizing WebHelp Classic Systems on page 999
WebHelp Classic System on page 988

WebHelp Classic With Feedback Output

To publish a *DITA map* as a **WebHelp Classic with Feedback** system, follow these steps:

1. Select the **Configure Transformation Scenario(s)** action from the ***DITA Maps Manager*** toolbar.
2. Select the **DITA Map WebHelp Classic with Feedback** scenario from the **DITA Map** section.
3. If you want to configure the transformation, click the **Edit** button.

   Step Result: This opens an **Edit scenario** configuration dialog box that allows you to configure various options in the following tabs:

   - *Skins Tab* - This tab contains a set of predefined skins that you can use for the layout of your WebHelp system output.
   - *Parameters Tab* - This tab includes numerous parameters that can be set to customize your WebHelp system output. See the *Parameters section* below for details about the most commonly used parameters for WebHelp Responsive transformations.
   - *Filters Tab* - This tab allows you to filter certain content elements from the generated output.
   - *Advanced Tab* - This tab allows you to specify some advanced options for the transformation scenario.
   - *Output Tab* - This tab allows you to configure options that are related to the location where the output is generated.
4. Click **Apply associated** to begin the transformation.
5. Enter the documentation product ID (value for the `webhelp.product.id` parameter) and the documentation version (value for the `webhelp.product.version` parameter).

   When the **DITA Map WebHelp Classic with Feedback** transformation is complete, your default browser opens the `installation.html` file. This file contains information about the output location, system requirements, installation instructions, and deployment of the output. Follow the *instructions to complete the system deployment*.

To watch our video demonstration about the feedback-enabled WebHelp system, go to *https://www.oxygenxml.com/demo/Feedback_Enabled_WebHelp.html*.

Parameters for Customizing WebHelp Classic with Feedback Output

To customize a transformation scenario, you can edit various parameters, including the following most commonly used ones:

args.default.language

This parameter is used if the language is not detected in the *DITA map*. The default value is `en-us`.

clean.output

Deletes all files from the output folder before the transformation is performed (only no and `yes` values are valid and the default value is no).

fix.external.refs.com.oxygenxml (Only supported when the DITA OT transformation process is started from Oxygen XML Editor)

The DITA Open Toolkit usually has problems processing references that point to locations outside of the directory of the processed *DITA map*. This parameter is used to specify whether or not the application should try to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references. The fix has no impact on your edited DITA content. Allowed values: `true` or `false` (default).

use.stemming

Controls whether or not you want to include stemming search algorithms into the published output (default setting is `false`).

webhelp.body.script

URL value that specifies the location of a well-formed XHTML file containing the custom script that will be copied in the `<body>` section of every WebHelp page.

webhelp.copyright

Adds a small copyright text that appears at the end of the **Table of Contents** pane.

webhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

webhelp.favicon

The file path that points to an image to be used as a *favicon* in the WebHelp output.

webhelp.footer.file

Path to an XML file that includes the footer content for your WebHelp output pages. You can use this parameter to integrate social media features (such as widgets for Facebook™, Twitter™, Google Analytics, or Google+™). The file must be well-formed, each widget must be in separate `div` or `span` element, and the code for each `script` element is included in an XML comment (also, the start and end tags for the XML comment must be on a separate line). The following code exert is an example for adding a Facebook™ widget:

```
<div id="facebook">
  <div id="fb-root"/>
  <script>
    <!-- (function(d, s, id) { var js, fjs = d.getElementsByTagName(s)[0];
      if (d.getElementById(id)) return;
      js = d.createElement(s); js.id = id;
      js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
      fjs.parentNode.insertBefore(js, fjs); }
       (document, 'script', 'facebook-jssdk')); -->
    </script>
    <div data-share="true" data-show-faces="true" data-action="like"
       data-layout="standard" class="fb-like"/>
</div>
```

webhelp.footer.include

Specifies whether or not to include footer in each WebHelp page. Possible values: `yes`, no. If set to no, no footer is added to the WebHelp pages. If set to `yes` and the `webhelp.footer.file` parameter has a value, then the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default Oxygen XML Editor footer is inserted in each WebHelp page.

webhelp.google.search.results

A file path that specifies the location of a well-formed XHTML file containing the Google Custom Search Engine element `gcse:searchresults-only`. You can use all supported attributes for this element. It is recommend to set the `linkTarget` attribute to `frm` for frameless (*iframe*) version of WebHelp or to `contentWin` for the frameset version of WebHelp. The default value for this attribute is `_blank` and the search results will be loaded in a new window. If this parameter is not specified, the following code will be used `<gcse:searchresults-only linkTarget="frm"></gcse:searchresults-only>`

webhelp.google.search.script

A file path that specifies the location of a well-formed XHTML file containing the Custom Search Engine script from Google.

webhelp.head.script

URL value that specifies the location of a well-formed XHTML file containing the custom script that will be copied in the `<head>` section of every WebHelp page.

webhelp.logo.image.target.url

Specifies a target URL that is set on the logo image. When you click the logo image, you will be redirected to this address.

webhelp.logo.image

Specifies a path to an image displayed as a logo in the left side of the output header.

webhelp.reload.stylesheet

Set this parameter to `true` if you have out of memory problems when generating WebHelp. It will increase processing time but decrease the memory footprint. The default value is `false`.

webhelp.search.custom.excludes.file

The path of the file that contains name patterns for HTML files that should not be indexed by the WebHelp search engine. Each exclude pattern must be on a new line. The patterns are considered to be relative to the output directory, and they accept wildcards such as `'*'` (matches zero or more characters) or `'?'` (matches one character). For more information about the patterns, see *https://ant.apache.org/manual/dirtasks.html#patterns*.

webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Editor uses for indexing Japanese content in the WebHelp pages.

webhelp.search.ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the search results that are displayed on the **Search** tab (default setting is `true`).

webhelp.show.changes.and.comments

When set to `yes`, user comments, replies to comments, and tracked changes are published in the WebHelp output. The default value is no.

webhelp.sitemap.base.url

Base URL for all the `loc` elements in the generated `sitemap.xml` file. The value of a `loc` element is computed as the relative file path from the `href` attribute of a `topicref` element from the *DITA map*, appended to this base URL value. The `loc` element is mandatory in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `sitemap.xml` file is not generated.

webhelp.sitemap.change.frequency

The value of the `changefreq` element in the generated `sitemap.xml` file. The `changefreq` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `changefreq` element is not added in `sitemap.xml`. Allowed values: <empty string> (default), `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, `never`.

webhelp.sitemap.priority

The value of the `priority` element in the generated `sitemap.xml` file. It can be set to any fractional number between 0.0 (least important priority) and 1.0 (most important priority). For example, 0.3, 0.5, or 0.8. The `priority` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `priority` element is not added in `sitemap.xml`.

Related Information:

Customizing WebHelp Classic Systems on page 999

WebHelp Classic with Feedback System on page 992

WebHelp Classic Mobile Output

To publish a *DITA map* as a **WebHelp Classic Mobile** system, follow these steps:

1. Select the **Configure Transformation Scenario(s)** action from the ***DITA Maps Manager*** toolbar.
2. Select the **DITA Map WebHelp Classic Mobile** scenario from the **DITA Map** section.
3. If you want to configure the transformation, click the **Edit** button.

   Step Result: This opens an **Edit scenario** configuration dialog box that allows you to configure various options in the following tabs:

   - *Parameters Tab* - This tab includes numerous parameters that can be set to customize your WebHelp system output. See the *Parameters section* below for details about the most commonly used parameters for WebHelp Responsive transformations.
   - *Filters Tab* - This tab allows you to filter certain content elements from the generated output.
   - *Advanced Tab* - This tab allows you to specify some advanced options for the transformation scenario.
   - *Output Tab* - This tab allows you to configure options that are related to the location where the output is generated.
4. Click **Apply associated** to process the transformation.

   When the **DITA Map WebHelp Classic Mobile** transformation is complete, the output is automatically opened in your default browser.

Parameters for Customizing WebHelp Classic Mobile Output

To customize a transformation scenario, you can edit various parameters, including the following most commonly used ones:

args.default.language

This parameter is used if the language is not detected in the *DITA map*. The default value is `en-us`.

clean.output

Deletes all files from the output folder before the transformation is performed (only no and `yes` values are valid and the default value is no).

fix.external.refs.com.oxygenxml (Only supported when the DITA OT transformation process is started from Oxygen XML Editor)

The DITA Open Toolkit usually has problems processing references that point to locations outside of the directory of the processed *DITA map*. This parameter is used to specify whether or not the application should try to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references. The fix has no impact on your edited DITA content. Allowed values: `true` or `false` (default).

use.stemming

Controls whether or not you want to include stemming search algorithms into the published output (default setting is `false`).

webhelp.copyright

Adds a small copyright text that appears at the end of the **Table of Contents** pane.

webhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

webhelp.favicon

The file path that points to an image to be used as a *favicon* in the WebHelp output.

webhelp.footer.file

Path to an XML file that includes the footer content for your WebHelp output pages. You can use this parameter to integrate social media features (such as widgets for Facebook™, Twitter™, Google Analytics, or Google+™). The file must be well-formed, each widget must be in separate `div` or `span` element, and the code for each `script` element is included in an XML comment (also, the start and end tags for the XML comment must be on a separate line). The following code exert is an example for adding a Facebook™ widget:

```
<div id="facebook">
  <div id="fb-root"/>
  <script>
    <!-- (function(d, s, id) { var js, fjs = d.getElementsByTagName(s)[0];
      if (d.getElementById(id)) return;
      js = d.createElement(s); js.id = id;
      js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
      fjs.parentNode.insertBefore(js, fjs); }
       (document, 'script', 'facebook-jssdk')); -->
    </script>
    <div data-share="true" data-show-faces="true" data-action="like"
       data-layout="standard" class="fb-like"/>
</div>
```

webhelp.footer.include

Specifies whether or not to include footer in each WebHelp page. Possible values: `yes`, no. If set to no, no footer is added to the WebHelp pages. If set to `yes` and the `webhelp.footer.file` parameter has a value, then the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default Oxygen XML Editor footer is inserted in each WebHelp page.

webhelp.google.search.results

A file path that specifies the location of a well-formed XHTML file containing the Google Custom Search Engine element `gcse:searchresults-only`. You can use all supported attributes for this element. It is recommend to set the `linkTarget` attribute to `frm` for frameless (*iframe*) version of WebHelp or to `contentWin` for the frameset version of WebHelp. The default value for this attribute is `_blank` and the search results will be loaded in a new window. If this parameter is not specified, the following code will be used `<gcse:searchresults-only linkTarget="frm"></gcse:searchresults-only>`

webhelp.google.search.script

A file path that specifies the location of a well-formed XHTML file containing the Custom Search Engine script from Google.

webhelp.head.script

URL value that specifies the location of a well-formed XHTML file containing the custom script that will be copied in the `<head>` section of every WebHelp page.

webhelp.reload.stylesheet

Set this parameter to `true` if you have out of memory problems when generating WebHelp. It will increase processing time but decrease the memory footprint. The default value is `false`.

webhelp.search.custom.excludes.file

The path of the file that contains name patterns for HTML files that should not be indexed by the WebHelp search engine. Each exclude pattern must be on a new line. The patterns are considered to be relative to the output directory, and they accept wildcards such as `'*'` (matches zero or more characters) or `'?'` (matches one character). For more information about the patterns, see *https://ant.apache.org/manual/dirtasks.html#patterns*.

webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Editor uses for indexing Japanese content in the WebHelp pages.

webhelp.sitemap.base.url

Base URL for all the `loc` elements in the generated `sitemap.xml` file. The value of a `loc` element is computed as the relative file path from the `href` attribute of a `topicref` element from the *DITA map*, appended to this base URL value. The `loc` element is mandatory in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `sitemap.xml` file is not generated.

webhelp.sitemap.change.frequency

The value of the `changefreq` element in the generated `sitemap.xml` file. The `changefreq` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `changefreq` element is not added in `sitemap.xml`. Allowed values: <empty string> (default), `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, `never`.

webhelp.sitemap.priority

The value of the `priority` element in the generated `sitemap.xml` file. It can be set to any fractional number between 0.0 (least important priority) and 1.0 (most important priority). For example, 0.3, 0.5, or 0.8. The `priority` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `priority` element is not added in `sitemap.xml`.

Related Information:

Customizing WebHelp Classic Mobile Systems on page 1024

WebHelp Classic Mobile System (Deprecated) on page 1023

DITA Map to PDF Output Customization

Oxygen XML Editor comes bundled with the DITA Open Toolkit that provides a mechanism for converting *DITA maps* to PDF output. There are numerous ways that you can customize the PDF output and the topics in this section discuss some of those possibilities.

Creating a DITA Map to PDF Transformation Scenario

To create a *DITA Map to PDF* transformation scenario, follow these steps:

1. Click the **Configure Transformation Scenario(s)** button from the ***DITA Maps Manager*** toolbar.
2. Select **DITA Map PDF** and click the **Edit** button (or use the **Duplicate** button if your *framework* is read-only).
3. Use the various tabs to configure the transformation scenario. In the **Parameters** tab, you can use a variety of parameters to customize the output. For example, the following parameters are just a few of the most commonly used ones:
   - `show.changes.and.comments` - If set to `yes`, user comments, replies to comments, and *tracked changes* are published in the PDF output.

- `customization.dir` - Specifies the path to a customization directory.

4. Click **OK** and then the **Apply Associated** button to run the transformation scenario.

Creating a Customization Directory for PDF Output

DITA Open Toolkit PDF output can be customized in several ways:

- Creating a DITA Open Toolkit plugin that adds extensions to the PDF plugin. More details can be found in the *DITA Open Toolkit Documentation*.
- Creating a customization directory and using it from the PDF transformation scenario. A small example of this procedure can be found below.

How to Create a Customization Directory for PDF Output

The following procedure explains how to do a basic customization of the PDF output by setting up a customization directory. An example of a common use case is embedding a company logo image in the front matter of the book.

1. Copy the entire directory: *DITA-OT-DIR*`\plugins\org.dita.pdf2\Customization` to another location (for instance, `C:\Customization`).
2. Copy your logo image to: `C:\Customization\common\artwork\logo.png`.
3. Rename `C:\Customization\catalog.xml.orig` to: `C:\Customization\catalog.xml`.
4. Open the `catalog.xml` in Oxygen XML Editor and *uncomment* this line:

```
<!--uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/-->
```

   It now looks like this:

```
<uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/>
```

5. Rename the file: `C:\Customization\fo\xsl\custom.xsl.orig` to: `C:\Customization\fo\xsl\custom.xsl`
6. Open the `custom.xsl` file in Oxygen XML Editor and create the template called **createFrontCoverContents** for DITA-OT 2.4.4 (or createFrontMatter_1.0 for DITA-OT 1.8.5).

   Tip: You can copy the same template from *DITA-OT-DIR*`\plugins\org.dita.pdf2\xsl\fo\front-matter.xsl` and modify it in whatever way necessary to achieve your specific goal. This new template in the `custom.xsl` file will override the same template from *DITA-OT-DIR*`\plugins\org.dita.pdf2\xsl\fo\front-matter.xsl`.

   DITA OT 2.4.4 Example:

   For example, if you are using DITA OT 2.4.4, the `custom.xsl` could look like this:

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www.w3.org/1999/XSL/Format"
    version="2.0">

<xsl:template name="createFrontCoverContents">
 <!-- set the title -->
 <fo:block xsl:use-attribute-sets="__frontmatter__title">
  <xsl:choose>
   <xsl:when test="$map/*[contains(@class,' topic/title ')][1]">
    <xsl:apply-templates select="$map/*[contains(@class,' topic/title ')][1]"/>
      </xsl:when>
      <xsl:when test="$map//*[contains(@class,' bookmap/mainbooktitle ')][1]">
        <xsl:apply-templates select="$map//*[contains
                                    (@class,' bookmap/mainbooktitle ')][1]"/>
      </xsl:when>
      <xsl:when test="//*[contains(@class, ' map/map ')]/@title">
        <xsl:value-of select="//*[contains(@class, ' map/map ')]/@title"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="/descendant::*[contains
           (@class, ' topic/topic ')][1]/*[contains(@class, ' topic/title ')]"/>
   </xsl:otherwise>
  </xsl:choose>
 </fo:block>

 <!-- set the subtitle -->
 <xsl:apply-templates select="$map//*[contains
                                    (@class,' bookmap/booktitlealt ')]"/>
 <fo:block xsl:use-attribute-sets="__frontmatter__owner">
  <xsl:apply-templates select="$map//*[contains(@class,' bookmap/bookmeta ')]"/>
```

```
 </fo:block>

 <!-- Load the image logo -->
  <fo:block text-align="center" width="100%">
   <fo:external-graphic
      src="url({concat($artworkPrefix,
                        '/Customization/OpenTopic/common/artwork/logo.png')})"
    />
  </fo:block>
 </xsl:template>

</xsl:stylesheet>
```

DITA OT 1.8.5 Example:

For example, if you are using DITA OT 1.8.5, the `custom.xsl` could look like this:

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www.w3.org/1999/XSL/Format"
    version="1.1">

<xsl:template name="createFrontMatter_1.0">
   <fo:page-sequence master-reference="front-matter"
                        xsl:use-attribute-sets="__force__page__count">
     <xsl:call-template name="insertFrontMatterStaticContents"/>
       <fo:flow flow-name="xsl-region-body">
         <fo:block xsl:use-attribute-sets="__frontmatter">
     <!-- set the title -->
           <fo:block xsl:use-attribute-sets="__frontmatter__title">
            <xsl:choose>
             <xsl:when test="$map/*[contains(@class,' topic/title ')][1]">
     <xsl:apply-templates select="$map/*[contains(@class,' topic/title ')][1]"/>
                 </xsl:when>
    <xsl:when test="$map//*[contains(@class,' bookmap/mainbooktitle ')][1]">
     <xsl:apply-templates select="$map//*[contains
                                   (@class,' bookmap/mainbooktitle ')][1]"/>
         </xsl:when>
    <xsl:when test="//*[contains(@class, ' map/map ')]/@title">
      <xsl:value-of select="//*[contains(@class, ' map/map ')]/@title"/>
         </xsl:when>
      <xsl:otherwise>
    <xsl:value-of select="/descendant::*[contains
           (@class, ' topic/topic ')][1]/*[contains(@class, ' topic/title ')]"/>
      </xsl:otherwise>
         </xsl:choose>
     </fo:block>

  <!-- set the subtitle -->
     <xsl:apply-templates select="$map//*[contains
                                        (@class,' bookmap/booktitlealt ')]"/>

        <fo:block xsl:use-attribute-sets="__frontmatter__owner">
            <xsl:apply-templates select="$map//*[contains
                                              (@class,' bookmap/bookmeta ')]"/>
        </fo:block>

        <fo:block text-align="center" width="100%">
           <fo:external-graphic src="url({concat($artworkPrefix,
                        '/Customization/OpenTopic/common/artwork/logo.png')})"/>
         </fo:block>

    </fo:block>

   <!--<xsl:call-template name="createPreface"/>-->

            </fo:flow>
        </fo:page-sequence>
        <xsl:call-template name="createNotices"/>
    </xsl:template>
</xsl:stylesheet>
```

7. Edit the **DITA Map PDF** transformation scenario and in the **Parameters** tab, set the customization.dir parameter to `C:\Customization`.

Tip: For other specific examples, see *DITA-OT 2.3 Documentation - PDF Customization Plugin*.

Related Information:

Automatic PDF plugin customization generator by Jarno Elovirta.

DITA-OT 1.8 Documentation - PDF Customization Plugin

DITA-OT 2.3 Documentation - PDF Customization Plugin

Customizing the Header and Footer in PDF Output

The XSLT stylesheet *DITA-OT-DIR*`/plugins/org.dita.pdf2/xsl/fo/static-content.xsl` contains templates that output the static header and footers for various parts of the PDF such as the prolog, table of contents, front matter, or body.

The templates for generating a footer for pages in the body are called `insertBodyOddFooter` or `insertBodyEvenFooter`.

These templates get the static content from resource files that depend on the language used for generating the PDF. The default resource file is *DITA-OT-DIR*`/plugins/org.dita.pdf2/cfg/common/vars/en.xml`. These resource files contain variables (such as *Body odd footer*) that can be set to specific user values.

Instead of modifying these resource files directly, they can be overwritten with modified versions of the resources in a PDF customization directory as explained in *Creating a Customization Directory for PDF Output* on page 1690.

Adding a Watermark to PDF Output

To add a watermark to the PDF output of a *DITA map* transformation, create a DITA-OT customization directory and use it from a **DITA Map to PDF** transformation scenario, as in the following procedure:

1. Copy the entire directory: *DITA-OT-DIR*`\plugins\org.dita.pdf2\Customization` to another location (for instance, `C:\Customization`).
2. Copy your watermark image (for example, `watermark.png`) to: `C:\Customization\common\artwork\watermark.png`.
3. Rename the `C:\Customization\catalog.xml.orig` file to: `C:\Customization\catalog.xml`.
4. Open the `catalog.xml` in Oxygen XML Editor and *uncomment* this line:

   ```
   <!--uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/-->
   ```

   The uncommented line should look like this:

   ```
   <uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/>
   ```

5. Rename the file: `C:\Customization\fo\xsl\custom.xsl.orig` to: `C:\Customization\fo\xsl\custom.xsl`
6. Open the `C:\Customization\fo\xsl\custom.xsl` file in Oxygen XML Editor to overwrite two XSLT templates:

   - The first template is located in the XSLT stylesheet *DITA-OT-DIR*`\plugins\org.dita.pdf2\xsl\fo\static-content.xsl` and we override it specifying a watermark image for every page in the PDF content, using a *block-container* element that references the watermark image file:

     ```
     <fo:static-content flow-name="odd-body-header">
            <fo:block-container absolute-position="absolute"
               top="-2cm" left="-3cm" width="21cm" height="29.7cm"
               background-image="{concat($artworkPrefix,
     '/Customization/OpenTopic/common/artwork/watermark.png')}">
              <fo:block/>
            </fo:block-container>
           <fo:block xsl:use-attribute-sets="__body__odd__header">
               <xsl:call-template name="insertVariable">
                   <xsl:with-param name="theVariableID" select="'Body odd header'"/>
                   <xsl:with-param name="theParameters">
                      <prodname>
                          <xsl:value-of select="$productName"/>
                      </prodname>
                      <heading>
                    <fo:inline xsl:use-attribute-sets="__body__odd__header__heading">
                           <fo:retrieve-marker retrieve-class-name="current-header"/>
                    </fo:inline>
                     </heading>
                    <pagenum>
                   <fo:inline xsl:use-attribute-sets="__body__odd__header__pagenum">
                          <fo:page-number/>
                   </fo:inline>
                 </pagenum>
             </xsl:with-param>
           </xsl:call-template>
         </fo:block>
       </fo:static-content>

     </xsl:template>
     ```

- The second template that we override is located in the XSLT stylesheet *DITA-OT-DIR*`\plugins\org.dita.pdf2\xsl\fo\commons.xsl` and is used for styling the first page of the output. We also override it by copying the original template content in our `custom.xsl` and adding the `block-container` element that references the watermark image file:

```
<xsl:template name="createFrontMatter_1.0">
      <fo:page-sequence master-reference="front-matter"
xsl:use-attribute-sets="__force__page__count">
          <xsl:call-template name="insertFrontMatterStaticContents"/>
          <fo:flow flow-name="xsl-region-body">
              <fo:block-container absolute-position="absolute"
                  top="-2cm" left="-3cm" width="21cm" height="29.7cm"
                  background-image="{concat($artworkPrefix,
'/Customization/OpenTopic/common/artwork/watermark.png')}">
                  <fo:block/>
              </fo:block-container>
  <fo:block xsl:use-attribute-sets="__frontmatter">
       <!-- set the title -->
    <fo:block xsl:use-attribute-sets="__frontmatter__title">
     <xsl:choose>
      <xsl:when test="$map/*[contains(@class,' topic/title ')][1]">
    <xsl:apply-templates select="$map/*[contains(@class,' topic/title ')][1]"/>
      </xsl:when>
      <xsl:when test="$map//*[contains(@class,' bookmap/mainbooktitle ')][1]">
    <xsl:apply-templates select="$map//*[contains
(@class,' bookmap/mainbooktitle ')][1]"/>
          </xsl:when>
          <xsl:when test="//*[contains(@class, ' map/map ')]/@title">
              <xsl:value-of select="//*[contains(@class, ' map/map ')]/@title"/>
          </xsl:when>
            <xsl:otherwise>
             <xsl:value-of select="/descendant::*[contains
(@class, ' topic/topic ')][1]/*[contains(@class, ' topic/title ')]"/>
            </xsl:otherwise>
      </xsl:choose>
    </fo:block>

   <!-- set the subtitle -->
   <xsl:apply-templates select="$map//*[contains
(@class,' bookmap/booktitlealt ')]"/>

      <fo:block xsl:use-attribute-sets="__frontmatter__owner">
         <xsl:apply-templates select="$map//*[contains
(@class,' bookmap/bookmeta ')]"/>
      </fo:block>

      </fo:block>

   <!--<xsl:call-template name="createPreface"/>-->

      </fo:flow>
        </fo:page-sequence>
        <xsl:if test="not($retain-bookmap-order)">
            <xsl:call-template name="createNotices"/>
        </xsl:if>
    </xsl:template>
```

7. Edit your **DITA Map PDF** transformation scenario. In the **Parameters** tab, set the customization.dir parameter to `C:\Customization`.

Related Information:

Adding a Watermark in DITA Map to XHTML Output on page 1697

Force Page Breaks Between Two Block Elements in PDF Output

Suppose that in your DITA content you have two *block elements*, such as two paragraphs:

```
<p>First para</p>
<p>Second para</p>
```

and you want to force a page break between them in the PDF output.

Here is how you can implement a DITA Open Toolkit *plugin* that would achieve this:

1. Define your custom processing instruction that marks the place where a page break should be inserted in the PDF, for example:

```
<p>First para</p>
  <?pagebreak?>
<p>Second para</p>
```

2. Locate the **DITA Open Toolkit** distribution and in the `plugins` directory create a new *plugin* folder (for example, *DITA-OT-DIR*`/plugins/pdf-page-break`).

3. In this new folder, create a new `plugin.xml` file with the following content:

```
<plugin id="com.yourpackage.pagebreak">
  <feature extension="package.support.name" value="Force Page Break Plugin"/>
  <feature extension="package.support.email" value="support@youremail.com"/>
  <feature extension="package.version"value="1.0.0"/>
  <feature extension="dita.xsl.xslfo" value="pageBreak.xsl" type="file"/>
</plugin>
```

   The most important feature in the *plugin* is that it will add a new XSLT stylesheet to the XSL processing that produces the PDF content.
4. In the same folder, create an XSLT stylesheet named `pageBreak.xsl` with the following content:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                     xmlns:fo="http://www.w3.org/1999/XSL/Format"version="1.0">
  <xsl:template match="processing-instruction('pagebreak')">
    <fo:block break-after="page"/>
  </xsl:template>
</xsl:stylesheet>
```

5. *Install your plugin in the DITA Open Toolkit.*

Customizing `note` Images in PDF

To customize the images that appear next to each type of note in the PDF output, use a *PDF customization folder* with the following procedure:

1. Copy the *`DITA-OT-DIR`*`/plugins/org.dita.pdf2/cfg/common/vars/en.xml` file to the `[CUSTOMIZATION_DIR]\common\vars` folder.
2. Edit the copied `en.xml` file and modify, for example, the path to the image for `<note>` element with the `type` attribute set to `notice` from:

```
<variable id="notice Note Image
  Path">Configuration/OpenTopic/cfg/common/artwork/important.png</variable>
```

   to:

```
<variable id="notice Note Image
  Path">Customization/OpenTopic/common/artwork/notice.gif</variable>
```

3. Add your custom **notice** image to `[CUSTOMIZATION_DIR]\common\artwork\notice.gif`.
4. Edit the **DITA Map PDF** transformation scenario and in the **Parameters** tab set the path for the customization.dir property to point to the customization folder.

Show Comments and Tracked Changes in PDF Output

To include comments and *tracked changes* (stored within your DITA topics) in the PDF output, follow these steps:

1. Click the **Configure Transformation Scenario(s)** button from the ***DITA Maps Manager*** toolbar.
2. Select **DITA Map PDF** and click the **Edit** button (or use the **Duplicate** button if your *framework* is read-only).
3. In the **Parameters** tab, set the value of the show.changes.and.comments parameter to `yes`.
4. Click **OK** and then the **Apply Associated** button to run the transformation scenario.

   Result: Comment threads and tracked changes will now appear in the PDF output. Details about each comment or change will be available in the footer section for each page.

Set a Font for PDF Output Generated with FO Processor

When a *DITA map* is transformed to PDF using an FO processor and it contains some Unicode characters that cannot be rendered by the default PDF fonts, a font that is capable of rendering these characters must be configured and embedded in the PDF result.

The settings that must be modified for configuring a font for the built-in FO processor are detailed in *Add a Font to the Built-in FO Processor*.

DITA OT PDF Font Mapping

The DITA OT contains a file *`DITA-OT-DIR`*`/plugins/org.dita.pdf2/cfg/fo/font-mappings.xml` that maps logical fonts used in the XSLT stylesheets to physical fonts that will be used by the FO processor to generate the PDF output.

The XSLT stylesheets used to generate the XSL-FO output contain code like this:

```
<xsl:attribute name="font-family">monospace</xsl:attribute>
```

The font-family is defined to be *monospace*, but *monospace* is just an alias. It is not a physical font name. Therefore, another stage in the PDF generation takes this *monospace* alias and looks in the `font-mappings.xml`.

If it finds a mapping like this:

```
<aliases>
      <alias name="monospace">Monospaced</alias>
 </aliases>
```

then it looks to see if the *Monospaced* has a *logical-font* definition and if so, it will use the *physical-font* specified there:

```
<logical-font name="Monospaced">
      <physical-font char-set="default">
        <font-face>Courier New, Courier</font-face>
      </physical-font>
............
</logical-font>
```

Important:

If no alias mapping is found for a font-family specified in the XSLT stylesheets, the processing defaults to **Helvetica**.

Related Information:

DITA Map to PDF WYSIWYG Transformation

Oxygen XML Editor comes bundled with a DITA OT plugin that allows you to convert *DITA maps* to PDF using a CSS layout processor. Oxygen XML Editor also comes bundled with a built-in CSS-based PDF processing engine called **Chemistry**. For those who are familiar with CSS, this makes it very easy to style and customize the PDF output of your DITA projects without having to work with *xsl:fo* customizations.

Oxygen XML Editor supports the following processors (not included in the Oxygen XML Editor installation kit):

- **Oxygen Chemistry** - A built-in processor that is bundled with Oxygen XML Editor.
- **Prince Print with CSS** - A third-party component that needs to be purchased from *http://www.princexml.com*.
- **Antenna House Formatter** - A third-party component that needs to be purchased from *http://www.antennahouse.com/antenna1/formatter/*.

The DITA-OT plugin is located in the following directory: *`DITA-OT-DIR`*`/plugins/com.oxygenxml.pdf.css`.

Although it includes a set of CSS files in its `css` subfolder, when this plugin is used in Oxygen XML Editor, the CSS files located in the `${frameworks}` directory take precedence.

Creating the Transformation Scenario

To create an **DITA Map to PDF WYSIWYG** transformation scenario, follow these steps:

1. Click the **Configure Transformation Scenario(s)** button from the ***DITA Maps Manager*** toolbar.
2. Select **DITA Map PDF - WYSIWYG**.
3. In the **Parameters** tab, configure the following parameters:
   - `css.processor.type` (if you want to use the built-in **Oxygen Chemistry** processor) - Set the value as `chemistry`.
   - `css.processor.path.prince` (if you are using the **Prince Print with CSS** processor) - Specifies the path to the Prince executable file that will be run to produce the PDF. If you installed Prince using its default settings, you can leave this blank.
   - `css.processor.path.antenna-house` (if you are using the **Antenna House Formatter** processor) - Specifies the path to the Antenna House executable file that will be run to produce the PDF. If you installed Antenna House using its default settings, you can leave this blank.

- `show.changes.and.comments` - When set to `yes`, user comments, replies to comments, and *tracked changes* are published in the PDF output. The default value is no.
- `dita.css.list` - Allows you to specify a list of CSS URLs to be used by the PDF processor. The files must have URL syntax and be separated using semicolons. If the value is empty, the current selection from the **Styles** drop-down menu is used.
- `args.css` - Allows you to specify a list of CSS URLs to be used in addition to those specified in the `dita.css.list` parameter OR in addition to the CSS that is currently selected in the **Styles** drop-down menu. The files must have URL syntax and be separated using semicolons. Also, the `dita.css.list` parameter must be left empty to use these files in addition to the selection in the **Styles** drop-down menu.

4. Click **OK** and run the transformation scenario.

Customizing the Styles (for Output and Editing)

If you need to change the styles in the associated CSS, make sure you install Oxygen XML Editor in a folder where you have full read and write privileges (for instance, your user home directory). This is due to the fact that the installed files are usually placed in a read-only folder (for instance, in Windows, Oxygen XML Editor is installed in the `Program Files` folder where the users do not have change rights).

To change the styles of an element you need to create an additional CSS file that will store the customization rules. Once you have created this file, you need to instruct the editor how to use this additional CSS. This can be done in two ways:

- Create an alternate CSS for the DITA document type:

  1. Follow the procedure for adding an alternate CSS file in *Customizing the Main CSS of a Framework* on page 1237.
  2. Once you have configured your CSS as an additional layer, you can *select it from the **Styles** drop-down menu (on the toolbar)*.
  3. Run the **DITA Map PDF - WYSIWYG** transformation scenario and the customization rules from the additional CSS will be visible in the produced PDF.

  This method allows you to have many customization CSS files and simply select the one that you need at any time for both the output and rendering **Author** mode while editing.
- Use the `args.css` parameter that allows you to specify a list of CSS URLs to be used in addition to the `dita.css.list` parameter:

  1. Configure a **DITA map to PDF WYSIWYG** transformation scenario, as described in the procedure *above*.
  2. In the **Parameters** tab, specify the path to your custom CSS files in the `args.css` parameter.
  3. Click **OK** and run the transformation scenario.

  This method is appropriate if you just want to apply the styling customization to the output.

Using the CSS Inspector to See Style Associated with an Element

To find the styles associated with an element, follow this procedure:

1. Open a document in the editor and select **Inspect Styles** from the contextual menu. This opens the ***CSS Inspector** view* that shows all the CSS rules that apply to the selected element.
2. Click the particular link for the CSS selector that you need to change and Oxygen XML Editor will open the CSS file and position the cursor at that selector.
3. After you identify the source of the styles you want to modify, copy the CSS rule in your customization CSS file and modify it according to your needs.

   To see the changes in **Author** mode, press **F5** to reload the document.

How to Make Remote Resources Appear in the Output When Using the Prince Processor

If your documentation references external resources (such as images that are stored on a Web-based repository) and your system is behind an HTTP(S) proxy, you may find that they do not appear in the output when using the **Prince Print with CSS** processor. To solve this, follow this procedure:

1. Edit the **`build.xml`** file that is located in *`DITA-OT-DIR`*`/plugins/com.oxygenxml.pdf.css`.

2. There are two instances in the file where a pair of arguments are commented out:

```
<!-- Please remove the comment wrapping the following two arguments
if you are behind a proxy and your documentation refers remote resources,
for example images.

Note: You also need to remove the backslash '\' that appears before the
property name: "http-proxy". That backslash is there because a sequence of
two dashes ('-') is not permited inside a comment.
-->
<!--
  <arg value="-\-http-proxy=${http.proxyHost}:${http.proxyPort}"/>
  <arg value="-\-http-proxy=${https.proxyHost}:${https.proxyPort}"/>
-->
```

3. Remove the comment in both instances.
4. Make sure you also remove the slash that appears before the property name `http-proxy` in each instance.

   Step Result: The arguments should now look like this:

```
<arg value="--http-proxy=${http.proxyHost}:${http.proxyPort}"/>
<arg value="--http-proxy=${https.proxyHost}:${https.proxyPort}"/>
```

5. Save the **`build.xml`** file.
6. Run the *DITA map to PDF WYSIWYG transformation scenario*.

   Result: Your external resources should now appear in your output.

Related Information:

Editing a Transformation Scenario on page 925
Configure Transformation Scenario(s) Dialog Box on page 926
Configuring and Managing Multiple CSS Styles on page 1256
CSS Inspector View on page 232

DITA Map to MS Office Word Transformation

Oxygen XML Editor comes bundled with a DITA OT plugin that allows you to convert *DITA maps* to Microsoft Office Word documents.

Creating the Transformation Scenario

To create an **DITA Map to MS Office Word** transformation scenario, follow these steps:

1. Click the **Configure Transformation Scenario(s)** button from the ***DITA Maps Manager*** toolbar.
2. Select **DITA Map MS Office Word**.
3. In the **Parameters** tab, configure the following parameters:
   - `css.processor.path.prince` (if you are using the **Prince Print with CSS** processor) - Specifies the path to the Prince executable file that will be run to produce the PDF. If you installed Prince using its default settings, you can leave this blank.
   - `css.processor.path.antenna-house` (if you are using the **Antenna House Formatter** processor) - Specifies the path to the Antenna House executable file that will be run to produce the PDF. If you installed Antenna House using its default settings, you can leave this blank.
   - `show.changes.and.comments` - When set to `yes`, user comments, replies to comments, and *tracked changes* are published in the PDF output. The default value is no.
   - `dita.css.list` - Allows you to specify a list of CSS URLs to be used by the PDF processor. The files must have URL syntax and be separated using semicolons. If the value is empty, the current selection from the **Styles** drop-down menu is used.
   - `args.css` - Allows you to specify a list of CSS URLs to be used in addition to those specified in the `dita.css.list` parameter OR in addition to the CSS that is currently selected in the **Styles** drop-down menu. The files must have URL syntax and be separated using semicolons. Also, the `dita.css.list` parameter must be left empty to use these files in addition to the selection in the **Styles** drop-down menu.
4. Click **OK** and run the transformation scenario.

Customizing the Styles (for Output and Editing)

If you need to change the styles in the associated CSS, make sure you install Oxygen XML Editor in a folder where you have full read and write privileges (for instance, your user home directory). This is due to the fact that the installed files are usually placed in a read-only folder (for instance, in Windows, Oxygen XML Editor is installed in the `Program Files` folder where the users do not have change rights).

To change the styles of an element you need to create an additional CSS file that will store the customization rules. Once you have created this file, you need to instruct the editor how to use this additional CSS. This can be done in two ways:

- Create an alternate CSS for the DITA document type:
  1. Follow the procedure for adding an alternate CSS file in *Customizing the Main CSS of a Framework* on page 1237.
  2. Once you have configured your CSS as an additional layer, you can *select it from the **Styles** drop-down menu (on the toolbar)*.
  3. Run the **DITA Map PDF - WYSIWYG** transformation scenario and the customization rules from the additional CSS will be visible in the produced PDF.

  This method allows you to have many customization CSS files and simply select the one that you need at any time for both the output and rendering **Author** mode while editing.
- Use the `args.css` parameter that allows you to specify a list of CSS URLs to be used in addition to the `dita.css.list` parameter:
  1. Configure a **DITA map to PDF WYSIWYG** transformation scenario, as described in the procedure above.
  2. In the **Parameters** tab, specify the path to your custom CSS files in the `args.css` parameter.
  3. Click **OK** and run the transformation scenario.

  This method is appropriate if you just want to apply the styling customization to the output.

Using the CSS Inspector to See Style Associated with an Element

To find the styles associated with an element, follow this procedure:

1. Open a document in the editor and select **Inspect Styles** from the contextual menu. This opens the ***CSS Inspector** view* that shows all the CSS rules that apply to the selected element.
2. Click the particular link for the CSS selector that you need to change and Oxygen XML Editor will open the CSS file and position the cursor at that selector.
3. After you identify the source of the styles you want to modify, copy the CSS rule in your customization CSS file and modify it according to your needs.

   To see the changes in **Author** mode, press **F5** to reload the document.

How to Make Remote Resources Appear in the Output When Using the Prince Processor

If your documentation references external resources (such as images that are stored on a Web-based repository) and your system is behind an HTTP(S) proxy, you may find that they do not appear in the output when using the **Prince Print with CSS** processor. To solve this, follow this procedure:

1. Edit the **`build.xml`** file that is located in *`DITA-OT-DIR`*`/plugins/com.oxygenxml.pdf.css`.
2. There are two instances in the file where a pair of arguments are commented out:

```
<!-- Please remove the comment wrapping the following two arguments
if you are behind a proxy and your documentation refers remote resources,
for example images.

Note: You also need to remove the backslash '\' that appears before the
property name: "http-proxy". That backslash is there because a sequence of
two dashes ('-') is not permited inside a comment.
-->
<!--
  <arg value="-\-http-proxy=${http.proxyHost}:${http.proxyPort}"/>
  <arg value="-\-http-proxy=${https.proxyHost}:${https.proxyPort}"/>
-->
```

3. Remove the comment in both instances.
4. Make sure you also remove the slash that appears before the property name `http-proxy` in each instance.

Step Result: The arguments should now look like this:

```
<arg value="--http-proxy=${http.proxyHost}:${http.proxyPort}"/>
<arg value="--http-proxy=${https.proxyHost}:${https.proxyPort}"/>
```

5. Save the **`build.xml`** file.
6. Run the *DITA map to PDF WYSIWYG* transformation scenario.

   Result: Your external resources should now appear in your output.

Related Information:

Editing a Transformation Scenario on page 925
Configure Transformation Scenario(s) Dialog Box on page 926
Migrating MS Office Documents to DITA on page 1718

Compiled HTML Help (CHM) Output Format

To perform a *Compiled HTML Help (CHM)* transformation, Oxygen XML Editor needs `Microsoft HTML Help Workshop` to be installed on your computer. Oxygen XML Editor automatically detects `HTML Help Workshop` and uses it.

Note: `HTML Help Workshop` might fail if the files used for transformation contain accents in their names, due to different encodings used when writing the *.hhp* and *.hhc* files. If the transformation fails to produce the CHM output but the *.hhp* (HTML Help Project) file is already generated, you can manually try to build the CHM output using `HTML Help Workshop`.

Changing the Output Encoding

Oxygen XML Editor uses the `htmlhelp.locale` parameter to correctly display specific characters of different languages in the output of the *Compiled HTML Help (CHM)* transformation. By default, the **DITA Map CHM** transformation scenario that comes bundled with Oxygen XML Editor has the `htmlhelp.locale` parameter set to `en-US`.

To customize this parameter, follow this procedure:

1. Use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu.
2. Select the **DITA Map CHM** transformation scenario and click the **Edit** button.
3. In the **Parameter** tab, search for the `htmlhelp.locale` parameter and change its value to the desired language tag.

   Note: The format of the `htmlhelp.locale` parameter is `LL-CC`, where `LL` represents the language code (en, for example) and `CC` represents the country code (US, for example). The language codes are contained in the `ISO 639-1` standard and the country codes are contained in the `ISO 3166-1` standard. For further details about language tags, go to *http://www.rfc-editor.org/rfc/rfc5646.txt*.

Kindle Output Format

Oxygen XML Editor requires KindleGento generate Kindle output from *DITA maps*. To install KindleGen for use by Oxygen XML Editor, follow these steps:

1. Go to *www.amazon.com/kindleformat/kindlegen* and download the zip file that matches your operating system.
2. Unzip the file on your local disk.
3. Start Oxygen XML Editor and open a *DITA map* in the ***DITA Maps Manager*** *view*.
4. Click the **Configure Transformation Scenario(s)** button.
5. Select the **DITA Map Kindle** transformation and press the **Edit** button to edit it.
6. Go to **Parameters** tab and set the **kindlegen.executable** parameter as the path to the *KindleGen* directory.
7. Accept the changes.

XHTML Document Type (Framework)

The Extensible HyperText Markup Language (XHTML), is a markup language that has the same depth of expression as HTML, but also conforms to XML syntax.

File Definition

A file is considered to be an XHTML document when the root element name is `html`.

Default Document Templates

There are a variety of default *XHTML* templates available when creating *new documents from templates* and they can be found in: **Framework Templates** > **XHTML**.

The default templates for XHTML documents are located in the *[OXYGEN_INSTALL_DIR]*`/frameworks/xhtml/templates/` folder.

Default Schema for Validation and Content Completion

Default schemas that are used if one is not detected in the XHTML file are stored in the following locations:

- XHTML 1.0 - *[OXYGEN_INSTALL_DIR]*`/frameworks/xhtml/dtd/` or *[OXYGEN_INSTALL_DIR]*`/frameworks/xhtml/nvdl/`.
- XHTML 1.1 - *[OXYGEN_INSTALL_DIR]*`/frameworks/xhtml11/dtd/` or *[OXYGEN_INSTALL_DIR]*`/frameworks/xhtml11/schema/`.
- XHTML 5 - *[OXYGEN_INSTALL_DIR]*`/frameworks/xhtml/xhtml5 (epub3)/`.

Default CSS

The default CSS files used for rendering XHTML content in **Author** mode are stored in *[OXYGEN_INSTALL_DIR]*`/frameworks/xhtml/css/`.

By default, these default CSS files are merged with any that are specified in the document. For more information, see *Configuring and Managing Multiple CSS Styles* on page 1256.

Default XML Catalogs

The default *XML Catalogs* for the XHTML document type are as follows:

- *[OXYGEN_INSTALL_DIR]*`/frameworks/xhtml/dtd/xhtmlcatalog.xml`
- *[OXYGEN_INSTALL_DIR]*`/frameworks/relaxng/catalog.xml`
- *[OXYGEN_INSTALL_DIR]*`/frameworks/nvdl/catalog.xml`
- *[OXYGEN_INSTALL_DIR]*`/frameworks/xhtml11/dtd/xhtmlcatalog.xml`
- *[OXYGEN_INSTALL_DIR]*`/frameworks/xhtml11/schema/xhtmlcatalog.xml`
- *[OXYGEN_INSTALL_DIR]*`/xhtml5 (epub3)/catalog-compat.xml`

Transformation Scenarios

Oxygen XML Editor includes built-in transformation scenarios that allow you to transform XHTML documents to several types of DITA document types (topic, task, concept, reference). They can be found in the **XHTML** section in the ***Configure Transformation Scenario(s)*** *dialog box*.

Resources

- *XHTML Specifications*

Related Information:

Editing XHTML Documents on page 706
Editing XML Documents in Text Mode on page 290
Editing XML Documents in Author Mode on page 329
Adding Tables in XHTML Documents on page 404

XHTML Author Mode Actions

A variety of actions are available in the XHTML *framework* that can be added to the **XHTML** menu, the **Author Custom Actions** toolbar, the contextual menu, and the *Content Completion Assistant*.

XHTML Toolbar Actions

The following default actions are readily available on the **XHTML (Author Custom Actions)** toolbar when editing in **Author** mode (by default, they are also available in the **XHTML** menu and some of them are in various submenus of the contextual menu):

Bold

Changes the style of the selected text to `bold` by surrounding it with b tag. You can use this action on multiple non-contiguous selections.

Italic

Changes the style of the selected text to `italic` by surrounding it with `i` tag. You can use this action on multiple non-contiguous selections.

Underline

Changes the style of the selected text to `underline` by surrounding it with u tag. You can use this action on multiple non-contiguous selections.

Link

Inserts an a element with an `href` attribute at the cursor position. You can type the URL of the reference you want to insert or use the **Browse** drop-down menu to select it using one of the following options:

- **Browse for local file** - Displays the **Open** dialog box to select a local file.
- **Browse for remote file** - Displays the ***Open URL*** *dialog box* to select a remote file.
- **Browse for archived file** - Opens the ***Archive Browser*** to select a file from an archive.
- **Browse Data Source Explorer** - Open the ***Data Source Explorer*** to select a file from a connected data source.
- **Search for file** - Opens the ***Find Resource*** *dialog box* to search for a file.

Insert Image

Inserts a graphic object at the cursor position. This is done by inserting an `img` element regardless of the current context. The following graphical formats are supported: GIF, JPG, JPEG, BMP, PNG, SVG.

Insert Media Resource

Opens a ***Choose Media*** *dialog box* that allows you to select the URL of a media object to be inserted into a document at the cursor position. The result will be that a reference to the specified video, audio, or embedded HTML frame is inserted and rendered in **Author** mode so that it can be played directly from there.

Headings Drop-down Menu

A drop-down menu that includes actions for inserting h1, h2, h3, h4, h5, h6 elements.

Insert Paragraph

Insert a new paragraph element at current cursor position.

Insert Equation

Opens the **XML Fragment Editor** that allows you to insert and *edit MathML notations*.

Insert List Item

Inserts a list item in the current list type.

Insert Unordered List

Inserts an unordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an unordered list.

Insert Ordered List

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

Insert a definition list at the cursor position

Inserts a definition list (`dl` element) with one list item (a `dt` child element and a `dd` child element). You can also use this action to convert selected paragraphs or other types of lists to a definition list.

Sort

Sorts cells or list items in a table.

Insert Table

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content — text + markup — that is rendered inside a *block element*) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

Note: If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables**.

Insert Row Below

Inserts a new table row with empty cells below the current row. This action is available when the cursor is positioned inside a table.

Insert Row Above

Inserts a new table row with empty cells above the current row. This action is available when the cursor is positioned inside a table.

Insert Column After

Inserts a new table column with empty cells after the current column. This action is available when the cursor is positioned inside a table.

Insert Cell

Inserts a new empty cell depending on the current context. If the cursor is positioned between two cells, Oxygen XML Editor a new cell at cursor position. If the cursor is inside a cell, the new cell is created after the current cell.

Delete Column(s)

Deletes the table column located at cursor position or multiple columns in a selection.

Delete Row(s)

Deletes the table row located at cursor position or multiple rows in a selection.

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

XHTML Menu Actions

In addition, the following default actions are available in the **XHTML** menu when editing in **Author** mode (some of them are also available in the contextual menu):

Table submenu

In addition to the table actions available on the toolbar, the following actions are available in this submenu:

Insert Rows

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (**Above** or **Below** the current row).

Insert Column Before

Inserts a column before the current one.

Insert Columns

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (**Above** or **Below** the current column).

Refresh References

You can use this action to manually trigger a refresh and update of all referenced resources.

Tags display mode Submenu

Full Tags with Attributes

Displays full tag names with attributes for both *block* and *inline elements*.

Full Tags

Displays full tag names without attributes for both *block* and *inline elements*.

Block Tags

Displays full tag names for *block elements* and simple tags without names for *inline elements*.

Inline Tags

Displays full tag names for *inline elements*, while *block elements* are not displayed.

Partial Tags

Displays simple tags without names for *inline elements*, while *block elements* are not displayed.

No Tags

No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Configure Tags Display Mode

Use this option to go to the ***Author*** *preferences page* where you can configure the **Tags Display Mode** options.

Profiling/Conditional Text Submenu

Edit Profiling Attributes

Allows you to configure the *profiling attributes* and their values.

Show Profiling Colors and Styles

Select this option to turn on conditional styling.

Show Profiling Attributes

Select this option to turn on conditional text markers. They are displayed at the end of conditional text blocks, as a list of attribute name and their currently set values.

Show Excluded Content

When this option is selected, the content filtered by the currently applied condition set is grayed-out. To show only the content that matches the currently applied condition set, deselect this option.

List of all profiling condition sets that match the current document type

You can click a listed *condition set* to activate it.

Profiling Settings

Opens the *profiling options preferences page*, where you can manage profiling attributes and profiling conditions sets. You can also configure the profiling styles and colors options from the *colors/styles preferences page* and the *attributes rendering preferences page*.

XHTML Contextual Menu Actions

In addition to many of the *XHTML toolbar actions* and the *general **Author** mode contextual menu actions*, the following XHTML *framework*-specific actions are also available in the contextual menu when editing in **Author** mode:

Image Map Editor

This action is available in the contextual menu when it is invoked on an image. This action applies an *image map* to the current image (if one does not already exist) and opens the **Image Map Editor** dialog box. This feature allows you to create hyperlinks in specific areas of an image that will link to various destinations.

Table Actions

The following table editing actions are available in the contextual menu when it is invoked on a table:

Insert Rows

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (**Above** or **Below** the current row).

Delete Row(s)

Deletes the table row located at cursor position or multiple rows in a selection.

Insert Columns

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (**Above** or **Below** the current column).

Delete Column(s)

Deletes the table column located at cursor position or multiple columns in a selection.

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

Sort

Sorts cells or list items in a table.

Table Properties

Opens the **Table properties** dialog box that allows you to configure properties of a table (such as frame borders).

Other Actions submenu

This submenu give you access to all the usual contextual menu actions.

XHTML Drag/Drop (or Copy/Paste) Actions

Dragging a file from the ***Project** view* or ***DITA Maps Manager** view* and dropping it into an XHTML document that is edited in **Author** mode, creates a link to the dragged file (the a element with the `href` attribute) at the drop location. Copy and paste actions work the same.

You can also drag images or media files from your system explorer or the ***Project** view* and drop them into an XHTML document (or copy and paste). This will insert the appropriate element at the drop or paste location (for example, dropping/pasting an image will insert the `img` element with the `src` attribute).

TEI P5 Document Type (Framework)

The *TEI (Text Encoding Initiative)* document type is an international and interdisciplinary standard that enables libraries, museums, publishers, and individual scholars to represent a variety of literary and linguistic texts for online research, teaching, and preservation.

File Definition

A file is considered to be a TEI P5 document when one of the following conditions are true:

- The document namespace is *http://www.tei-c.org/ns/1.0*.
- The public ID of the document is *-//TEI P5*.

Default Document Templates

There are a variety of default *TEI P5* templates available when creating *new documents from templates* and they can be found in: **Framework Templates** > **TEI P5**.

The default templates for TEI P5 documents are located in the `[OXYGEN_INSTALL_DIR]/frameworks/tei/templates/TEI P5` folder.

Default Schema for Validation and Content Completion

The default schema that is used if one is not detected in the TEI P5 document is `tei_all.rng` and it is stored in `[OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/custom/schema/relaxng/`.

Default CSS

The default CSS files used for rendering TEI content in **Author** mode are stored in `[OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/css/`.

By default, these default CSS files are merged with any that are specified in the document. For more information, see *Configuring and Managing Multiple CSS Styles* on page 1256.

Default XML Catalogs

The default *XML Catalogs* for the TEI P5 document type are as follows:

- `[OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/schema/dtd/catalog.xml`
- `[OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/custom/schema/dtd/catalog.xml`
- `[OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/stylesheet/catalog.xml`

Transformation Scenarios

Oxygen XML Editor includes built-in transformation scenarios that allow you to transform TEI P5 documents to a variety of outputs, such as PDF, XHTML, EPUB, DOCX, and ODT. They can be found in the **TEI P5** section in the ***Configure Transformation Scenario(s)*** *dialog box*.

Resources

- *Oxygen Video Tutorial: Editing TEI Documents in Author Mode*
- *TEI: P5 Guidelines*

Related Information:

Editing XML Documents in Author Mode on page 329
Editing XML Documents in Text Mode on page 290
Adding Tables in TEI Documents on page 406

TEI P5 Author Mode Actions

A variety of actions are available in the TEI P5 *framework* that can be added to the **TEI P5** menu, the **Author Custom Actions** toolbar, the contextual menu, and the *Content Completion Assistant*.

TEI P5 Toolbar Actions

The following default actions are readily available on the **TEI P5 (Author Custom Actions)** toolbar when editing in **Author** mode (by default, they are also available in the **TEI P5** menu and some of them are in various submenus of the contextual menu):

Bold

Changes the style of the selected text to `bold` by surrounding it with `hi` tag and setting the `rend` attribute to `bold`. You can use this action on multiple non-contiguous selections.

Italic

Changes the style of the selected text to `italic` by surrounding it with `hi` tag and setting the `rend` attribute to `italic`. You can use this action on multiple non-contiguous selections.

Underline

Changes the style of the selected text to `underline` by surrounding it with `hi` tag and setting the `rend` attribute to `ul`. You can use this action on multiple non-contiguous selections.

Insert Section

Inserts a new section or subsection, depending on the current context. For example, if the current context is `div1`, then a `div2` is inserted. By default, this action also inserts a paragraph element as a child node.

Insert Paragraph

Insert a new paragraph element at current cursor position.

Insert Image

Inserts an image reference at the cursor position. Depending on the current location, an image-type element is inserted.

Insert List Item

Inserts a list item in the current list type.

Insert Ordered List

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

Insert Itemized List

Inserts an itemized list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an itemized list.

Sort

Sorts cells or list items in a table.

Insert Table

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content — text + markup — that is rendered inside a *block element*) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

Note: If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables**.

Insert Row Below

Inserts a new table row with empty cells below the current row. This action is available when the cursor is positioned inside a table.

Insert Column After

Inserts a new table column with empty cells after the current column. This action is available when the cursor is positioned inside a table.

Insert Cell

Inserts a new empty cell depending on the current context. If the cursor is positioned between two cells, Oxygen XML Editor a new cell at cursor position. If the cursor is inside a cell, the new cell is created after the current cell.

Delete Column(s)

Deletes the table column located at cursor position or multiple columns in a selection.

Delete Row(s)

Deletes the table row located at cursor position or multiple rows in a selection.

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

TEI P5 Menu Actions

In addition, the following default actions are available in the **TEI P5** menu when editing in **Author** mode (some of them are also available in the contextual menu):

Table submenu

In addition to the table actions available on the toolbar, the following actions are available in this submenu:

Insert Row Above

Inserts a new table row with empty cells above the current row. This action is available when the cursor is positioned inside a table.

Insert Rows

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (**Above** or **Below** the current row).

Insert Column Before

Inserts a column before the current one.

Insert Columns

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (**Above** or **Below** the current column).

ID Options

Opens the **ID Options** dialog box that allows you to configure options for generating IDs in **Author** mode. The dialog box includes the following:

ID Pattern

The pattern for the ID values that will be generated. This text field can be customized using constant strings or any of the Oxygen XML Editor *Editor Variables*.

Element name or class value to generate ID for

The elements for which ID values will be generated, specified using class attribute values. To customize the list, use the **Add**, **Edit**, or **Remove** buttons.

Auto generate IDs for elements

If selected, Oxygen XML Editor will automatically generate unique IDs for the elements listed in this dialog box when they are created in **Author** mode.

Remove IDs when copying content in the same document

When copying and pasting content in the same document, this option allows you to control whether or not pasted elements that are listed in this dialog box should retain their existing IDs. To retain the element IDs, deselect this option.

Note: This option does not have an effect on content that is *cut* and *pasted*.

Generate IDs

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements from the list in the **ID Options** dialog box that are found in the current selection.

Note: The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `id` attribute.

Refresh References

You can use this action to manually trigger a refresh and update of all referenced resources.

Tags display mode Submenu

Full Tags with Attributes

Displays full tag names with attributes for both *block* and *inline elements*.

Full Tags

Displays full tag names without attributes for both *block* and *inline elements*.

Block Tags

Displays full tag names for *block elements* and simple tags without names for *inline elements*.

Inline Tags

Displays full tag names for *inline elements*, while *block elements* are not displayed.

Partial Tags

Displays simple tags without names for *inline elements*, while *block elements* are not displayed.

No Tags

No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Configure Tags Display Mode

Use this option to go to the ***Author*** *preferences page* where you can configure the **Tags Display Mode** options.

Profiling/Conditional Text Submenu

Edit Profiling Attributes

Allows you to configure the *profiling attributes* and their values.

Show Profiling Colors and Styles

Select this option to turn on conditional styling.

Show Profiling Attributes

Select this option to turn on conditional text markers. They are displayed at the end of conditional text blocks, as a list of attribute name and their currently set values.

Show Excluded Content

When this option is selected, the content filtered by the currently applied condition set is grayed-out. To show only the content that matches the currently applied condition set, deselect this option.

List of all profiling condition sets that match the current document type

You can click a listed *condition set* to activate it.

Profiling Settings

Opens the *profiling options preferences page*, where you can manage profiling attributes and profiling conditions sets. You can also configure the profiling styles and colors options from the *colors/styles preferences page* and the *attributes rendering preferences page*.

TEI Contextual Menu Actions

In addition to many of the *TEI toolbar actions* and the *general* ***Author*** *mode contextual menu actions*, the following TEI *framework*-specific actions are also available in the contextual menu when editing in **Author** mode:

Image Map Editor

This action is available in the contextual menu when it is invoked on an image. This action applies an *image map* to the current image (if one does not already exist) and opens the **Image Map Editor** dialog box. This feature allows you to create hyperlinks in specific areas of an image that will link to various destinations.

Table Actions

The following table editing actions are available in the contextual menu when it is invoked on a table:

Insert Rows

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (**Above** or **Below** the current row).

Delete Row(s)

Deletes the table row located at cursor position or multiple rows in a selection.

Insert Columns

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (**Above** or **Below** the current column).

Delete Column(s)

Deletes the table column located at cursor position or multiple columns in a selection.

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

Sort

Sorts cells or list items in a table.

Table Properties

Opens the **Table properties** dialog box that allows you to configure properties of a table (such as frame borders).

Other Actions submenu

This submenu give you access to all the usual contextual menu actions.

TEI P5 Drag/Drop Actions

Dragging a file from the ***Project*** *view* or ***DITA Maps Manager*** *view* and dropping it into a TEI P5 document that is edited in **Author** mode, creates a link to the dragged file (the `ptr` element with the `target` attribute) at the drop location. Dragging an image file from the default file system application (Windows Explorer on Windows or Finder on Mac OS X, for example) and dropping it into a TEI P5 document inserts a graphic element (the `graphic` element with the `url` attribute) at the drop location, similar to the **Insert Image** toolbar action.

Customization of TEI Frameworks Using the Latest Sources

The *TEI P5 framework* is available as a public project at the following SVN repository: *https://github.com/TEIC/oxygen-tei*. This project is the base for customizing a TEI *framework*.

To customize a TEI *framework*, follow this procedure:

1. Check out the project on a local computer from the SVN repository.

   This action is done with an SVN client application that creates a working copy of the SVN repository on a local computer.
2. Customize the TEI *framework* in Oxygen XML Editor.
   a) Set the Oxygen XML Editor `frameworks` folder to the `oxygen/frameworks` subfolder of the folder of the SVN working copy.

*Open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Global**, and set the path of the SVN working copy in the **Use custom frameworks** option.

b) *Open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association** > **Locations**, and select **Custom**.

3. Build a *JAR* file with the TEI *framework*.

   The SVN project includes a `build.xml` file that can be used for building a *JAR* file using the Ant tool. The command that should be used:

```
ant -f build.xml
```

4. Distribute the *JAR* file to the users that need the customized TEI *framework*.

   The command from the above step creates a file `tei.zip` in the `dist` subfolder of the SVN project. Each user that needs the customized TEI *framework* will receive the `tei.zip` file and will unzip it in the `frameworks` folder of the Oxygen XML Editor install folder.

Customization of TEI Frameworks Using the Compiled Sources

The following procedure describes how to update to the latest stable version of TEI Schema and TEI XSL, already integrated in the TEI *framework* for Oxygen XML Editor.

1. Go to *https://code.google.com/p/oxygen-tei/*;
2. Go to **Downloads**;
3. Download the latest uploaded `.zip` file;
4. Unpack the `.zip` file and copy its content in the Oxygen XML Editor `frameworks` folder.

TEI ODD Document Type (Framework)

The *TEI ODD (Text Encoding Initiative - One Document Does it all)* document type is a TEI XML-conformant specification format that allows you to create a custom TEI P5 schema in a literate programming fashion. A system of XSLT stylesheets called *Roma* was created by the TEI Consortium for manipulating the ODD files.

File Definition

A file is considered to be a TEI ODD document when the following conditions are true:

- The file extension is `.odd`.
- The document namespace is *http://www.tei-c.org/ns/1.0*.

Default Document Templates

There is a default *TEI ODD* document template available when creating *new documents from templates* and they can be found in: **Framework Templates** > **TEI ODD**.

The default template is located in the *`[OXYGEN_INSTALL_DIR]`*`/frameworks/tei/templates/TEI ODD` folder.

Default Schema for Validation and Content Completion

The default schema that is used if one is not detected in the TEI ODD document is `tei_odds.rng` and it is stored in *`[OXYGEN_INSTALL_DIR]`*`/frameworks/tei/xml/tei/custom/schema/relaxng/`.

Default CSS

The default CSS files used for rendering TEI ODD content in **Author** mode are stored in *`[OXYGEN_INSTALL_DIR]`*`/frameworks/tei/xml/tei/css/`.

By default, these default CSS files are merged with any that are specified in the document. For more information, see *Configuring and Managing Multiple CSS Styles* on page 1256.

Default XML Catalogs

The default *XML Catalogs* for the TEI ODD document type are as follows:

- `[OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/custom/schema/catalog.xml`
- `[OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/schema/catalog.xml`

Transformation Scenarios

Oxygen XML Editor includes built-in transformation scenarios that allow you to transform TEI ODD documents to a variety of outputs, such as PDF, XHTML, EPUB, DOCX, ODT, RNG, DTD, and XML Schema. They can be found in the **TEI ODD** section in the ***Configure Transformation Scenario(s)*** *dialog box*.

Resources

- *Oxygen Video Tutorial: Editing TEI Documents in Author Mode*
- *TEI: Getting Started with ODD*

Related Information:

Editing XML Documents in Author Mode on page 329
Editing XML Documents in Text Mode on page 290

TEI ODD Author Mode Actions

A variety of actions are available in the TEI ODD *framework* that can be added to the **TEI ODD** menu, the **Author Custom Actions** toolbar, the contextual menu, and the *Content Completion Assistant*.

TEI ODD Toolbar Actions

The following default actions are readily available on the **TEI ODD (Author Custom Actions)** toolbar when editing in **Author** mode (by default, they are also available in the **TEI ODD** menu and some of them are in various submenus of the contextual menu):

Bold

Changes the style of the selected text to `bold` by surrounding it with `hi` tag and setting the `rend` attribute to `bold`. You can use this action on multiple non-contiguous selections.

Italic

Changes the style of the selected text to `italic` by surrounding it with `hi` tag and setting the `rend` attribute to `italic`. You can use this action on multiple non-contiguous selections.

Underline

Changes the style of the selected text to `underline` by surrounding it with `hi` tag and setting the `rend` attribute to `ul`. You can use this action on multiple non-contiguous selections.

Insert Section

Inserts a new section or subsection, depending on the current context. For example, if the current context is `div1`, then a `div2` is inserted. By default, this action also inserts a paragraph element as a child node.

Insert Paragraph

Insert a new paragraph element at current cursor position.

Insert Image

Inserts an image reference at the cursor position. Depending on the current location, an image-type element is inserted.

Insert List Item

Inserts a list item in the current list type.

Insert Ordered List

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

Insert Itemized List

Inserts an itemized list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an itemized list.

Insert Table

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content — text + markup — that is rendered inside a *block element*) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

Note: If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables**.

Insert Row Below

Inserts a new table row with empty cells below the current row. This action is available when the cursor is positioned inside a table.

Insert Column After

Inserts a new table column with empty cells after the current column. This action is available when the cursor is positioned inside a table.

Insert Cell

Inserts a new empty cell depending on the current context. If the cursor is positioned between two cells, Oxygen XML Editor a new cell at cursor position. If the cursor is inside a cell, the new cell is created after the current cell.

Delete Column(s)

Deletes the table column located at cursor position or multiple columns in a selection.

Delete Row(s)

Deletes the table row located at cursor position or multiple rows in a selection.

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

TEI ODD Menu Actions

In addition, the following default actions are available in the **TEI ODD** menu when editing in **Author** mode (some of them are also available in the contextual menu):

Table submenu

In addition to the table actions available on the toolbar, the following actions are available in this submenu:

Insert Row Above

Inserts a new table row with empty cells above the current row. This action is available when the cursor is positioned inside a table.

Insert Rows

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (**Above** or **Below** the current row).

Insert Column Before

Inserts a column before the current one.

Insert Columns

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (**Above** or **Below** the current column).

ID Options

Opens the **ID Options** dialog box that allows you to configure options for generating IDs in **Author** mode. The dialog box includes the following:

ID Pattern

The pattern for the ID values that will be generated. This text field can be customized using constant strings or any of the Oxygen XML Editor *Editor Variables*.

Element name or class value to generate ID for

The elements for which ID values will be generated, specified using class attribute values. To customize the list, use the **Add**, **Edit**, or **Remove** buttons.

Auto generate IDs for elements

If selected, Oxygen XML Editor will automatically generate unique IDs for the elements listed in this dialog box when they are created in **Author** mode.

Remove IDs when copying content in the same document

When copying and pasting content in the same document, this option allows you to control whether or not pasted elements that are listed in this dialog box should retain their existing IDs. To retain the element IDs, deselect this option.

Note: This option does not have an effect on content that is *cut* and *pasted*.

Generate IDs

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements from the list in the **ID Options** dialog box that are found in the current selection.

Note: The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `id` attribute.

Refresh References

You can use this action to manually trigger a refresh and update of all referenced resources.

Tags display mode Submenu

Full Tags with Attributes

Displays full tag names with attributes for both *block* and *inline elements*.

Full Tags

Displays full tag names without attributes for both *block* and *inline elements*.

Block Tags

Displays full tag names for *block elements* and simple tags without names for *inline elements*.

Inline Tags

Displays full tag names for *inline elements*, while *block elements* are not displayed.

Partial Tags

Displays simple tags without names for *inline elements*, while *block elements* are not displayed.

No Tags

No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Configure Tags Display Mode

Use this option to go to the ***Author*** *preferences page* where you can configure the **Tags Display Mode** options.

Profiling/Conditional Text Submenu

Edit Profiling Attributes

Allows you to configure the *profiling attributes* and their values.

Show Profiling Colors and Styles

Select this option to turn on conditional styling.

Show Profiling Attributes

Select this option to turn on conditional text markers. They are displayed at the end of conditional text blocks, as a list of attribute name and their currently set values.

Show Excluded Content

When this option is selected, the content filtered by the currently applied condition set is grayed-out. To show only the content that matches the currently applied condition set, deselect this option.

List of all profiling condition sets that match the current document type

You can click a listed *condition set* to activate it.

Profiling Settings

Opens the *profiling options preferences page*, where you can manage profiling attributes and profiling conditions sets. You can also configure the profiling styles and colors options from the *colors/styles preferences page* and the *attributes rendering preferences page*.

TEI Contextual Menu Actions

In addition to many of the *TEI toolbar actions* and the *general* ***Author*** *mode contextual menu actions*, the following TEI *framework*-specific actions are also available in the contextual menu when editing in **Author** mode:

Table Actions

The following table editing actions are available in the contextual menu when it is invoked on a tabl

e:

Insert Rows

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (**Above** or **Below** the current row).

Delete Row(s)

Deletes the table row located at cursor position or multiple rows in a selection.

Insert Columns

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (**Above** or **Below** the current column).

Delete Column(s)

Deletes the table column located at cursor position or multiple columns in a selection.

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

Sort

Sorts cells or list items in a table.

Table Properties

Opens the **Table properties** dialog box that allows you to configure properties of a table (such as frame borders).

Other Actions submenu

This submenu give you access to all the usual contextual menu actions.

TEI ODD Drag/Drop Actions

Dragging a file from the ***Project** view* or ***DITA Maps Manager** view* and dropping it into a TEI ODD document that is edited in **Author** mode, creates a link to the dragged file (the `ptr` element with the `target` attribute) at the drop location.

jTEI Document Type (Framework)

The *jTEI (Journal of the Text Encoding Initiative)* document type is highly restrictive customization (only about 80 elements are included) of the TEI P5 *framework*.

File Definition

A file is considered to be a *jTEI* document when the root element is named *TEI*, it is in the namespace *http://www.tei-c.org/ns/1.0*, and the `@rend` attribute is set to *"jTEI"*.

Default Document Templates

There is a default **jTEI Article** template available when creating *new documents from templates* and they can be found in: **Framework Templates** > **TEI JTEI**.

The default template is located in the *[OXYGEN_INSTALL_DIR]*`/frameworks/tei/templates/TEI jTEI` folder.

Default Schema for Validation and Content Completion

The default schema that is used if one is not detected is `tei_jtei.rng` and it is stored in *[OXYGEN_INSTALL_DIR]*`/frameworks/tei/xml/tei/custom/schema/relaxng/`.

Default CSS

The default CSS file (`jtei.css`) that is used for rendering jTEI in **Author** mode is stored in *[OXYGEN_INSTALL_DIR]*`/frameworks/tei/xml/tei/css/`.

By default, these default CSS files are merged with any that are specified in the document. For more information, see *Configuring and Managing Multiple CSS Styles* on page 1256.

Default XML Catalogs

The default *XML Catalogs* for jTEI are as follows:

- *[OXYGEN_INSTALL_DIR]*`/frameworks/tei/xml/tei/schema/dtd/catalog.xml`
- *[OXYGEN_INSTALL_DIR]*`/frameworks/tei/xml/tei/custom/schema/dtd/catalog.xml`
- *[OXYGEN_INSTALL_DIR]*`/frameworks/tei/xml/tei/stylesheet/catalog.xml`

Transformation Scenarios

Oxygen XML Editor includes built-in transformation scenarios that allow you to transform jTEI documents to PDF and ODT. They can be found in the **TEI JTEI** section in the ***Configure Transformation Scenario(s)** dialog box*.

Resources

- *Oxygen Video Tutorial: Editing TEI Documents in Author Mode*
- *jTEI Article Guidelines*

Related Information:

Editing XML Documents in Author Mode on page 329
Editing XML Documents in Text Mode on page 290
Adding Tables in TEI Documents on page 406

JATS Document Type (Framework)

The *JATS (NISO Journal Article Tag Suite)* document type is a technical standard that defines an XML format for scientific literature.

File Definition

A file is considered to be a JATS document when the PUBLIC ID of the document contains the string `-//NLM//DTD`.

Default Document Templates

There are some default *JATS* templates available when creating *new documents from templates* and they can be found in: **Framework Templates** > **JATSKit - NISO JATS and NLM BITS**

The default templates for JATS documents are located in the *`[OXYGEN_INSTALL_DIR]`*`/frameworks/jats/templates/` folder.

Default Schema for Validation and Content Completion

Default schemas that are used if one is not detected in the JATS document are stored in *`[OXYGEN_INSTALL_DIR]`*`/frameworks/jats/lib/schemas/`.

Default CSS

The default CSS files used for rendering JATS content in **Author** mode are stored in *`[OXYGEN_INSTALL_DIR]`*`/frameworks/jats/lib/author-css/`.

By default, these default CSS files are merged with any that are specified in the document. For more information, see *Configuring and Managing Multiple CSS Styles* on page 1256.

Default XML Catalog

The default *XML Catalog*, `jatskit-catalog.xml`, is stored in *`[OXYGEN_INSTALL_DIR]`*`/frameworks/jats/lib/schemas/`.

Transformation Scenarios

Oxygen XML Editor includes built-in transformation scenarios that allow you to transform JATS documents to a variety of outputs, such as PDF, HTML, and EPUB. They can be found in the **JATSKit** section in the ***Configure Transformation Scenario(s)*** *dialog box*.

Resources

- *Oxygen Video Tutorial: Configuring a JATS Framework*
- *NLM Journal Archiving and Interchange Tag Suite*

Related Information:

Editing XML Documents in Author Mode on page 329
Editing XML Documents in Text Mode on page 290

JATS Author Mode Actions

A variety of actions are available in the JATS *framework* that can be added to the **JATS** menu, the **Author Custom Actions** toolbar, the contextual menu, and the *Content Completion Assistant*.

JATS Toolbar Actions

The following default actions are readily available on the **JATS (Author Custom Actions)** toolbar when editing in **Author** mode (by default, they are also available in the **JATS** menu and in various submenus of the contextual menu):

Paragraph Level Drop-Down Menu

Insert Paragraph

Insert a new paragraph element at current cursor position.

Insert Unordered List

Inserts an unordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an unordered list.

Insert Ordered List

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

Boxed Text

Inserts or wraps content in a box with a shaded background.

Code

Inserts or wraps content in a `code` element.

Display Quote

Inserts or wraps content in a `disp-quote` element.

Figure

Inserts a `fig` element with a `title` (inside a `caption` element). This action opens a dialog box that allows you to enter the text for the title for the figure.

Graphic Figure

Inserts a `fig` element with a `title` (inside a `caption` element), and a `graphic` element. A dialog box is displayed that allows you to enter the title for the figure, followed by a dialog box that allows you to select the URL of the graphic to be inserted.

Bold

Surrounds the selected text with a `bold` tag. You can use this action on multiple non-contiguous selections.

Italic

Surrounds the selected text with an `italic` tag. You can use this action on multiple non-contiguous selections.

Underline

Surrounds the selected text with an `underline` tag. You can use this action on multiple non-contiguous selections.

Monospace

Inserts or wraps content with a `monospace` element.

Insert Image

Inserts an image reference at the cursor position. Depending on the current location, an image-type element is inserted.

Insert List Item

Inserts a list item in the current list type.

Insert MathML

Opens the **XML Fragment Editor** that allows you to insert and *edit MathML notations*.

Subscript

Surrounds the selected text with a `sub` tag. The text will appear lower than the baseline and slightly smaller.

Superscript

Surrounds the selected text with a sup tag. The text will appear higher than the baseline and slightly smaller.

JATS Menu Actions

In addition, the following default actions are available in the **JATS** menu when editing in **Author** mode:

Refresh References

You can use this action to manually trigger a refresh and update of all referenced resources.

Tags display mode Submenu

Full Tags with Attributes

Displays full tag names with attributes for both *block* and *inline elements*.

Full Tags

Displays full tag names without attributes for both *block* and *inline elements*.

Block Tags

Displays full tag names for *block elements* and simple tags without names for *inline elements*.

Inline Tags

Displays full tag names for *inline elements*, while *block elements* are not displayed.

Partial Tags

Displays simple tags without names for *inline elements*, while *block elements* are not displayed.

No Tags

No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Configure Tags Display Mode

Use this option to go to the ***Author*** *preferences page* where you can configure the **Tags Display Mode** options.

Profiling/Conditional Text Submenu

Edit Profiling Attributes

Allows you to configure the *profiling attributes* and their values.

Show Profiling Colors and Styles

Select this option to turn on conditional styling.

Show Profiling Attributes

Select this option to turn on conditional text markers. They are displayed at the end of conditional text blocks, as a list of attribute name and their currently set values.

Show Excluded Content

When this option is selected, the content filtered by the currently applied condition set is grayed-out. To show only the content that matches the currently applied condition set, deselect this option.

List of all profiling condition sets that match the current document type

You can click a listed *condition set* to activate it.

Profiling Settings

Opens the *profiling options preferences page*, where you can manage profiling attributes and profiling conditions sets. You can also configure the profiling styles and colors options from the *colors/styles preferences page* and the *attributes rendering preferences page*.

JATS Drag/Drop Actions

Dragging a file from the ***Project*** *view* or ***DITA Maps Manager*** *view* and dropping it into a JATS document that is edited in **Author** mode, creates a link to the dragged file (the `ext-link` element with the `xlink:href` attribute) at the drop location. Dragging an image file from the default file system application (Windows Explorer on

Windows or Finder on Mac OS X, for example) and dropping it into a JATS document inserts an image element (the `inline-graphic` element with the `xlink:href` attribute) at the drop location, similar to the **Insert Image** toolbar action.

EPUB Document Type (Framework)

EPUB is an e-book file format that is a ZIP archive and can be downloaded and read on devices such as phones, tablets, computers, or e-readers. Oxygen XML Editor includes an ***Archive Browser*** *view* that allows you to view the contents and structure of this type of file.

Three distinct *frameworks* are supported for the EPUB document type:

- **NCX** - A declarative global navigation definition.
- **OCF** - The Open Container Format (OCF) defines a mechanism by which all components of an Open Publication Structure (OPS) can be combined into a single file system entity.
- **OPF** - The Open Packaging Format (OPF) defines the mechanism by which all components of a published work that conforms to the Open Publication Structure (OPS) standard (including metadata, reading order, and navigational information) are packaged in an OPS Publication.

  Note: Oxygen XML Editor supports both OPF 2.0 and OPF 3.0.

File Definition

A file is considered to be an *EPUB* document if it has a file extension of `.epub`.

Default Document Templates

There are a variety of default *EPUB* templates available when creating *new documents from templates* and they can be found the following folders in **Framework Templates**: **NCX**, **OCF**, **OPF 2.0**, and **OPF 3.0**.

The default templates for the *NCX* document types are located in the *`[OXYGEN_INSTALL_DIR]`*`/frameworks/ncx/templates` folder.

The default templates for the *OCF* document types are located in the *`[OXYGEN_INSTALL_DIR]`*`/frameworks/ocf/templates` folder.

The default template for the *OPF 2.0* document type is located in the *`[OXYGEN_INSTALL_DIR]`*`/frameworks/opf/templates/2.0` folder.

The default template for the *OPF 3.0* document type is located in the *`[OXYGEN_INSTALL_DIR]`*`/frameworks/opf/templates/3.0` folder.

Related Information:

Working with Archive Files on page 1065

Other Supported Document Types

Along with the *fully supported predefined frameworks (document types)*, Oxygen XML Editor also provides limited support (including file templates) for editing a variety of other document types. All the specialized views, editors, actions, and options are dynamic according to the type of file that is opened or created. Other document types that are supported in Oxygen XML Editor include:

- *EPUB (NCX, OCF, OPF 2.0 & 3.0)* - A standard for e-book files.
- *OOXML* - An XML-based file format for representing spreadsheets, charts, presentations, and word processing documents.
- *ODF* - An free and open-source XML-based file format for electronic office documents, such as spreadsheets, charts, presentations, and word processing documents.
- *DocBook Targetset* - For resolving cross-references when using *olinks*.
- *Ant Build Scripts* - A tool for automating software build processes, written in Java and primarily intended for use with Java.
- *XSLT Stylesheets* - A document type that provides a visual mode for editing XSLT stylesheets.

- *WSDL* - Web Services Description Language is an XML language for describing the functionality offered by a web service.
- *Schematron* - For making assertions about the presence or absence of patterns in XML documents. This document type applies to the ISO Schematron version.
- *Schematron Quick Fixes (SQF)* - An extension of the ISO standard Schematron, allows developers to define *Quick Fixes* for Schematron errors.
- *StratML (Part 1 & 2)* - Part 1 and 2 of the Strategy Markup Language specification.
- *XProc* - A document type for processing XProc script files.
- *XML Schema* - Documents that provide support for editing annotations.
- *SVG* - Scalable Vector Graphics is a language for describing two-dimensional graphics in XML.
- *MathML* - Mathematical Markup Language (2.0 and 3.0) is an application of XML for describing mathematical notations.
- *XLIFF (1.2 & 2.0)* - XML Localization Interchange File Format is a standard for passing data between tools during a localization process.
- *XQuery* - The common query language for XML.
- *CSS* - Cascading Style Sheets is a language used for describing the look and formatting of a document.
- *LESS* - A dynamic style sheet language that can be compiled into CSS.
- *Relax NG Schema* - A schema language that specifies a pattern for the structure and content of an XML document.
- *NVDL Schema* - Namespace Validation Dispatching Language allows you to specify sections of XML documents to be validated against various schemas.
- *JSON* - JavaScript Object Notation is a lightweight data-interchange format.
- *Markdown* - A lightweight markup language with plain text formatting syntax that can be converted to HTML or DITA.
- *JavaScript* - Programming language of HTML and the Web.
- XMLSpec - A markup language for W3C specifications and other technical reports.
- DITAVAL - DITA conditional processing profile to identify the values you want to conditionally process for a particular output, build, or other purpose.
- Daisy - A technical standard for digital audio books, periodicals, and computerized text. It is designed to be an audio substitute for print material.
- EAD - Encoded Archival Description is an XML standard for encoding archival finding aids.
- KML - Keyhole Markup Language is an XML notation for expressing geographic visualization in maps and browsers.
- Maven Project & Settings - Project or settings file for Maven build automation tool that is primarily used for Java projects.
- Oasis *XML Catalog* - An *XML Catalog* document that describes a mapping between external entity references and locally-cached equivalents.
- *Other Non-XML Files*- Oxygen XML Editor also includes a simple text editor and a variety of helpful features for creating and editing non-XML files.

Transforming Documents

9

Topics:

- *Transformation Scenarios*
- *WebHelp System Output*

XML documents can be transformed into a variety of user-friendly output formats that can be viewed by other users. This process is known as a *transformation*.

Oxygen XML Editor allows you to use transformation scenarios to publish XML content in various output formats (such as WebHelp, PDF, CHM, EPUB, JavaHelp, Eclipse Help, XHTML, etc.)

For transformations that are not included in your installed version of Oxygen XML Editor, simply install the tool chain required to perform the specific transformation and process the files in accordance with the processor instructions. A multitude of target formats are possible. The basic condition for transformation to any format is that your source document is well-formed.

Note: You need to use the appropriate stylesheet according to the source definition and the desired output. For example, if you want to transform into an HTML format using a DocBook stylesheet, your source XML document should conform with the DocBook DTD.

Transformation Scenarios

A transformation scenario is a set of complex operations and settings that gives you the possibility to obtain outputs of multiple types (XML, HTML, PDF, EPUB, etc.) from the same source of XML files and stylesheets.

Executing a transformation scenario implies multiple actions, such as:

- Validating the input file.
- Obtaining intermediate output files (for example, formatting objects for the XML to PDF transformation).
- Using transformation engines to produce the output.

Before transforming an XML document in Oxygen XML Editor, you need to define a transformation scenario to apply to that document. A scenario is a set of values for various parameters that define a transformation. It is not related to a particular document, but rather to a document type. Types of transformation scenarios include:

- **Scenarios that Apply to XML Files** - This type of scenario contains the location of an XSLT stylesheet that is applied on the edited XML document, as well as other transformation parameters.
- **Scenarios that Apply to XSLT Files** - This type of scenario contains the location of an XML document that the edited XSLT stylesheet is applied to, as well as other transform parameters.
- **Scenarios that Apply to XQuery Files** - This type of scenario contains the location of an XML source, that the edited XQuery file is applied to, as well as other transform parameters. When the XML source is a native XML database, the XML source field of the scenario is empty because the XML data is read with XQuery-specific functions, such as `document()`. When the XML source is a local XML file, the URL of the file is specified in the XML input field of the scenario.
- **Scenarios that Apply to SQL Files** - This type of scenario specifies a database connection for the database server that runs the SQL file that is associated with the scenario. The data processed by the SQL script is located in the database.
- **Scenarios that Apply to XProc Files** - This type of scenario contains the location of an XProc script, as well as other transform parameters.
- **DITA-OT Scenarios** - This type of scenario provides the parameters for an Ant transformation that executes a DITA-OT build script. Oxygen XML Editor includes a built-in version of Ant and a built-in version of DITA-OT, although you can also set other versions in the scenario.

- **ANT Scenarios** - This type of scenario contains the location of an Ant build script, as well as other transform parameters.

Note:

Status messages generated during the transformation process are displayed in the ***Information*** *view*.

Built-in Transformation Scenarios

Oxygen XML Editor included preconfigured built-in transformation scenarios that are used for common transformations. They can be found in the various sections in the ***Configure Transformation Scenario(s)*** *dialog box*. All the predefined *document types (frameworks)* that are included in Oxygen XML Editor have various transformation scenarios in their specific sections, including the most popular *frameworks*, such as DITA, DocBook, TEI, XHTML, JATS, OOXML, and more.

To obtain the desired output, use one of the following actions from the toolbar:

- **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on OS X))**
- **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))**

Then choose one of the built-in scenarios for the current document and click the **Apply Associated** button.

Note:

- You can apply a transformation even if the current document is not associated with a transformation scenario.
- If the document contains an `xml-stylesheet` processing instruction that refers to an XSLT stylesheet (commonly used to display the document in web browsers), Oxygen XML Editor prompts you to associate the document with a built-in transformation scenario.
- The default transformation scenario is suggested based on the processing instruction from the edited document.

Related Information:

Creating New Transformation Scenarios on page 886
Editing a Transformation Scenario on page 925
Configure Transformation Scenario(s) Dialog Box on page 926

DocBook 4 Transformation Scenarios

Default transformation scenarios allow you to convert DocBook 4 to DocBook 5 documents and transform DocBook documents to a variety of outputs, such as WebHelp, PDF, HTML, HTML Chunk, XHTML, XHTML Chunk, and EPUB. All of them are listed in the **DocBook 4** section in the ***Configure Transformation Scenario(s)*** *dialog box*.

Related Information:

Editing a Transformation Scenario on page 925
Configure Transformation Scenario(s) Dialog Box on page 926

DocBook 4 to WebHelp Output

DocBook 4 documents can be transformed into several types of WebHelp systems.

WebHelp Classic Output

To publish a DocBook 4 document as a **WebHelp Classic** system, follow these steps:

1. Click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
2. Select the **DocBook WebHelp Classic** scenario from the **DocBook 4** section.
3. Click **Apply associated**.

   When the **DocBook WebHelp Classic** transformation is complete, the output is automatically opened in your default browser.

WebHelp Classic with Feedback Output

To publish a DocBook 4 document as a **WebHelp Classic with Feedback** system, follow these steps:

1. Click **Configure Transformation Scenarios**.
2. Select the **DocBook WebHelp Classic with Feedback** scenario from the **DocBook 4** section.
3. Click **Apply associated**.
4. Enter the documentation product ID and the documentation version.

   When the **DocBook WebHelp Classic with Feedback** transformation is complete, your default browser opens the `installation.html` file. This file contains information about the output location, system requirements, installation instructions, and deployment of the output. Follow the instructions to complete the system deployment. For more information, see *Deploying a Feedback-Enabled WebHelp System*.

To watch our video demonstration about the feedback-enabled WebHelp system, go to *https://www.oxygenxml.com/demo/Feedback_Enabled_WebHelp.html*.

WebHelp Classic Mobile Output

To publish a DocBook 4 document as a **WebHelp Classic Mobile** system, follow these steps:

1. Click **Configure Transformation Scenarios**.
2. Select the **DocBook WebHelp Classic Mobile** scenario from the **DocBook 4** section.
3. Click **Apply associated**.

When the **DocBook WebHelp Classic Mobile** transformation is complete, the output is automatically opened in your default browser.

Customizing WebHelp Transformation Scenarios

To customize a DocBook WebHelp transformation scenario, you can edit various parameters, including the following most commonly used ones:

args.default.language

This parameter is used if the language is not detected in the *DITA map*. The default value is `en-us`.

clean.output

Deletes all files from the output folder before the transformation is performed (only no and `yes` values are valid and the default value is no).

l10n.gentext.default.language

This parameter is used to identify the correct stemmer that differs from language to language. For example, for English the value of this parameter is `en` or for French it is `fr`, and so on.

use.stemming

Controls whether or not you want to include stemming search algorithms into the published output (default setting is `false`).

webhelp.copyright

Adds a small copyright text that appears at the end of the **Table of Contents** pane.

webhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

webhelp.favicon

The file path that points to an image to be used as a *favicon* in the WebHelp output.

webhelp.footer.file

Path to an XML file that includes the footer content for your WebHelp output pages. You can use this parameter to integrate social media features (such as widgets for Facebook™, Twitter™, Google Analytics, or Google+™). The file must be well-formed, each widget must be in separate `div` or `span` element, and the code

for each `script` element is included in an XML comment (also, the start and end tags for the XML comment must be on a separate line). The following code exert is an example for adding a Facebook™ widget:

```
<div id="facebook">
  <div id="fb-root"/>
  <script>
    <!-- (function(d, s, id) { var js, fjs = d.getElementsByTagName(s)[0];
      if (d.getElementById(id)) return;
      js = d.createElement(s); js.id = id;
      js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
      fjs.parentNode.insertBefore(js, fjs); }
       (document, 'script', 'facebook-jssdk')); -->
    </script>
    <div data-share="true" data-show-faces="true" data-action="like"
       data-layout="standard" class="fb-like"/>
</div>
```

webhelp.footer.include

Specifies whether or not to include footer in each WebHelp page. Possible values: `yes`, no. If set to no, no footer is added to the WebHelp pages. If set to `yes` and the `webhelp.footer.file` parameter has a value, then the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default Oxygen XML Editor footer is inserted in each WebHelp page.

webhelp.logo.image.target.url

Specifies a target URL that is set on the logo image. When you click the logo image, you will be redirected to this address.

webhelp.logo.image

Specifies a path to an image displayed as a logo in the left side of the output header.

webhelp.product.id (available only for Feedback-enabled systems)

This parameter specifies a short name for the documentation target, or product (for example, `mobile-phone-user-guide`, `hvac-installation-guide`).

Note: You can deploy documentation for multiple products on the same server.

Restriction: The following characters are not allowed in the value of this parameter: `< > / \ ' ( ) { } = ; * % + &`.

webhelp.product.version (available only for Feedback-enabled systems)

Specifies the documentation version number (for example, 1.0, 2.5, etc.). New user comments are bound to this version.

Note: Multiple documentation versions can be deployed on the same server.

Restriction: The following characters are not allowed in the value of this parameter: `< > / \ ' ( ) { } = ; * % + &`.

webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Editor uses for indexing Japanese content in the WebHelp pages.

webhelp.search.ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the search results that are displayed on the **Search** tab (default setting is `true`).

webhelp.skin.css

Path to a CSS file that sets the style theme in the output WebHelp pages. It can be one of the predefined skin CSS from the `OXYGEN_INSTALL_DIR\frameworks\docbook\xsl\com.oxygenxml.webhelp\predefined-skins` directory, or it can be a custom skin CSS generated with the *Oxygen Skin Builder* web application.

For more information about all the DocBook transformation parameters, go to *http://docbook.sourceforge.net/release/xsl/current/doc/fo/index.html*.

Related Information:

WebHelp Classic Output on page 815
WebHelp Classic With Feedback Output on page 818

WebHelp Classic Mobile System (Deprecated) on page 1023
Customizing WebHelp Classic Systems on page 999
Customizing WebHelp Classic Mobile Systems on page 1024

DocBook to PDF Output Customization

When the default layout and output of the DocBook to PDF transformation needs to be customized, follow these steps:

1. Create a custom version of the DocBook title spec file.

   You should start from a copy of the file `[OXYGEN_INSTALL_DIR]/frameworks/docbook/xsl/fo/titlepage.templates.xml` and customize it. The instructions for the spec file can be found *here.*

   An example of spec file:

```
<t:titlepage-content t:side="recto">
    <mediaobject/>
    <title
        t:named-template="book.verso.title"
        font-size="&hsize2;"
        font-weight="bold"
        font-family="{$title.font.family}"/>
    <corpauthor/>
      ...
</t:titlepage-content>
```

2. Generate a new XSLT stylesheet from the title spec file from the previous step.

   Apply `[OXYGEN_INSTALL_DIR]/frameworks/docbook/xsl/template/titlepage.xsl` to the title spec file. The result is an XSLT stylesheet (for example, `mytitlepages.xsl`).
3. Import `mytitlepages.xsl` in a *DocBook customization layer*.

   The customization layer is the stylesheet that will be applied to the XML document. The `mytitlepages.xsl` should be imported with an element like this:

```
<xsl:import href="dir-name/mytitlepages.xsl"/>
```

4. Insert a logo image in the XML document.

   The path to the logo image must be inserted in the `book/info/mediaobject` element of the XML document.
5. Apply the customization layer to the XML document.

   A quick way is to duplicate the transformation scenario **DocBook PDF** that is included with Oxygen XML Editor and set the customization layer in *the **XSL URL** property of the scenario*.

Related Information:

*The book **DocBook XSL: The Complete Guide** by Bob Stayton contains more details about customizing the PDF output.*
Video demonstration for creating a DocBook customization layer in Oxygen XML Editor.

DocBook to DITA Transformation

Oxygen XML Editor includes a built-in transformation scenario that is designed to convert DocBook content to DITA. This transformation scenario is based upon a DITA Open Toolkit plugin that is available at *sourceforge.net*.

To convert a DocBook document to DITA, follow these steps:

1. Use one of the following two methods to begin the transformation process:
   - To apply the transformation scenario to a newly opened file, use the **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on OS X))** action from the toolbar or the **Document** > **Transformation** menu.
   - To customize the transformation or change the scenario that is associated with the document, use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu.
2. Select the **DocBook to DITA** transformation scenario in the **DocBook 4** or **DocBook 5** section.
3. Click the **Apply associated** button to run the transformation.

Step Result: The transformation will convert as many of the DocBook elements into equivalent DITA elements as it can recognize in its mapping process. For elements that cannot be mapped, the transformation will insert XML comments so that you can see which elements could not be converted.

4. Adjust the resulting DITA composite to suit your needs. You may have to remove comments, fix validation errors, adjust certain attributes, or split the content into individual topics.

Related Information:

Editing a Transformation Scenario on page 925
Configure Transformation Scenario(s) Dialog Box on page 926

DocBook to EPUB Transformation

The EPUB specification recommends the use of *OpenType* fonts (recognized by their `.otf` file extension) when possible. To use a specific font, follow these steps:

1. Declare it in your CSS file, as in the following example:

```
@font-face {
font-family: "MyFont";
font-weight: bold;
font-style: normal;
src: url(fonts/MyFont.otf);
}
```

2. In the CSS, specify where this font is used. To set it as default for `h1` elements, use the `font-family` rule, as in the following example:

```
h1 {
font-size:20pt;
margin-bottom:20px;
font-weight: bold;
font-family: "MyFont";
text-align: center;
}
```

3. Open the ***Configure Transformation Scenario(s)*** *dialog box*, select the **DocBook EPUB** transformation scenario, and click **Edit**.
4. In the **Parameters** tab, set the `epub.embedded.fonts` parameter to `fonts/MyFont.otf`. If you need to provide more files, use commas to separate their file paths.

   Note: The `html.stylesheet` parameter allows you to include a custom CSS in the output EPUB.
5. Run the transformation scenario.

DocBook 5 Transformation Scenarios

Built-in transformation scenarios allow you to transform DocBook 5 documents to a variety of outputs, such as WebHelp, PDF, HTML, HTML Chunk, XHTML, XHTML Chunk, and EPUB. Oxygen XML Editor also includes a DocBook 5.1 transformation scenario for *Assembly* documents. All of them are listed in the **DocBook 5** section in the ***Configure Transformation Scenario(s)*** *dialog box*.

Related Information:

Editing a Transformation Scenario on page 925
Configure Transformation Scenario(s) Dialog Box on page 926

DocBook 5 to WebHelp Output

DocBook 5 documents can be transformed into several types of WebHelp systems.

WebHelp Classic Output

To publish a DocBook 5 document as a **WebHelp Classic** system, follow these steps:

1. Click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
2. Select the **DocBook WebHelp Classic** scenario from the **DocBook 5** section.
3. Click **Apply associated**.

When the **DocBook WebHelp Classic** transformation is complete, the output is automatically opened in your default browser.

WebHelp Classic with Feedback Output

To publish a DocBook 5 document as a **WebHelp Classic with Feedback** system, follow these steps:

1. Click **Configure Transformation Scenarios**.
2. Select the **DocBook WebHelp Classic with Feedback** scenario from the **DocBook 5** section.
3. Click **Apply associated**.
4. Enter the documentation product ID and the documentation version.

   When the **DocBook WebHelp Classic with Feedback** transformation is complete, your default browser opens the `installation.html` file. This file contains information about the output location, system requirements, installation instructions, and deployment of the output. Follow the instructions to complete the system deployment. For more information, see *Deploying a Feedback-Enabled WebHelp System*.

To watch our video demonstration about the feedback-enabled WebHelp system, go to *https://www.oxygenxml.com/demo/Feedback_Enabled_WebHelp.html*.

WebHelp Classic Mobile Output

To publish a DocBook 5 document as a **WebHelp Classic Mobile** system, follow these steps:

1. Click **Configure Transformation Scenarios**.
2. Select the **DocBook WebHelp Classic Mobile** scenario from the **DocBook 5** section.
3. Click **Apply associated**.

When the **DocBook WebHelp Classic Mobile** transformation is complete, the output is automatically opened in your default browser.

Customizing WebHelp Transformation Scenarios

To customize a DocBook WebHelp transformation scenario, you can edit various parameters, including the following most commonly used ones:

args.default.language

This parameter is used if the language is not detected in the *DITA map*. The default value is `en-us`.

clean.output

Deletes all files from the output folder before the transformation is performed (only no and `yes` values are valid and the default value is no).

l10n.gentext.default.language

This parameter is used to identify the correct stemmer that differs from language to language. For example, for English the value of this parameter is `en` or for French it is `fr`, and so on.

use.stemming

Controls whether or not you want to include stemming search algorithms into the published output (default setting is `false`).

webhelp.copyright

Adds a small copyright text that appears at the end of the **Table of Contents** pane.

webhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

webhelp.favicon

The file path that points to an image to be used as a *favicon* in the WebHelp output.

webhelp.footer.file

Path to an XML file that includes the footer content for your WebHelp output pages. You can use this parameter to integrate social media features (such as widgets for Facebook™, Twitter™, Google Analytics, or

Google+™). The file must be well-formed, each widget must be in separate `div` or `span` element, and the code for each `script` element is included in an XML comment (also, the start and end tags for the XML comment must be on a separate line). The following code exert is an example for adding a Facebook™ widget:

```
<div id="facebook">
  <div id="fb-root"/>
  <script>
    <!-- (function(d, s, id) { var js, fjs = d.getElementsByTagName(s)[0];
      if (d.getElementById(id)) return;
      js = d.createElement(s); js.id = id;
      js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
      fjs.parentNode.insertBefore(js, fjs); }
       (document, 'script', 'facebook-jssdk')); -->
    </script>
    <div data-share="true" data-show-faces="true" data-action="like"
       data-layout="standard" class="fb-like"/>
</div>
```

webhelp.footer.include

Specifies whether or not to include footer in each WebHelp page. Possible values: `yes`, no. If set to no, no footer is added to the WebHelp pages. If set to `yes` and the `webhelp.footer.file` parameter has a value, then the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default Oxygen XML Editor footer is inserted in each WebHelp page.

webhelp.logo.image.target.url

Specifies a target URL that is set on the logo image. When you click the logo image, you will be redirected to this address.

webhelp.logo.image

Specifies a path to an image displayed as a logo in the left side of the output header.

webhelp.product.id (available only for Feedback-enabled systems)

This parameter specifies a short name for the documentation target, or product (for example, `mobile-phone-user-guide`, `hvac-installation-guide`).

Note: You can deploy documentation for multiple products on the same server.

Restriction: The following characters are not allowed in the value of this parameter: `< > / \ ' ( ) { } = ; * % + &`.

webhelp.product.version (available only for Feedback-enabled systems)

Specifies the documentation version number (for example, 1.0, 2.5, etc.). New user comments are bound to this version.

Note: Multiple documentation versions can be deployed on the same server.

Restriction: The following characters are not allowed in the value of this parameter: `< > / \ ' ( ) { } = ; * % + &`.

webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Editor uses for indexing Japanese content in the WebHelp pages.

webhelp.search.ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the search results that are displayed on the **Search** tab (default setting is `true`).

webhelp.skin.css

Path to a CSS file that sets the style theme in the output WebHelp pages. It can be one of the predefined skin CSS from the `OXYGEN_INSTALL_DIR\frameworks\docbook\xsl\com.oxygenxml.webhelp\predefined-skins` directory, or it can be a custom skin CSS generated with the *Oxygen Skin Builder* web application.

For more information about all the DocBook transformation parameters, go to *http://docbook.sourceforge.net/release/xsl/current/doc/fo/index.html*.

Related Information:

WebHelp Classic Output on page 815

WebHelp Classic With Feedback Output on page 818
WebHelp Classic Mobile System (Deprecated) on page 1023
Customizing WebHelp Classic Systems on page 999
Customizing WebHelp Classic Mobile Systems on page 1024

DocBook to PDF Output Customization

When the default layout and output of the DocBook to PDF transformation needs to be customized, follow these steps:

1. Create a custom version of the DocBook title spec file.

   You should start from a copy of the file `[OXYGEN_INSTALL_DIR]/frameworks/docbook/xsl/fo/titlepage.templates.xml` and customize it. The instructions for the spec file can be found *here.*

   An example of spec file:

```
<t:titlepage-content t:side="recto">
    <mediaobject/>
    <title
        t:named-template="book.verso.title"
        font-size="&hsize2;"
        font-weight="bold"
        font-family="{$title.font.family}"/>
    <corpauthor/>
      ...
</t:titlepage-content>
```

2. Generate a new XSLT stylesheet from the title spec file from the previous step.

   Apply `[OXYGEN_INSTALL_DIR]/frameworks/docbook/xsl/template/titlepage.xsl` to the title spec file. The result is an XSLT stylesheet (for example, `mytitlepages.xsl`).

3. Import `mytitlepages.xsl` in a *DocBook customization layer*.

   The customization layer is the stylesheet that will be applied to the XML document. The `mytitlepages.xsl` should be imported with an element like this:

```
<xsl:import href="dir-name/mytitlepages.xsl"/>
```

4. Insert a logo image in the XML document.

   The path to the logo image must be inserted in the `book/info/mediaobject` element of the XML document.

5. Apply the customization layer to the XML document.

   A quick way is to duplicate the transformation scenario **DocBook PDF** that is included with Oxygen XML Editor and set the customization layer in *the **XSL URL** property of the scenario*.

Related Information:

*The book **DocBook XSL: The Complete Guide** by Bob Stayton contains more details about customizing the PDF output.*

Video demonstration for creating a DocBook customization layer in Oxygen XML Editor.

Show Comments and Tracked Changes in DocBook 5 PDF Output

To include comments and *tracked changes* (stored within your DocBook 5 documents) in the PDF output, follow these steps:

1. Click the **Configure Transformation Scenario(s)** button.
2. Select **DocBook PDF (Show Change Tracking and Comments)** in the **DocBook 5** section.
3. If you need to configure the transformation, click the ***Edit*** or ***Duplicate*** button, make your changes to the scenario, and click **OK**.
4. Click the **Apply Associated** button to run the transformation scenario.

   Result: Comment threads and tracked changes will now appear in the PDF output. Details about each comment or change will be available in the footer section for each page.

DocBook to DITA Transformation

Oxygen XML Editor includes a built-in transformation scenario that is designed to convert DocBook content to DITA. This transformation scenario is based upon a DITA Open Toolkit plugin that is available at *sourceforge.net*.

To convert a DocBook document to DITA, follow these steps:

1. Use one of the following two methods to begin the transformation process:
   - To apply the transformation scenario to a newly opened file, use the **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on OS X))** action from the toolbar or the **Document** > **Transformation** menu.
   - To customize the transformation or change the scenario that is associated with the document, use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu.
2. Select the **DocBook to DITA** transformation scenario in the **DocBook 4** or **DocBook 5** section.
3. Click the **Apply associated** button to run the transformation.

   Step Result: The transformation will convert as many of the DocBook elements into equivalent DITA elements as it can recognize in its mapping process. For elements that cannot be mapped, the transformation will insert XML comments so that you can see which elements could not be converted.
4. Adjust the resulting DITA composite to suit your needs. You may have to remove comments, fix validation errors, adjust certain attributes, or split the content into individual topics.

Related Information:

Editing a Transformation Scenario on page 925
Configure Transformation Scenario(s) Dialog Box on page 926

DocBook to EPUB Transformation

The EPUB specification recommends the use of *OpenType* fonts (recognized by their `.otf` file extension) when possible. To use a specific font, follow these steps:

1. Declare it in your CSS file, as in the following example:

```
@font-face {
font-family: "MyFont";
font-weight: bold;
font-style: normal;
src: url(fonts/MyFont.otf);
}
```

2. In the CSS, specify where this font is used. To set it as default for `h1` elements, use the `font-family` rule, as in the following example:

```
h1 {
font-size:20pt;
margin-bottom:20px;
font-weight: bold;
font-family: "MyFont";
text-align: center;
}
```

3. Open the ***Configure Transformation Scenario(s)** dialog box*, select the **DocBook EPUB** transformation scenario, and click **Edit**.
4. In the **Parameters** tab, set the `epub.embedded.fonts` parameter to `fonts/MyFont.otf`. If you need to provide more files, use commas to separate their file paths.

   Note: The `html.stylesheet` parameter allows you to include a custom CSS in the output EPUB.
5. Run the transformation scenario.

DITA Map Transformation Scenarios

The following types of transformations scenarios are available for *DITA maps*:

- Predefined transformation scenarios allow you to transform a *DITA map* to a variety of outputs, such as WebHelp, PDF, ODF, XHTML, EPUB, CHM, Kindle, and MS Word.

- **Run DITA-OT Integrator** - Use this transformation scenario if you want to *integrate a DITA-OT plugin*. This scenario runs an Ant task that integrates all the plugins from the DITA-OT/plugins directory.
- **DITA Map Metrics Report** - Use this type of transformation scenario if you want to generate a *DITA map* statistics report. They contain information such as:
  - The number of processed maps and topics.
  - Content reuse percentage.
  - Number of elements, attributes, words, and characters used in the entire *DITA map* structure.
  - DITA conditional processing attributes used in the *DITA maps*.
  - Words count.
  - Information types such as number of containing *maps*, *bookmaps*, or topics.

Related Information:

Editing a Transformation Scenario on page 925
Configure Transformation Scenario(s) Dialog Box on page 926

DITA Map to WebHelp Output

DITA maps can be transformed into a variety of WebHelp systems designed to suit your specific needs. This section contains the procedures for obtaining the output for the variants of the WebHelp system.

Related Information:

WebHelp System Output on page 939

WebHelp Responsive Output

To publish a *DITA map* as a **WebHelp Responsive** system, follow these steps:

1. Select the **Configure Transformation Scenario(s)** action from the ***DITA Maps Manager*** toolbar.
2. Select the **DITA Map WebHelp Responsive** scenario from the **DITA Map** section.
3. If you want to configure the transformation, click the **Edit** button.

   Step Result: This opens an **Edit scenario** configuration dialog box that allows you to configure various options in the following tabs:

   - *Templates Tab* - This tab contains a set of predefined skins that you can use for the layout of your WebHelp system output.
   - *Parameters Tab* - This tab includes numerous parameters that can be set to customize your WebHelp system output. See the *Parameters section* below for details about the most commonly used parameters for WebHelp Responsive transformations.
   - *Filters Tab* - This tab allows you to filter certain content elements from the generated output.
   - *Advanced Tab* - This tab allows you to specify some advanced options for the transformation scenario.
   - *Output Tab* - This tab allows you to configure options that are related to the location where the output is generated.
4. Click **Apply associated** to process the transformation.

   When the **DITA Map WebHelp Responsive** transformation is complete, the output is automatically opened in your default browser.

Parameters for Customizing WebHelp Responsive Output

To customize a transformation scenario, you can edit various parameters, including the following most commonly used ones:

args.default.language

This parameter is used if the language is not detected in the *DITA map*. The default value is `en-us`.

clean.output

Deletes all files from the output folder before the transformation is performed (only `no` and `yes` values are valid and the default value is `no`).

editlink.web.author.url

Use this parameter in conjunction with `editlink.web.author.url` to add an *Edit* link next to the topic title in the WebHelp output. When a user clicks the link, the topic is opened in Oxygen XML Web Author where they can make changes that can be saved to a file server. The value should be set as the custom URL of the *main DITA map*. For example, a GitHub custom URL might look like this: `https://getFileContent/oxyengxml/userguide/master/UserGuide.ditamap`. For more details about custom URLs, see *this section*.

editlink.web.author.url

This parameter needs to be used in conjunction with `editlink.remote.ditamap.url` to add an *Edit* link next to the topic title in the WebHelp output. When a user clicks the link, the topic is opened in Oxygen XML Web Author where they can make changes that can be saved to a file server. The value should be set as the URL of the Web Author installation. For example: `https://www.oxygenxml.com/webapp-demo-aws/app/oxygen.html`.

fix.external.refs.com.oxygenxml (Only supported when the DITA OT transformation process is started from Oxygen XML Editor)

The DITA Open Toolkit usually has problems processing references that point to locations outside of the directory of the processed *DITA map*. This parameter is used to specify whether or not the application should try to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references. The fix has no impact on your edited DITA content. Allowed values: `true` or `false` (default).

use.stemming

Controls whether or not you want to include stemming search algorithms into the published output (default setting is `false`).

webhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

webhelp.favicon

The file path that points to an image to be used as a *favicon* in the WebHelp output.

webhelp.reload.stylesheet

Set this parameter to `true` if you have out of memory problems when generating WebHelp. It will increase processing time but decrease the memory footprint. The default value is `false`.

webhelp.search.custom.excludes.file

The path of the file that contains name patterns for HTML files that should not be indexed by the WebHelp search engine. Each exclude pattern must be on a new line. The patterns are considered to be relative to the output directory, and they accept wildcards such as `'*'` (matches zero or more characters) or `'?'` (matches one character). For more information about the patterns, see *https://ant.apache.org/manual/dirtasks.html#patterns*.

webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Editor uses for indexing Japanese content in the WebHelp pages.

webhelp.search.ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the search results that are displayed on the **Search** tab (default setting is `true`).

webhelp.show.changes.and.comments

When set to `yes`, user comments, replies to comments, and tracked changes are published in the WebHelp output. The default value is no.

webhelp.sitemap.base.url

Base URL for all the `loc` elements in the generated `sitemap.xml` file. The value of a `loc` element is computed as the relative file path from the `href` attribute of a `topicref` element from the *DITA map*, appended to this base URL value. The `loc` element is mandatory in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `sitemap.xml` file is not generated.

webhelp.sitemap.change.frequency

The value of the `changefreq` element in the generated `sitemap.xml` file. The `changefreq` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `changefreq` element is not added in `sitemap.xml`. Allowed values: <empty string> (default), `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, `never`.

webhelp.sitemap.priority

The value of the `priority` element in the generated `sitemap.xml` file. It can be set to any fractional number between 0.0 (least important priority) and 1.0 (most important priority). For example, 0.3, 0.5, or 0.8. The `priority` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `priority` element is not added in `sitemap.xml`.

Parameters Specific to WebHelp Responsive Output

webhelp.enable.search.autocomplete

Specifies if the *Autocomplete* feature is enabled in the WebHelp search text field. The default value is `yes`.

webhelp.fragment.after.body

In the generated output it displays a given XHTML fragment after the page body. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.logo_and_title

In the generated output it displays a given XHTML fragment after the logo and title. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.main.page.search

In the generated output it displays a given XHTML fragment after the search field. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.toc_or_tiles

In the generated output it displays a given XHTML fragment after the table of contents or tiles in the main page. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.top_menu

In the generated output it displays a given XHTML fragment after the top menu. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.body

In the generated output it displays a given XHTML fragment before the page body. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.logo_and_title

In the generated output it displays a given XHTML fragment before the logo and title. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.main.page.search

In the generated output it displays a given XHTML fragment before the search field. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.toc_or_tiles

In the generated output it displays a given XHTML fragment before the table of contents or tiles in the main page. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.top_menu

In the generated output it displays a given XHTML fragment before the top menu. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.footer

In the generated output it displays a given XHTML fragment as the page footer. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.head

In the generated output it displays a given XHTML fragment as a page header. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.welcome

In the generated output it displays a given XHTML fragment as a welcome message (or title). The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.merge.nested.topics.related.links

Specifies if the related links from nested topics will be merged with the links in the parent topic. Thus the links will be moved from the topic content to the related links component and all of the links from the same group (for example, *Related Tasks*, *Related References*, *Related Information*) are merged into a single group. The default value is yes.

webhelp.show.breadcrumb

Specifies if the breadcrumb component will be presented in the output. The default value is yes.

webhelp.show.child.links

Specifies if child links will be generated in the output for all topics that have subtopics. The default value is no.

webhelp.show.indexterms.link

Specifies if an icon that links to the index will be presented in the output. The default value is yes.

webhelp.show.main.page.tiles

Specifies if the tiles component will be presented in the main page of the output. For a *tree* style layout, this parameter should be set to no.

webhelp.show.main.page.toc

Specifies if the table of contents will be presented in the main page of the output. The default value is yes.

webhelp.show.navigation.links

Specifies if navigation links will be presented in the output. The default value is yes.

webhelp.show.print.link

Specifies if a print link or icon will be presented within each topic in the output. The default value is yes.

webhelp.show.related.links

Specifies if the related links component will be presented in the WebHelp Responsive output. The default value is yes. The webhelp.merge.nested.topics.related.links parameter can used in conjunction with this one to merge the related links from nested topics into the links in the parent topic.

webhelp.show.side.toc

Specifies if a side table of contents will be presented on the right side of each topic in the output. The default value is yes.

webhelp.show.top.menu

Specifies if a menu will be presented at the topic of the main page in the output. The default value is yes.

webhelp.top.menu.depth

Specifies the maximum depth level of the topics that will be included in the top menu. The default value is 2.

Related Information:

Customizing the WebHelp Responsive Output on page 963

WebHelp Responsive System on page 940

WebHelp Responsive with Feedback Output

To publish a *DITA map* as a **WebHelp Responsive with Feedback** system, follow these steps:

1. Select the **Configure Transformation Scenario(s)** action from the ***DITA Maps Manager*** toolbar.
2. Select the **DITA Map WebHelp Responsive with Feedback** scenario from the **DITA Map** section.
3. If you want to configure the transformation, click the **Edit** button.

   Step Result: This opens an **Edit scenario** configuration dialog box that allows you to configure various options in the following tabs:

- *Templates Tab* - This tab contains a set of predefined skins that you can use for the layout of your WebHelp system output.
- *Parameters Tab* - This tab includes numerous parameters that can be set to customize your WebHelp system output. See the *Parameters section* below for details about the most commonly used parameters for WebHelp Responsive transformations.
- *Filters Tab* - This tab allows you to filter certain content elements from the generated output.
- *Advanced Tab* - This tab allows you to specify some advanced options for the transformation scenario.
- *Output Tab* - This tab allows you to configure options that are related to the location where the output is generated.

4. Click **Apply associated** to begin the transformation.
5. Enter the documentation product ID (value for the `webhelp.product.id` parameter) and the documentation version (value for the `webhelp.product.version` parameter).

   When the **DITA Map WebHelp Responsive with Feedback** transformation is complete, your default browser opens the `installation.html` file. This file contains information about the output location, system requirements, installation instructions, and deployment of the output. Follow the *instructions to complete the system deployment*.

Parameters for Customizing WebHelp Responsive with Feedback Output

To customize a transformation scenario, you can edit various parameters, including the following most commonly used ones:

args.default.language

This parameter is used if the language is not detected in the *DITA map*. The default value is `en-us`.

clean.output

Deletes all files from the output folder before the transformation is performed (only no and `yes` values are valid and the default value is no).

editlink.web.author.url

Use this parameter in conjunction with `editlink.web.author.url` to add an *Edit* link next to the topic title in the WebHelp output. When a user clicks the link, the topic is opened in Oxygen XML Web Author where they can make changes that can be saved to a file server. The value should be set as the custom URL of the *main DITA map*. For example, a GitHub custom URL might look like this: `https://getFileContent/oxyengxml/userguide/master/UserGuide.ditamap`. For more details about custom URLs, see *this section*.

editlink.web.author.url

This parameter needs to be used in conjunction with `editlink.remote.ditamap.url` to add an *Edit* link next to the topic title in the WebHelp output. When a user clicks the link, the topic is opened in Oxygen XML Web Author where they can make changes that can be saved to a file server. The value should be set as the URL of the Web Author installation. For example: `https://www.oxygenxml.com/webapp-demo-aws/app/oxygen.html`.

fix.external.refs.com.oxygenxml (Only supported when the DITA OT transformation process is started from Oxygen XML Editor)

The DITA Open Toolkit usually has problems processing references that point to locations outside of the directory of the processed *DITA map*. This parameter is used to specify whether or not the application should try to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references. The fix has no impact on your edited DITA content. Allowed values: `true` or `false` (default).

use.stemming

Controls whether or not you want to include stemming search algorithms into the published output (default setting is `false`).

webhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

webhelp.favicon

The file path that points to an image to be used as a *favicon* in the WebHelp output.

webhelp.reload.stylesheet

Set this parameter to `true` if you have out of memory problems when generating WebHelp. It will increase processing time but decrease the memory footprint. The default value is `false`.

webhelp.search.custom.excludes.file

The path of the file that contains name patterns for HTML files that should not be indexed by the WebHelp search engine. Each exclude pattern must be on a new line. The patterns are considered to be relative to the output directory, and they accept wildcards such as `'*'` (matches zero or more characters) or `'?'` (matches one character). For more information about the patterns, see *https://ant.apache.org/manual/dirtasks.html#patterns*.

webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Editor uses for indexing Japanese content in the WebHelp pages.

webhelp.search.ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the search results that are displayed on the **Search** tab (default setting is `true`).

webhelp.show.changes.and.comments

When set to `yes`, user comments, replies to comments, and tracked changes are published in the WebHelp output. The default value is no.

webhelp.sitemap.base.url

Base URL for all the `loc` elements in the generated `sitemap.xml` file. The value of a `loc` element is computed as the relative file path from the `href` attribute of a `topicref` element from the *DITA map*, appended to this base URL value. The `loc` element is mandatory in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `sitemap.xml` file is not generated.

webhelp.sitemap.change.frequency

The value of the `changefreq` element in the generated `sitemap.xml` file. The `changefreq` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `changefreq` element is not added in `sitemap.xml`. Allowed values: <empty string> (default), `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, `never`.

webhelp.sitemap.priority

The value of the `priority` element in the generated `sitemap.xml` file. It can be set to any fractional number between 0.0 (least important priority) and 1.0 (most important priority). For example, 0.3, 0.5, or 0.8. The `priority` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `priority` element is not added in `sitemap.xml`.

Parameters Specific to WebHelp Responsive with Feedback Output

webhelp.enable.search.autocomplete

Specifies if the *Autocomplete* feature is enabled in the WebHelp search text field. The default value is `yes`.

webhelp.fragment.after.body

In the generated output it displays a given XHTML fragment after the page body. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.logo_and_title

In the generated output it displays a given XHTML fragment after the logo and title. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.main.page.search

In the generated output it displays a given XHTML fragment after the search field. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.toc_or_tiles

In the generated output it displays a given XHTML fragment after the table of contents or tiles in the main page. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.top_menu

In the generated output it displays a given XHTML fragment after the top menu. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.body

In the generated output it displays a given XHTML fragment before the page body. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.logo_and_title

In the generated output it displays a given XHTML fragment before the logo and title. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.main.page.search

In the generated output it displays a given XHTML fragment before the search field. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.toc_or_tiles

In the generated output it displays a given XHTML fragment before the table of contents or tiles in the main page. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.top_menu

In the generated output it displays a given XHTML fragment before the top menu. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.footer

In the generated output it displays a given XHTML fragment as the page footer. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.head

In the generated output it displays a given XHTML fragment as a page header. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.welcome

In the generated output it displays a given XHTML fragment as a welcome message (or title). The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.merge.nested.topics.related.links

Specifies if the related links from nested topics will be merged with the links in the parent topic. Thus the links will be moved from the topic content to the related links component and all of the links from the same group (for example, *Related Tasks*, *Related References*, *Related Information*) are merged into a single group. The default value is yes.

webhelp.show.breadcrumb

Specifies if the breadcrumb component will be presented in the output. The default value is yes.

webhelp.show.child.links

Specifies if child links will be generated in the output for all topics that have subtopics. The default value is no.

webhelp.show.indexterms.link

Specifies if an icon that links to the index will be presented in the output. The default value is yes.

webhelp.show.main.page.tiles

Specifies if the tiles component will be presented in the main page of the output. For a *tree* style layout, this parameter should be set to no.

webhelp.show.main.page.toc

Specifies if the table of contents will be presented in the main page of the output. The default value is yes.

webhelp.show.navigation.links

Specifies if navigation links will be presented in the output. The default value is yes.

webhelp.show.print.link

Specifies if a print link or icon will be presented within each topic in the output. The default value is yes.

webhelp.show.related.links

Specifies if the related links component will be presented in the WebHelp Responsive output. The default value is `yes`. The webhelp.merge.nested.topics.related.links parameter can used in conjunction with this one to merge the related links from nested topics into the links in the parent topic.

webhelp.show.side.toc

Specifies if a side table of contents will be presented on the right side of each topic in the output. The default value is `yes`.

webhelp.show.top.menu

Specifies if a menu will be presented at the topic of the main page in the output. The default value is `yes`.

webhelp.top.menu.depth

Specifies the maximum depth level of the topics that will be included in the top menu. The default value is 2.

Related Information:

Customizing the WebHelp Responsive Output on page 963
WebHelp Responsive with Feedback System on page 944

WebHelp Classic Output

To publish a *DITA map* as a **WebHelp Classic** system, follow these steps:

1. Select the **Configure Transformation Scenario(s)** action from the ***DITA Maps Manager*** toolbar.
2. Select the **DITA Map WebHelp Classic** scenario from the **DITA Map** section.
3. If you want to configure the transformation, click the **Edit** button.

   Step Result: This opens an **Edit scenario** configuration dialog box that allows you to configure various options in the following tabs:

   - *Skins Tab* - This tab contains a set of predefined skins that you can use for the layout of your WebHelp system output.
   - *Parameters Tab* - This tab includes numerous parameters that can be set to customize your WebHelp system output. See the *Parameters section* below for details about the most commonly used parameters for WebHelp Responsive transformations.
   - *Filters Tab* - This tab allows you to filter certain content elements from the generated output.
   - *Advanced Tab* - This tab allows you to specify some advanced options for the transformation scenario.
   - *Output Tab* - This tab allows you to configure options that are related to the location where the output is generated.
4. Click **Apply associated** to process the transformation.

   When the **DITA Map WebHelp Classic** transformation is complete, the output is automatically opened in your default browser.

To further customize this transformation, you can edit various parameters, including the following most commonly used ones:

Parameters for Customizing WebHelp Classic Output

To customize a transformation scenario, you can edit various parameters, including the following most commonly used ones:

args.default.language

This parameter is used if the language is not detected in the *DITA map*. The default value is `en-us`.

clean.output

Deletes all files from the output folder before the transformation is performed (only `no` and `yes` values are valid and the default value is `no`).

fix.external.refs.com.oxygenxml (Only supported when the DITA OT transformation process is started from Oxygen XML Editor)

The DITA Open Toolkit usually has problems processing references that point to locations outside of the directory of the processed *DITA map*. This parameter is used to specify whether or not the application should

try to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references. The fix has no impact on your edited DITA content. Allowed values: `true` or `false` (default).

use.stemming

Controls whether or not you want to include stemming search algorithms into the published output (default setting is `false`).

webhelp.body.script

URL value that specifies the location of a well-formed XHTML file containing the custom script that will be copied in the `<body>` section of every WebHelp page.

webhelp.copyright

Adds a small copyright text that appears at the end of the **Table of Contents** pane.

webhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

webhelp.favicon

The file path that points to an image to be used as a *favicon* in the WebHelp output.

webhelp.footer.file

Path to an XML file that includes the footer content for your WebHelp output pages. You can use this parameter to integrate social media features (such as widgets for Facebook™, Twitter™, Google Analytics, or Google+™). The file must be well-formed, each widget must be in separate `div` or `span` element, and the code for each `script` element is included in an XML comment (also, the start and end tags for the XML comment must be on a separate line). The following code exert is an example for adding a Facebook™ widget:

```
<div id="facebook">
  <div id="fb-root"/>
  <script>
    <!-- (function(d, s, id) { var js, fjs = d.getElementsByTagName(s)[0];
      if (d.getElementById(id)) return;
      js = d.createElement(s); js.id = id;
      js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
      fjs.parentNode.insertBefore(js, fjs); }
       (document, 'script', 'facebook-jssdk')); -->
    </script>
    <div data-share="true" data-show-faces="true" data-action="like"
       data-layout="standard" class="fb-like"/>
</div>
```

webhelp.footer.include

Specifies whether or not to include footer in each WebHelp page. Possible values: `yes`, no. If set to no, no footer is added to the WebHelp pages. If set to `yes` and the `webhelp.footer.file` parameter has a value, then the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default Oxygen XML Editor footer is inserted in each WebHelp page.

webhelp.google.search.results

A file path that specifies the location of a well-formed XHTML file containing the Google Custom Search Engine element `gcse:searchresults-only`. You can use all supported attributes for this element. It is recommend to set the `linkTarget` attribute to `frm` for frameless (*iframe*) version of WebHelp or to `contentWin` for the frameset version of WebHelp. The default value for this attribute is `_blank` and the search results will be loaded in a new window. If this parameter is not specified, the following code will be used `<gcse:searchresults-only linkTarget="frm"></gcse:searchresults-only>`

webhelp.google.search.script

A file path that specifies the location of a well-formed XHTML file containing the Custom Search Engine script from Google.

webhelp.head.script

URL value that specifies the location of a well-formed XHTML file containing the custom script that will be copied in the `<head>` section of every WebHelp page.

webhelp.logo.image.target.url

Specifies a target URL that is set on the logo image. When you click the logo image, you will be redirected to this address.

webhelp.logo.image

Specifies a path to an image displayed as a logo in the left side of the output header.

webhelp.reload.stylesheet

Set this parameter to `true` if you have out of memory problems when generating WebHelp. It will increase processing time but decrease the memory footprint. The default value is `false`.

webhelp.search.custom.excludes.file

The path of the file that contains name patterns for HTML files that should not be indexed by the WebHelp search engine. Each exclude pattern must be on a new line. The patterns are considered to be relative to the output directory, and they accept wildcards such as `'*'` (matches zero or more characters) or `'?'` (matches one character). For more information about the patterns, see *https://ant.apache.org/manual/dirtasks.html#patterns*.

webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Editor uses for indexing Japanese content in the WebHelp pages.

webhelp.search.ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the search results that are displayed on the **Search** tab (default setting is `true`).

webhelp.show.changes.and.comments

When set to `yes`, user comments, replies to comments, and tracked changes are published in the WebHelp output. The default value is no.

webhelp.sitemap.base.url

Base URL for all the `loc` elements in the generated `sitemap.xml` file. The value of a `loc` element is computed as the relative file path from the `href` attribute of a `topicref` element from the *DITA map*, appended to this base URL value. The `loc` element is mandatory in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `sitemap.xml` file is not generated.

webhelp.sitemap.change.frequency

The value of the `changefreq` element in the generated `sitemap.xml` file. The `changefreq` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `changefreq` element is not added in `sitemap.xml`. Allowed values: <empty string> (default), `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, `never`.

webhelp.sitemap.priority

The value of the `priority` element in the generated `sitemap.xml` file. It can be set to any fractional number between 0.0 (least important priority) and 1.0 (most important priority). For example, 0.3, 0.5, or 0.8. The `priority` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `priority` element is not added in `sitemap.xml`.

Related Information:

Customizing WebHelp Classic Systems on page 999

WebHelp Classic System on page 988

WebHelp Classic With Feedback Output

To publish a *DITA map* as a **WebHelp Classic with Feedback** system, follow these steps:

1. Select the **Configure Transformation Scenario(s)** action from the ***DITA Maps Manager*** toolbar.
2. Select the **DITA Map WebHelp Classic with Feedback** scenario from the **DITA Map** section.
3. If you want to configure the transformation, click the **Edit** button.

   Step Result: This opens an **Edit scenario** configuration dialog box that allows you to configure various options in the following tabs:

   - *Skins Tab* - This tab contains a set of predefined skins that you can use for the layout of your WebHelp system output.

- *Parameters Tab* - This tab includes numerous parameters that can be set to customize your WebHelp system output. See the *Parameters section* below for details about the most commonly used parameters for WebHelp Responsive transformations.
- *Filters Tab* - This tab allows you to filter certain content elements from the generated output.
- *Advanced Tab* - This tab allows you to specify some advanced options for the transformation scenario.
- *Output Tab* - This tab allows you to configure options that are related to the location where the output is generated.

4. Click **Apply associated** to begin the transformation.
5. Enter the documentation product ID (value for the `webhelp.product.id` parameter) and the documentation version (value for the `webhelp.product.version` parameter).

   When the **DITA Map WebHelp Classic with Feedback** transformation is complete, your default browser opens the `installation.html` file. This file contains information about the output location, system requirements, installation instructions, and deployment of the output. Follow the *instructions to complete the system deployment*.

To watch our video demonstration about the feedback-enabled WebHelp system, go to *https://www.oxygenxml.com/demo/Feedback_Enabled_WebHelp.html*.

Parameters for Customizing WebHelp Classic with Feedback Output

To customize a transformation scenario, you can edit various parameters, including the following most commonly used ones:

args.default.language

This parameter is used if the language is not detected in the *DITA map*. The default value is `en-us`.

clean.output

Deletes all files from the output folder before the transformation is performed (only no and `yes` values are valid and the default value is no).

fix.external.refs.com.oxygenxml (Only supported when the DITA OT transformation process is started from Oxygen XML Editor)

The DITA Open Toolkit usually has problems processing references that point to locations outside of the directory of the processed *DITA map*. This parameter is used to specify whether or not the application should try to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references. The fix has no impact on your edited DITA content. Allowed values: `true` or `false` (default).

use.stemming

Controls whether or not you want to include stemming search algorithms into the published output (default setting is `false`).

webhelp.body.script

URL value that specifies the location of a well-formed XHTML file containing the custom script that will be copied in the `<body>` section of every WebHelp page.

webhelp.copyright

Adds a small copyright text that appears at the end of the **Table of Contents** pane.

webhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

webhelp.favicon

The file path that points to an image to be used as a *favicon* in the WebHelp output.

webhelp.footer.file

Path to an XML file that includes the footer content for your WebHelp output pages. You can use this parameter to integrate social media features (such as widgets for Facebook™, Twitter™, Google Analytics, or Google+™). The file must be well-formed, each widget must be in separate `div` or `span` element, and the code for each `script` element is included in an XML comment (also, the start and end tags for the XML comment must be on a separate line). The following code exert is an example for adding a Facebook™ widget:

```
<div id="facebook">
```

```
  <div id="fb-root"/>
  <script>
    <!-- (function(d, s, id) { var js, fjs = d.getElementsByTagName(s)[0];
      if (d.getElementById(id)) return;
      js = d.createElement(s); js.id = id;
      js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
      fjs.parentNode.insertBefore(js, fjs); }
       (document, 'script', 'facebook-jssdk')); -->
    </script>
    <div data-share="true" data-show-faces="true" data-action="like"
       data-layout="standard" class="fb-like"/>
</div>
```

webhelp.footer.include

Specifies whether or not to include footer in each WebHelp page. Possible values: `yes`, no. If set to no, no footer is added to the WebHelp pages. If set to `yes` and the `webhelp.footer.file` parameter has a value, then the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default Oxygen XML Editor footer is inserted in each WebHelp page.

webhelp.google.search.results

A file path that specifies the location of a well-formed XHTML file containing the Google Custom Search Engine element `gcse:searchresults-only`. You can use all supported attributes for this element. It is recommend to set the `linkTarget` attribute to `frm` for frameless (*iframe*) version of WebHelp or to `contentWin` for the frameset version of WebHelp. The default value for this attribute is `_blank` and the search results will be loaded in a new window. If this parameter is not specified, the following code will be used `<gcse:searchresults-only linkTarget="frm"></gcse:searchresults-only>`

webhelp.google.search.script

A file path that specifies the location of a well-formed XHTML file containing the Custom Search Engine script from Google.

webhelp.head.script

URL value that specifies the location of a well-formed XHTML file containing the custom script that will be copied in the `<head>` section of every WebHelp page.

webhelp.logo.image.target.url

Specifies a target URL that is set on the logo image. When you click the logo image, you will be redirected to this address.

webhelp.logo.image

Specifies a path to an image displayed as a logo in the left side of the output header.

webhelp.reload.stylesheet

Set this parameter to `true` if you have out of memory problems when generating WebHelp. It will increase processing time but decrease the memory footprint. The default value is `false`.

webhelp.search.custom.excludes.file

The path of the file that contains name patterns for HTML files that should not be indexed by the WebHelp search engine. Each exclude pattern must be on a new line. The patterns are considered to be relative to the output directory, and they accept wildcards such as `'*'` (matches zero or more characters) or `'?'` (matches one character). For more information about the patterns, see *https://ant.apache.org/manual/dirtasks.html#patterns*.

webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Editor uses for indexing Japanese content in the WebHelp pages.

webhelp.search.ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the search results that are displayed on the **Search** tab (default setting is `true`).

webhelp.show.changes.and.comments

When set to `yes`, user comments, replies to comments, and tracked changes are published in the WebHelp output. The default value is no.

webhelp.sitemap.base.url

Base URL for all the `loc` elements in the generated `sitemap.xml` file. The value of a `loc` element is computed as the relative file path from the `href` attribute of a `topicref` element from the *DITA map*,

appended to this base URL value. The `loc` element is mandatory in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `sitemap.xml` file is not generated.

webhelp.sitemap.change.frequency

The value of the `changefreq` element in the generated `sitemap.xml` file. The `changefreq` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `changefreq` element is not added in `sitemap.xml`. Allowed values: <empty string> (default), `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, `never`.

webhelp.sitemap.priority

The value of the `priority` element in the generated `sitemap.xml` file. It can be set to any fractional number between 0.0 (least important priority) and 1.0 (most important priority). For example, 0.3, 0.5, or 0.8. The `priority` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `priority` element is not added in `sitemap.xml`.

Related Information:

Customizing WebHelp Classic Systems on page 999
WebHelp Classic with Feedback System on page 992

WebHelp Classic Mobile Output

To publish a *DITA map* as a **WebHelp Classic Mobile** system, follow these steps:

1. Select the **Configure Transformation Scenario(s)** action from the ***DITA Maps Manager*** toolbar.
2. Select the **DITA Map WebHelp Classic Mobile** scenario from the **DITA Map** section.
3. If you want to configure the transformation, click the **Edit** button.

   Step Result: This opens an **Edit scenario** configuration dialog box that allows you to configure various options in the following tabs:

   - *Parameters Tab* - This tab includes numerous parameters that can be set to customize your WebHelp system output. See the *Parameters section* below for details about the most commonly used parameters for WebHelp Responsive transformations.
   - *Filters Tab* - This tab allows you to filter certain content elements from the generated output.
   - *Advanced Tab* - This tab allows you to specify some advanced options for the transformation scenario.
   - *Output Tab* - This tab allows you to configure options that are related to the location where the output is generated.
4. Click **Apply associated** to process the transformation.

   When the **DITA Map WebHelp Classic Mobile** transformation is complete, the output is automatically opened in your default browser.

Parameters for Customizing WebHelp Classic Mobile Output

To customize a transformation scenario, you can edit various parameters, including the following most commonly used ones:

args.default.language

This parameter is used if the language is not detected in the *DITA map*. The default value is `en-us`.

clean.output

Deletes all files from the output folder before the transformation is performed (only no and `yes` values are valid and the default value is no).

fix.external.refs.com.oxygenxml (Only supported when the DITA OT transformation process is started from Oxygen XML Editor)

The DITA Open Toolkit usually has problems processing references that point to locations outside of the directory of the processed *DITA map*. This parameter is used to specify whether or not the application should try to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references. The fix has no impact on your edited DITA content. Allowed values: `true` or `false` (default).

use.stemming

Controls whether or not you want to include stemming search algorithms into the published output (default setting is `false`).

webhelp.copyright

Adds a small copyright text that appears at the end of the **Table of Contents** pane.

webhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

webhelp.favicon

The file path that points to an image to be used as a *favicon* in the WebHelp output.

webhelp.footer.file

Path to an XML file that includes the footer content for your WebHelp output pages. You can use this parameter to integrate social media features (such as widgets for Facebook™, Twitter™, Google Analytics, or Google+™). The file must be well-formed, each widget must be in separate `div` or `span` element, and the code for each `script` element is included in an XML comment (also, the start and end tags for the XML comment must be on a separate line). The following code exert is an example for adding a Facebook™ widget:

```
<div id="facebook">
  <div id="fb-root"/>
  <script>
    <!-- (function(d, s, id) { var js, fjs = d.getElementsByTagName(s)[0];
      if (d.getElementById(id)) return;
      js = d.createElement(s); js.id = id;
      js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
      fjs.parentNode.insertBefore(js, fjs); }
       (document, 'script', 'facebook-jssdk')); -->
    </script>
    <div data-share="true" data-show-faces="true" data-action="like"
       data-layout="standard" class="fb-like"/>
</div>
```

webhelp.footer.include

Specifies whether or not to include footer in each WebHelp page. Possible values: `yes`, no. If set to no, no footer is added to the WebHelp pages. If set to `yes` and the `webhelp.footer.file` parameter has a value, then the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default Oxygen XML Editor footer is inserted in each WebHelp page.

webhelp.google.search.results

A file path that specifies the location of a well-formed XHTML file containing the Google Custom Search Engine element `gcse:searchresults-only`. You can use all supported attributes for this element. It is recommend to set the `linkTarget` attribute to `frm` for frameless (*iframe*) version of WebHelp or to `contentWin` for the frameset version of WebHelp. The default value for this attribute is `_blank` and the search results will be loaded in a new window. If this parameter is not specified, the following code will be used `<gcse:searchresults-only linkTarget="frm"></gcse:searchresults-only>`

webhelp.google.search.script

A file path that specifies the location of a well-formed XHTML file containing the Custom Search Engine script from Google.

webhelp.head.script

URL value that specifies the location of a well-formed XHTML file containing the custom script that will be copied in the `<head>` section of every WebHelp page.

webhelp.reload.stylesheet

Set this parameter to `true` if you have out of memory problems when generating WebHelp. It will increase processing time but decrease the memory footprint. The default value is `false`.

webhelp.search.custom.excludes.file

The path of the file that contains name patterns for HTML files that should not be indexed by the WebHelp search engine. Each exclude pattern must be on a new line. The patterns are considered to be relative to the output directory, and they accept wildcards such as `'*'` (matches zero or more characters) or `'?'` (matches one character). For more information about the patterns, see *https://ant.apache.org/manual/dirtasks.html#patterns*.

webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Editor uses for indexing Japanese content in the WebHelp pages.

webhelp.sitemap.base.url

Base URL for all the `loc` elements in the generated `sitemap.xml` file. The value of a `loc` element is computed as the relative file path from the `href` attribute of a `topicref` element from the *DITA map*, appended to this base URL value. The `loc` element is mandatory in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `sitemap.xml` file is not generated.

webhelp.sitemap.change.frequency

The value of the `changefreq` element in the generated `sitemap.xml` file. The `changefreq` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `changefreq` element is not added in `sitemap.xml`. Allowed values: <empty string> (default), `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, `never`.

webhelp.sitemap.priority

The value of the `priority` element in the generated `sitemap.xml` file. It can be set to any fractional number between 0.0 (least important priority) and 1.0 (most important priority). For example, 0.3, 0.5, or 0.8. The `priority` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `priority` element is not added in `sitemap.xml`.

Related Information:

Customizing WebHelp Classic Mobile Systems on page 1024

WebHelp Classic Mobile System (Deprecated) on page 1023

DITA Map to PDF Output Customization

Oxygen XML Editor comes bundled with the DITA Open Toolkit that provides a mechanism for converting *DITA maps* to PDF output. There are numerous ways that you can customize the PDF output and the topics in this section discuss some of those possibilities.

Creating a DITA Map to PDF Transformation Scenario

To create a *DITA Map to PDF* transformation scenario, follow these steps:

1. Click the **Configure Transformation Scenario(s)** button from the ***DITA Maps Manager*** toolbar.
2. Select **DITA Map PDF** and click the **Edit** button (or use the **Duplicate** button if your *framework* is read-only).
3. Use the various tabs to configure the transformation scenario. In the **Parameters** tab, you can use a variety of parameters to customize the output. For example, the following parameters are just a few of the most commonly used ones:
   - `show.changes.and.comments` - If set to `yes`, user comments, replies to comments, and *tracked changes* are published in the PDF output.
   - `customization.dir` - Specifies the path to a customization directory.
4. Click **OK** and then the **Apply Associated** button to run the transformation scenario.

Creating a Customization Directory for PDF Output

DITA Open Toolkit PDF output can be customized in several ways:

- Creating a DITA Open Toolkit plugin that adds extensions to the PDF plugin. More details can be found in the *DITA Open Toolkit Documentation*.
- Creating a customization directory and using it from the PDF transformation scenario. A small example of this procedure can be found below.

How to Create a Customization Directory for PDF Output

The following procedure explains how to do a basic customization of the PDF output by setting up a customization directory. An example of a common use case is embedding a company logo image in the front matter of the book.

1. Copy the entire directory: *DITA-OT-DIR*`\plugins\org.dita.pdf2\Customization` to another location (for instance, `C:\Customization`).
2. Copy your logo image to: `C:\Customization\common\artwork\logo.png`.
3. Rename `C:\Customization\catalog.xml.orig` to: `C:\Customization\catalog.xml`.
4. Open the `catalog.xml` in Oxygen XML Editor and *uncomment* this line:

```
<!--uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/-->
```

   It now looks like this:

```
<uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/>
```

5. Rename the file: `C:\Customization\fo\xsl\custom.xsl.orig` to: `C:\Customization\fo\xsl\custom.xsl`
6. Open the `custom.xsl` file in Oxygen XML Editor and create the template called **createFrontCoverContents** for DITA-OT 2.4.4 (or createFrontMatter_1.0 for DITA-OT 1.8.5).

   Tip: You can copy the same template from *DITA-OT-DIR*`\plugins\org.dita.pdf2\xsl\fo\front-matter.xsl` and modify it in whatever way necessary to achieve your specific goal. This new template in the `custom.xsl` file will override the same template from *DITA-OT-DIR*`\plugins\org.dita.pdf2\xsl\fo\front-matter.xsl`.

   DITA OT 2.4.4 Example:

   For example, if you are using DITA OT 2.4.4, the `custom.xsl` could look like this:

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www.w3.org/1999/XSL/Format"
    version="2.0">

<xsl:template name="createFrontCoverContents">
 <!-- set the title -->
 <fo:block xsl:use-attribute-sets="__frontmatter__title">
  <xsl:choose>
   <xsl:when test="$map/*[contains(@class,' topic/title ')][1]">
    <xsl:apply-templates select="$map/*[contains(@class,' topic/title ')][1]"/>
      </xsl:when>
      <xsl:when test="$map//*[contains(@class,' bookmap/mainbooktitle ')][1]">
        <xsl:apply-templates select="$map//*[contains
                                        (@class,' bookmap/mainbooktitle ')][1]"/>
      </xsl:when>
      <xsl:when test="//*[contains(@class, ' map/map ')]/@title">
        <xsl:value-of select="//*[contains(@class, ' map/map ')]/@title"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="/descendant::*[contains
           (@class, ' topic/topic ')][1]/*[contains(@class, ' topic/title ')]"/>
   </xsl:otherwise>
  </xsl:choose>
 </fo:block>

 <!-- set the subtitle -->
 <xsl:apply-templates select="$map//*[contains
                                        (@class,' bookmap/booktitlealt ')]"/>
 <fo:block xsl:use-attribute-sets="__frontmatter__owner">
  <xsl:apply-templates select="$map//*[contains(@class,' bookmap/bookmeta ')]"/>
 </fo:block>

 <!-- Load the image logo -->
  <fo:block text-align="center" width="100%">
   <fo:external-graphic
      src="url({concat($artworkPrefix,
                        '/Customization/OpenTopic/common/artwork/logo.png')})"
    />
  </fo:block>
 </xsl:template>

</xsl:stylesheet>
```

   DITA OT 1.8.5 Example:

   For example, if you are using DITA OT 1.8.5, the `custom.xsl` could look like this:

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www.w3.org/1999/XSL/Format"
    version="1.1">

<xsl:template name="createFrontMatter_1.0">
   <fo:page-sequence master-reference="front-matter"
```

```
                        xsl:use-attribute-sets="__force__page__count">
     <xsl:call-template name="insertFrontMatterStaticContents"/>
       <fo:flow flow-name="xsl-region-body">
         <fo:block xsl:use-attribute-sets="__frontmatter">
     <!-- set the title -->
           <fo:block xsl:use-attribute-sets="__frontmatter__title">
            <xsl:choose>
             <xsl:when test="$map/*[contains(@class,' topic/title ')][1]">
     <xsl:apply-templates select="$map/*[contains(@class,' topic/title ')][1]"/>
                  </xsl:when>
    <xsl:when test="$map//*[contains(@class,' bookmap/mainbooktitle ')][1]">
     <xsl:apply-templates select="$map//*[contains
                                  (@class,' bookmap/mainbooktitle ')][1]"/>
         </xsl:when>
    <xsl:when test="//*[contains(@class, ' map/map ')]/@title">
      <xsl:value-of select="//*[contains(@class, ' map/map ')]/@title"/>
         </xsl:when>
      <xsl:otherwise>
    <xsl:value-of select="/descendant::*[contains
           (@class, ' topic/topic ')][1]/*[contains(@class, ' topic/title ')]"/>
      </xsl:otherwise>
         </xsl:choose>
     </fo:block>

  <!-- set the subtitle -->
     <xsl:apply-templates select="$map//*[contains
                                        (@class,' bookmap/booktitlealt ')]"/>

        <fo:block xsl:use-attribute-sets="__frontmatter__owner">
            <xsl:apply-templates select="$map//*[contains
                                              (@class,' bookmap/bookmeta ')]"/>
        </fo:block>

        <fo:block text-align="center" width="100%">
           <fo:external-graphic src="url({concat($artworkPrefix,
                        '/Customization/OpenTopic/common/artwork/logo.png')})"/>
         </fo:block>

    </fo:block>

   <!--<xsl:call-template name="createPreface"/>-->

            </fo:flow>
        </fo:page-sequence>
        <xsl:call-template name="createNotices"/>
    </xsl:template>
</xsl:stylesheet>
```

7. Edit the **DITA Map PDF** transformation scenario and in the **Parameters** tab, set the customization.dir parameter to `C:\Customization`.

Tip: For other specific examples, see *DITA-OT 2.3 Documentation - PDF Customization Plugin*.

Related Information:

Automatic PDF plugin customization generator by Jarno Elovirta.

DITA-OT 1.8 Documentation - PDF Customization Plugin

DITA-OT 2.3 Documentation - PDF Customization Plugin

Customizing the Header and Footer in PDF Output

The XSLT stylesheet *`DITA-OT-DIR`*`/plugins/org.dita.pdf2/xsl/fo/static-content.xsl` contains templates that output the static header and footers for various parts of the PDF such as the prolog, table of contents, front matter, or body.

The templates for generating a footer for pages in the body are called `insertBodyOddFooter` or `insertBodyEvenFooter`.

These templates get the static content from resource files that depend on the language used for generating the PDF. The default resource file is *`DITA-OT-DIR`*`/plugins/org.dita.pdf2/cfg/common/vars/en.xml`. These resource files contain variables (such as *Body odd footer*) that can be set to specific user values.

Instead of modifying these resource files directly, they can be overwritten with modified versions of the resources in a PDF customization directory as explained in *Creating a Customization Directory for PDF Output*.

Adding a Watermark to PDF Output

To add a watermark to the PDF output of a *DITA map* transformation, create a DITA-OT customization directory and use it from a **DITA Map to PDF** transformation scenario, as in the following procedure:

1. Copy the entire directory: *`DITA-OT-DIR`*`\plugins\org.dita.pdf2\Customization` to another location (for instance, `C:\Customization`).

2. Copy your watermark image (for example, `watermark.png`) to: `C:\Customization\common\artwork\watermark.png`.
3. Rename the `C:\Customization\catalog.xml.orig` file to: `C:\Customization\catalog.xml`.
4. Open the `catalog.xml` in Oxygen XML Editor and *uncomment* this line:

```
<!--uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/-->
```

   The uncommented line should look like this:

```
<uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/>
```

5. Rename the file: `C:\Customization\fo\xsl\custom.xsl.orig` to: `C:\Customization\fo\xsl\custom.xsl`
6. Open the `C:\Customization\fo\xsl\custom.xsl` file in Oxygen XML Editor to overwrite two XSLT templates:
   - The first template is located in the XSLT stylesheet *`DITA-OT-DIR`*`\plugins\org.dita.pdf2\xsl\fo\static-content.xsl` and we override it specifying a watermark image for every page in the PDF content, using a *block-container* element that references the watermark image file:

```
<fo:static-content flow-name="odd-body-header">
       <fo:block-container absolute-position="absolute"
          top="-2cm" left="-3cm" width="21cm" height="29.7cm"
          background-image="{concat($artworkPrefix,
'/Customization/OpenTopic/common/artwork/watermark.png')}">
         <fo:block/>
       </fo:block-container>
      <fo:block xsl:use-attribute-sets="__body__odd__header">
          <xsl:call-template name="insertVariable">
              <xsl:with-param name="theVariableID" select="'Body odd header'"/>
              <xsl:with-param name="theParameters">
                 <prodname>
                     <xsl:value-of select="$productName"/>
                 </prodname>
                 <heading>
               <fo:inline xsl:use-attribute-sets="__body__odd__header__heading">
                      <fo:retrieve-marker retrieve-class-name="current-header"/>
               </fo:inline>
                </heading>
               <pagenum>
             <fo:inline xsl:use-attribute-sets="__body__odd__header__pagenum">
                    <fo:page-number/>
             </fo:inline>
           </pagenum>
        </xsl:with-param>
      </xsl:call-template>
    </fo:block>
  </fo:static-content>

</xsl:template>
```

   - The second template that we override is located in the XSLT stylesheet *`DITA-OT-DIR`*`\plugins\org.dita.pdf2\xsl\fo\commons.xsl` and is used for styling the first page of the output. We also override it by copying the original template content in our `custom.xsl` and adding the `block-container` element that references the watermark image file:

```
<xsl:template name="createFrontMatter_1.0">
      <fo:page-sequence master-reference="front-matter"
xsl:use-attribute-sets="__force__page__count">
          <xsl:call-template name="insertFrontMatterStaticContents"/>
          <fo:flow flow-name="xsl-region-body">
              <fo:block-container absolute-position="absolute"
                  top="-2cm" left="-3cm" width="21cm" height="29.7cm"
                  background-image="{concat($artworkPrefix,
'/Customization/OpenTopic/common/artwork/watermark.png')}">
                  <fo:block/>
              </fo:block-container>
  <fo:block xsl:use-attribute-sets="__frontmatter">
       <!-- set the title -->
    <fo:block xsl:use-attribute-sets="__frontmatter__title">
     <xsl:choose>
      <xsl:when test="$map/*[contains(@class,' topic/title ')][1]">
    <xsl:apply-templates select="$map/*[contains(@class,' topic/title ')][1]"/>
      </xsl:when>
      <xsl:when test="$map//*[contains(@class,' bookmap/mainbooktitle ')][1]">
    <xsl:apply-templates select="$map//*[contains
(@class,' bookmap/mainbooktitle ')][1]"/>
          </xsl:when>
          <xsl:when test="//*[contains(@class, ' map/map ')]/@title">
              <xsl:value-of select="//*[contains(@class, ' map/map ')]/@title"/>
          </xsl:when>
            <xsl:otherwise>
```

```
              <xsl:value-of select="/descendant::*[contains
(@class, ' topic/topic ')][1]/*[contains(@class, ' topic/title ')]"/>
             </xsl:otherwise>
      </xsl:choose>
    </fo:block>

   <!-- set the subtitle -->
   <xsl:apply-templates select="$map//*[contains
(@class,' bookmap/booktitlealt ')]"/>

      <fo:block xsl:use-attribute-sets="__frontmatter__owner">
         <xsl:apply-templates select="$map//*[contains
(@class,' bookmap/bookmeta ')]"/>
      </fo:block>

      </fo:block>

   <!--<xsl:call-template name="createPreface"/>-->

      </fo:flow>
        </fo:page-sequence>
        <xsl:if test="not($retain-bookmap-order)">
            <xsl:call-template name="createNotices"/>
        </xsl:if>
    </xsl:template>
```

7. Edit your **DITA Map PDF** transformation scenario. In the **Parameters** tab, set the customization.dir parameter to `C:\Customization`.

Related Information:

Adding a Watermark in DITA Map to XHTML Output on page 1697

Force Page Breaks Between Two Block Elements in PDF Output

Suppose that in your DITA content you have two *block elements*, such as two paragraphs:

```
<p>First para</p>
<p>Second para</p>
```

and you want to force a page break between them in the PDF output.

Here is how you can implement a DITA Open Toolkit *plugin* that would achieve this:

1. Define your custom processing instruction that marks the place where a page break should be inserted in the PDF, for example:

```
<p>First para</p>
  <?pagebreak?>
<p>Second para</p>
```

2. Locate the **DITA Open Toolkit** distribution and in the `plugins` directory create a new *plugin* folder (for example, *`DITA-OT-DIR`*`/plugins/pdf-page-break`).
3. In this new folder, create a new `plugin.xml` file with the following content:

```
<plugin id="com.yourpackage.pagebreak">
  <feature extension="package.support.name" value="Force Page Break Plugin"/>
  <feature extension="package.support.email" value="support@youremail.com"/>
  <feature extension="package.version"value="1.0.0"/>
  <feature extension="dita.xsl.xslfo" value="pageBreak.xsl" type="file"/>
</plugin>
```

   The most important feature in the *plugin* is that it will add a new XSLT stylesheet to the XSL processing that produces the PDF content.
4. In the same folder, create an XSLT stylesheet named `pageBreak.xsl` with the following content:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                     xmlns:fo="http://www.w3.org/1999/XSL/Format"version="1.0">
  <xsl:template match="processing-instruction('pagebreak')">
    <fo:block break-after="page"/>
  </xsl:template>
</xsl:stylesheet>
```

5. *Install your plugin in the DITA Open Toolkit.*

Customizing `note` Images in PDF

To customize the images that appear next to each type of note in the PDF output, use a *PDF customization folder* with the following procedure:

1. Copy the *DITA-OT-DIR*`/plugins/org.dita.pdf2/cfg/common/vars/en.xml` file to the `[CUSTOMIZATION_DIR]\common\vars` folder.
2. Edit the copied `en.xml` file and modify, for example, the path to the image for `<note>` element with the `type` attribute set to `notice` from:

```
<variable id="notice Note Image
  Path">Configuration/OpenTopic/cfg/common/artwork/important.png</variable>
```

   to:

```
<variable id="notice Note Image
  Path">Customization/OpenTopic/common/artwork/notice.gif</variable>
```

3. Add your custom **notice** image to `[CUSTOMIZATION_DIR]\common\artwork\notice.gif`.
4. Edit the **DITA Map PDF** transformation scenario and in the **Parameters** tab set the path for the customization.dir property to point to the customization folder.

Show Comments and Tracked Changes in PDF Output

To include comments and *tracked changes* (stored within your DITA topics) in the PDF output, follow these steps:

1. Click the **Configure Transformation Scenario(s)** button from the ***DITA Maps Manager*** toolbar.
2. Select **DITA Map PDF** and click the **Edit** button (or use the **Duplicate** button if your *framework* is read-only).
3. In the **Parameters** tab, set the value of the show.changes.and.comments parameter to `yes`.
4. Click **OK** and then the **Apply Associated** button to run the transformation scenario.

   Result: Comment threads and tracked changes will now appear in the PDF output. Details about each comment or change will be available in the footer section for each page.

Set a Font for PDF Output Generated with FO Processor

When a *DITA map* is transformed to PDF using an FO processor and it contains some Unicode characters that cannot be rendered by the default PDF fonts, a font that is capable of rendering these characters must be configured and embedded in the PDF result.

The settings that must be modified for configuring a font for the built-in FO processor are detailed in *Add a Font to the Built-in FO Processor*.

DITA OT PDF Font Mapping

The DITA OT contains a file *DITA-OT-DIR*`/plugins/org.dita.pdf2/cfg/fo/font-mappings.xml` that maps logical fonts used in the XSLT stylesheets to physical fonts that will be used by the FO processor to generate the PDF output.

The XSLT stylesheets used to generate the XSL-FO output contain code like this:

```
<xsl:attribute name="font-family">monospace</xsl:attribute>
```

The font-family is defined to be *monospace*, but *monospace* is just an alias. It is not a physical font name. Therefore, another stage in the PDF generation takes this *monospace* alias and looks in the `font-mappings.xml`.

If it finds a mapping like this:

```
<aliases>
      <alias name="monospace">Monospaced</alias>
 </aliases>
```

then it looks to see if the *Monospaced* has a *logical-font* definition and if so, it will use the *physical-font* specified there:

```
<logical-font name="Monospaced">
      <physical-font char-set="default">
        <font-face>Courier New, Courier</font-face>
      </physical-font>
...........
</logical-font>
```

Important:

If no alias mapping is found for a font-family specified in the XSLT stylesheets, the processing defaults to **Helvetica**.

Related Information:

DITA Map to PDF WYSIWYG Transformation

Oxygen XML Editor comes bundled with a DITA OT plugin that allows you to convert *DITA maps* to PDF using a CSS layout processor. Oxygen XML Editor also comes bundled with a built-in CSS-based PDF processing engine called **Chemistry**. For those who are familiar with CSS, this makes it very easy to style and customize the PDF output of your DITA projects without having to work with *xsl:fo* customizations.

Oxygen XML Editor supports the following processors (not included in the Oxygen XML Editor installation kit):

- **Oxygen Chemistry** - A built-in processor that is bundled with Oxygen XML Editor.
- **Prince Print with CSS** - A third-party component that needs to be purchased from *http://www.princexml.com*.
- **Antenna House Formatter** - A third-party component that needs to be purchased from *http://www.antennahouse.com/antenna1/formatter/*.

The DITA-OT plugin is located in the following directory: *`DITA-OT-DIR`*`/plugins/com.oxygenxml.pdf.css`.

Although it includes a set of CSS files in its `css` subfolder, when this plugin is used in Oxygen XML Editor, the CSS files located in the `${frameworks}` directory take precedence.

Creating the Transformation Scenario

To create an **DITA Map to PDF WYSIWYG** transformation scenario, follow these steps:

1. Click the **Configure Transformation Scenario(s)** button from the ***DITA Maps Manager*** toolbar.
2. Select **DITA Map PDF - WYSIWYG**.
3. In the **Parameters** tab, configure the following parameters:
   - `css.processor.type` (if you want to use the built-in **Oxygen Chemistry** processor) - Set the value as `chemistry`.
   - `css.processor.path.prince` (if you are using the **Prince Print with CSS** processor) - Specifies the path to the Prince executable file that will be run to produce the PDF. If you installed Prince using its default settings, you can leave this blank.
   - `css.processor.path.antenna-house` (if you are using the **Antenna House Formatter** processor) - Specifies the path to the Antenna House executable file that will be run to produce the PDF. If you installed Antenna House using its default settings, you can leave this blank.
   - `show.changes.and.comments` - When set to `yes`, user comments, replies to comments, and *tracked changes* are published in the PDF output. The default value is no.
   - `dita.css.list` - Allows you to specify a list of CSS URLs to be used by the PDF processor. The files must have URL syntax and be separated using semicolons. If the value is empty, the current selection from the **Styles** drop-down menu is used.
   - `args.css` - Allows you to specify a list of CSS URLs to be used in addition to those specified in the `dita.css.list` parameter OR in addition to the CSS that is currently selected in the **Styles** drop-down menu. The files must have URL syntax and be separated using semicolons. Also, the `dita.css.list` parameter must be left empty to use these files in addition to the selection in the **Styles** drop-down menu.
4. Click **OK** and run the transformation scenario.

Customizing the Styles (for Output and Editing)

If you need to change the styles in the associated CSS, make sure you install Oxygen XML Editor in a folder where you have full read and write privileges (for instance, your user home directory). This is due to the fact that the installed files are usually placed in a read-only folder (for instance, in Windows, Oxygen XML Editor is installed in the `Program Files` folder where the users do not have change rights).

To change the styles of an element you need to create an additional CSS file that will store the customization rules. Once you have created this file, you need to instruct the editor how to use this additional CSS. This can be done in two ways:

- Create an alternate CSS for the DITA document type:
  1. Follow the procedure for adding an alternate CSS file in *Customizing the Main CSS of a Framework* on page 1237.
  2. Once you have configured your CSS as an additional layer, you can *select it from the **Styles** drop-down menu (on the toolbar)*.
  3. Run the **DITA Map PDF - WYSIWYG** transformation scenario and the customization rules from the additional CSS will be visible in the produced PDF.

  This method allows you to have many customization CSS files and simply select the one that you need at any time for both the output and rendering **Author** mode while editing.
- Use the `args.css` parameter that allows you to specify a list of CSS URLs to be used in addition to the `dita.css.list` parameter:
  1. Configure a **DITA map to PDF WYSIWYG** transformation scenario, as described in the procedure *above*.
  2. In the **Parameters** tab, specify the path to your custom CSS files in the `args.css` parameter.
  3. Click **OK** and run the transformation scenario.

  This method is appropriate if you just want to apply the styling customization to the output.

Using the CSS Inspector to See Style Associated with an Element

To find the styles associated with an element, follow this procedure:

1. Open a document in the editor and select **Inspect Styles** from the contextual menu. This opens the ***CSS Inspector** view* that shows all the CSS rules that apply to the selected element.
2. Click the particular link for the CSS selector that you need to change and Oxygen XML Editor will open the CSS file and position the cursor at that selector.
3. After you identify the source of the styles you want to modify, copy the CSS rule in your customization CSS file and modify it according to your needs.

   To see the changes in **Author** mode, press **F5** to reload the document.

How to Make Remote Resources Appear in the Output When Using the Prince Processor

If your documentation references external resources (such as images that are stored on a Web-based repository) and your system is behind an HTTP(S) proxy, you may find that they do not appear in the output when using the **Prince Print with CSS** processor. To solve this, follow this procedure:

1. Edit the **`build.xml`** file that is located in *`DITA-OT-DIR`*`/plugins/com.oxygenxml.pdf.css`.
2. There are two instances in the file where a pair of arguments are commented out:

```
<!-- Please remove the comment wrapping the following two arguments
if you are behind a proxy and your documentation refers remote resources,
for example images.

Note: You also need to remove the backslash '\' that appears before the
property name: "http-proxy". That backslash is there because a sequence of
two dashes ('-') is not permited inside a comment.
-->
<!--
  <arg value="-\-http-proxy=${http.proxyHost}:${http.proxyPort}"/>
  <arg value="-\-http-proxy=${https.proxyHost}:${https.proxyPort}"/>
-->
```

3. Remove the comment in both instances.
4. Make sure you also remove the slash that appears before the property name `http-proxy` in each instance.

   Step Result: The arguments should now look like this:

```
<arg value="--http-proxy=${http.proxyHost}:${http.proxyPort}"/>
<arg value="--http-proxy=${https.proxyHost}:${https.proxyPort}"/>
```

5. Save the **`build.xml`** file.
6. Run the *DITA map to PDF WYSIWYG transformation scenario*.

Result: Your external resources should now appear in your output.

Related Information:

Editing a Transformation Scenario on page 925
Configure Transformation Scenario(s) Dialog Box on page 926
Configuring and Managing Multiple CSS Styles on page 1256
CSS Inspector View on page 232

Compiled HTML Help (CHM) Output Format

To perform a *Compiled HTML Help (CHM)* transformation, Oxygen XML Editor needs `Microsoft HTML Help Workshop` to be installed on your computer. Oxygen XML Editor automatically detects `HTML Help Workshop` and uses it.

Note: `HTML Help Workshop` might fail if the files used for transformation contain accents in their names, due to different encodings used when writing the *.hhp* and *.hhc* files. If the transformation fails to produce the CHM output but the *.hhp* (HTML Help Project) file is already generated, you can manually try to build the CHM output using `HTML Help Workshop`.

Changing the Output Encoding

Oxygen XML Editor uses the `htmlhelp.locale` parameter to correctly display specific characters of different languages in the output of the *Compiled HTML Help (CHM)* transformation. By default, the **DITA Map CHM** transformation scenario that comes bundled with Oxygen XML Editor has the `htmlhelp.locale` parameter set to `en-US`.

To customize this parameter, follow this procedure:

1. Use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu.
2. Select the **DITA Map CHM** transformation scenario and click the **Edit** button.
3. In the **Parameter** tab, search for the `htmlhelp.locale` parameter and change its value to the desired language tag.

   Note: The format of the `htmlhelp.locale` parameter is `LL-CC`, where `LL` represents the language code (en, for example) and `CC` represents the country code (US, for example). The language codes are contained in the `ISO 639-1` standard and the country codes are contained in the `ISO 3166-1` standard. For further details about language tags, go to *http://www.rfc-editor.org/rfc/rfc5646.txt*.

Kindle Output Format

Oxygen XML Editor requires KindleGento generate Kindle output from *DITA maps*. To install KindleGen for use by Oxygen XML Editor, follow these steps:

1. Go to *www.amazon.com/kindleformat/kindlegen* and download the zip file that matches your operating system.
2. Unzip the file on your local disk.
3. Start Oxygen XML Editor and open a *DITA map* in the ***DITA Maps Manager*** *view*.
4. Click the **Configure Transformation Scenario(s)** button.
5. Select the **DITA Map Kindle** transformation and press the **Edit** button to edit it.
6. Go to **Parameters** tab and set the **kindlegen.executable** parameter as the path to the *KindleGen* directory.
7. Accept the changes.

Creating New Transformation Scenarios

Defining a transformation scenario is the first step in the process of transforming a document. This section includes information on the types of scenarios that are available in Oxygen XML Editor and how to create each type of transformation.

XML Transformation with XSLT
This type of transformation specifies the transformation parameters and location of an XSLT stylesheet that is applied to the edited XML document. This scenario is useful when you develop an XML document and the XSLT document is in its final form.

To create an **XML transformation with XSLT** scenario, use one of the following methods:

- Use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **XML transformation with XSLT**.
- Use the **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **XML transformation with XSLT**.

  Note: If a scenario is already associated with the edited document, selecting **Apply Transformation Scenario(s)** runs the associated scenario automatically. You can check to see if transformation scenarios are associated with the edited document by hovering your cursor over the **Apply Transformation Scenario** button.
- Go to **Window** > **Show View** and select **Transformation Scenarios** to display *this view*. Click the **New** button and select **XML transformation with XSLT**.

All three methods open the **New Scenario** dialog box.

The upper part of the dialog box allows you to specify the **Name** of the transformation scenario and the following **Storage** options:

- ***Project Options*** - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- ***Global Options*** - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.

XSLT Tab

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **XSLT** tab contains the following options:

XML URL

Specifies the source XML file. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list. You can also use the **Open in editor** button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, then the file is used directly from its remote location.

Note: If the transformer engine is Saxon 9.x and a custom URI resolver is configured in the *advanced Saxon preferences page*, the XML input of the transformation is passed to that URI resolver. If the transformer engine is one of the built-in XSLT 2.0 / 3.0 engines and *the name of an initial template* is specified in the scenario, the **XML URL** field can be empty. The **XML URL** field can also be empty if you use *external XSLT processors*. Otherwise, a value is mandatory in this field.

XSL URL

Specifies the source XSL file that the transformation will use. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list. You can also use the **Open in editor** button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, the file is used directly from its remote location.

Use "xml-stylesheet" declaration

If selected, the scenario applies the stylesheet specified explicitly in the XML document with the `xml-stylesheet` processing instruction. By default, this option is deselected and the transformation applies the XSLT stylesheet that is specified in the **XSL URL** field.

Transformer

This drop-down menu presents all the transformation engines available to Oxygen XML Editor for performing a transformation. These include the built-in engines and *the external engines defined in the Custom Engines preferences page*. The engine you choose is used as the default transformation engine. Also, if an XSLT or XQuery document does not have an associated validation scenario, this transformation engine is used in the validation process (if it provides validation support).

Advanced options

Allows you to configure the *advanced options of the Saxon HE/PE/EE engine* for the current transformation scenario. To configure the same options globally, go to the *Saxon-HE/PE/EE preferences page*. For the current transformation scenario, these **Advanced options** override the options configured in that preferences page.

Parameters

Opens a ***Configure parameters*** *dialog box* that allows you to configure the XSLT parameters used in the current transformation. In this dialog box, you can also configure the parameters for *additional XSLT stylesheets*. If the XSLT transformation engine is custom-defined, you can not use this dialog box to configure the parameters sent to the custom engine. Instead, you can copy all parameters from the dialog box using contextual menu actions and edit the custom XSLT engine to include the necessary parameters in the command line that starts the transformation process.

Extensions

Opens a *dialog box for configuring the XSLT extension JARS or classes* that define extension Java functions or extension XSLT elements used in the transformation.

Additional XSLT stylesheets

Opens a *dialog box for adding XSLT stylesheets* that are applied on the main stylesheet specified in the **XSL URL** field. This is useful when a chain of XSLT stylesheets must be applied to the input XML document.

XSLT Parameters

The global parameters of the XSLT stylesheet used in a transformation scenario can be configured by using the **Parameters** button in the **XSLT** tab of a new or edited transformation scenario dialog box.

The resulting dialog box includes a table that displays all the parameters of the current XSLT stylesheet, all imported and included stylesheets, and all *additional stylesheets*, along with their descriptions and current values. You can also add, edit, and remove parameters, and you can use the **Filter** text box to search for a specific term in the entire parameters collection. Note that edited parameters are displayed with their name in bold.

If the **XPath** column is selected, the parameter value is evaluated as an XPath expression before starting the XSLT transformation.

Example:

For example, you can use expressions such as:

```
doc('test.xml')//entry
//person[@atr='val']
```

Note:

1. The **doc** function solves the argument relative to the XSL stylesheet location. You can use full paths or *editor variables* (such as *${cfdu}* [current file directory]) to specify other locations: `doc('${cfdu}/test.xml')//*`
2. You cannot use XSLT Functions. Only XPath functions are allowed.

Below the table, the following actions are available for managing the parameters:

New

Opens the **Add Parameter** dialog box that allows you to add a new parameter to the list. An *editor variable* can be inserted in the text box using the **Insert Editor Variables** button. If the **Evaluate as XPath** option is selected, the parameter will be evaluated as an XPath expression.

Edit

Opens the **Edit Parameter** dialog box that allows you to edit the selected parameter. An *editor variable* can be inserted in the text box using the **Insert Editor Variables** button. If the **Evaluate as XPath** option is selected, the parameter will be evaluated as an XPath expression.

Unset

Resets the selected parameter to its default value. Available only for edited parameters with set values.

Delete

Removes the selected parameter from the list. It is available only for new parameters that have been added to the list.

The bottom panel presents the following:

- The default value of the parameter selected in the table.
- A description of the parameter, if available.
- The system ID of the stylesheet that declares it.

Related Information:

Editor Variables on page 169

XSLT Extensions

The **Extensions** button is used to specify the *JARS* and classes that contain extension functions called from the XSLT file of the current transformation scenario. You can use the **Add**, **Edit**, and **Remove** buttons to configure the extensions.

Tip: You can specify the path to the resources using wildcards (for example, `${oxygenHome}/lib/*.jar`).

An extension function called from the XSLT file of the current transformation scenario will be searched, in the specified extensions, in the order displayed in this dialog box. To change the order of the items, select the item to be moved and press the **Move up** or **Move down** buttons.

Additional XSLT Stylesheets

Use the **Additional XSLT Stylesheets** button in the **XSLT** tab to display a list of additional XSLT stylesheets to be used in the transformation and you can add files to the list or edit existing entries. The following actions are available:

Add

Adds a stylesheet in the **Additional XSLT stylesheets** list using a file browser dialog box. You can type an *editor variable* in the file name field of the browser dialog box. The name of the stylesheet will be added in the list after the current selection.

Remove

Deletes the selected stylesheet from the **Additional XSLT stylesheets** list.

Open

Opens the selected stylesheet in a separate view.

Up

Moves the selected stylesheet up in the list.

Down

Moves the selected stylesheet down in the list.

Advanced Saxon HE/PE/EE XSLT Transformation Options

The XSLT transformation scenario allows you to configure advanced options that are specific for the Saxon HE (Home Edition), PE (Professional Edition), and EE (Enterprise Edition) engines. They are the same options as *those in the **Saxon HE/PE/EE** preferences page* but they are configured as a specific set of transformation options

for each transformation scenario, while the values set in the preferences page apply as global options. The advanced options configured in a transformation scenario override the *global options* defined in the preferences page.

Saxon-HE/PE/EE Options

The advanced options for Saxon 9.7.0.15 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE) are as follows:

Mode ("-im")

A Saxon-specific option that sets the initial mode for the transformation.

Template ("-it")

A Saxon-specific option that sets the name of the initial XSLT template to be executed.

Use a configuration file ("-config")

Sets a Saxon 9.7.0.15 configuration file that is executed for XSLT transformation and validation processes.

Debugger trace into XPath expressions (applies to debugging sessions)

Instructs the *XSLT Debugger* to *step into* XPath expressions.

Version warnings ("-versmsg")

Warns you when the transformation is applied to an XSLT 1.0 stylesheet.

Line numbering ("-l")

Line numbers where errors occur are included in the output messages.

Expand attributes defaults ("-expand")

Specifies whether or not the attributes defined in the associated DTD or XML Schema are expanded in the output of the transformation you are executing.

DTD validation of the source ("-dtd")

Specifies whether or not the source document will be validated against their associated DTD. You can choose from the following:

- **On** - Requests DTD validation of the source file and of any files read using the `document()` function.
- **Off** - (default setting) Suppresses DTD validation.
- **Recover** - Performs DTD validation but treats the errors as non-fatal.

  Note: Any external DTD is likely to be read even if not used for validation, since DTDs can contain definitions of entities.

Recoverable errors ("-warnings")

Allows you to choose how dynamic errors are handled. The following options can be selected:

- **Recover silently ("silent")** - Continues processing without reporting the error.
- **Recover with warnings ("recover")** - Issues a warning but continues processing.
- **Signal the error and do not attempt recovery ("fatal")** - Issues an error and stops processing.

Strip whitespaces ("-strip")

Allows you to choose how the *strip whitespaces* operation is handled. You can choose one of the following values:

- **All ("all")** - Strips *all* whitespace text nodes from source documents before any further processing, regardless of any `xml:space` attributes in the source document.
- **Ignore ("ignorable")** - Strips all *ignorable* whitespace text nodes from source documents before any further processing, regardless of any `xml:space` attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.
- **None ("none")** - Strips *no* whitespace before further processing.

Optimization level ("-opt")

Allows you to set the optimization level. It is the value is an integer in the range of 0 (no optimization) to 10 (full optimization). This option allows optimization to be suppressed when reducing the compiling time

is important, optimization conflicts with debugging, or optimization causes extension functions with side-effects to behave unpredictably.

Saxon-PE/EE Options

The following advanced options are specific for Saxon 9.7.0.15 Professional Edition (PE) and Enterprise Edition (EE) only:

Register Saxon-CE extension functions and instructions

Registers the Saxon-CE extension functions and instructions when compiling a stylesheet using the Saxon 9.7.0.15 processors.

Note: Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Editor only for developing the Saxon-CE stylesheet, leaving the execution part to a web browser. See more details about *executing such a stylesheet on Saxonica's website*.

Allow calls on extension functions ("-ext")

If selected, the stylesheet is allowed to call external Java functions. This does not affect calls on integrated extension functions, including Saxon and EXSLT extension functions. This option is useful when loading an untrusted stylesheet (such as from a remote site using `http://[URL]`). It ensures that the stylesheet cannot call arbitrary Java methods and thus gain privileged access to resources on your machine.

Enable assertions ("-ea")

In XSLT 3.0, you can use the **xsl:assert** element to make assertions in the form of XPath expressions, causing a dynamic error if the assertion turns out to be false. If this option is selected, XSLT 3.0 `xsl:assert` instructions are enabled. If it is not selected (default), the assertions are ignored.

Saxon-EE Options

The advanced options that are specific for Saxon 9.7.0.15 Enterprise Edition (EE) are as follows:

XML Schema version

Use this option to change the default XML Schema version for this transformation. To change the default XML Schema version globally, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XML Parser** > **XML Schema** and use the ***Default XML Schema version** option*.

Validation of the source file ("-val")

Requests schema-based validation of the source file and of any files read using `document()` or similar functions. It can have the following values:

- **Schema validation ("strict")** - This mode requires an XML Schema and allows for parsing the source documents with strict schema-validation enabled.
- **Lax schema validation ("lax")** - If an XML Schema is provided, this mode allows for parsing the source documents with schema-validation enabled but the validation will not fail if, for example, element declarations are not found.
- **Disable schema validation** - This specifies that the source documents should be parsed with schema-validation disabled.

Validation errors in the result tree treated as warnings ("-outval")

Normally, if validation of result documents is requested, a validation error is fatal. Selecting this option causes such validation failures to be treated as warnings.

Write comments for non-fatal validation errors of the result document

The validation messages for non-fatal errors are written (wherever possible) as a comment in the result document itself.

Generate bytecode ("--generateByteCode:(on|off)")

If you select this option, Saxon-EE attempts to generate Java bytecode for evaluation of parts of a query or stylesheet that are amenable to such an action. For further details regarding this option, go to *http://www.saxonica.com/documentation9.5/index.html#!javadoc*.

Enable streaming mode

Selecting this option will allow an XSLT to run in streaming mode. It is not selected by default.

Other Options

Initializer class

Equivalent to the `-init` Saxon command-line argument. The value is the name of a user-supplied class that implements the `net.sf.saxon.lib.Initializer` interface. This initializer is called during the initialization process, and may be used to set any options required on the configuration programmatically. It is particularly useful for tasks such as registering extension functions, collations, or external object models, especially in Saxon-HE where the option cannot be set via a configuration file. Saxon only calls the initializer when running from the command line, but the same code may be invoked to perform initialization when running user application code.

FO Processor Tab (XSLT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **FO Processor** tab contains the following options:

Perform FO Processing

Specifies whether or not an FO processor is applied (either the built-in Apache FOP engine or an external engine defined in **Preferences**) during the transformation.

Input

Choose between the following options to specify which input file to use:

- **XSLT result as input** - The FO processor is applied to the result of the XSLT transformation that is defined in the **XSLT** tab.
- **XML URL as input** - The FO processor is applied to the input XML file.

Method

The output format of the FO processing. The available options depend on the selected processor type.

Processor

Specifies the FO processor to be used for the transformation. It can be the built-in Apache FOP processor or an *external processor*.

Output Tab (XSLT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Output** tab contains the following options:

Prompt for file

At the end of the transformation, a file browser dialog box is displayed for specifying the path and name of the file that stores the transformation result.

Save As

The path of the file where the result of the transformation is stored. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Open in Browser/System Application

If selected, Oxygen XML Editor automatically opens the result of the transformation in a system application associated with the file type of the result (for example, in Windows *PDF* files are often opened in *Acrobat Reader*).

Note: To set the web browser that is used for displaying HTML/XHTML pages, go to **Options** > **Preferences** > **Global**, and set it in the **Default Internet browser** field.

- **Output file** - When **Open in Browser/System Application** is selected, you can use this button to automatically open the default output file at the end of the transformation.
- **Other location** - When **Open in Browser/System Application** is selected, you can use this option to open the file specified in this field at the end of the transformation. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Open in editor

When this is option is selected, at the end of the transformation, the default output file is opened in a new editor panel with the appropriate built-in editor type (for example, if the result is an XML file it is opened in the built-in XML editor, or if it is an XSL-FO file it is opened with the built-in FO editor).

Show in results view as

You can choose to view the results in one of the following:

- **XML** - If this is selected, Oxygen XML Editor displays the transformation result in an XML viewer panel at the bottom of the application window with *syntax highlighting*.
- **SVG** - If this is selected, Oxygen XML Editor displays the transformation result in an *integrated SVG viewer in the **Results** panel* at the bottom of the application window and renders the result as an SVG image.
- **XHTML** - This option is only available if **Open in Browser/System Application** is not selected. If selected, Oxygen XML Editor displays the transformation result in a built-in XHTML browser panel at the bottom of the application window.

  Important: When transforming very large documents, you should be aware that selecting this option may result in very long processing times. This drawback is due to the built-in Java XHTML browser implementation. To avoid delays for large documents, if you want to see the XHTML result of the transformation, you should use an external browser by selecting the **Open in Browser/System Application** option instead.

  - **Image URLs are relative to** - If **Show in results view as XHTML** is selected, this option specifies the path used to resolve image paths contained in the transformation result. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

XML Transformation with XQuery

This type of transformation specifies the transform parameters and location of an XQuery file that is applied to the edited XML document.

Use the **XML transformation with XQuery** scenario to apply a transformation in which an XQuery file queries an XML file for the output results.

To create an **XML transformation with XQuery** scenario, use one of the following methods:

- Use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **XML transformation with XQuery**.
- Use the **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **XML transformation with XQuery**.

  Note: If a scenario is already associated with the edited document, selecting **Apply Transformation Scenario(s)** runs the associated scenario automatically. You can check to see if transformation scenarios are associated with the edited document by hovering your cursor over the **Apply Transformation Scenario** button.
- Go to **Window** > **Show View** and select **Transformation Scenarios** to display *this view*. Click the **New** button and select **XML transformation with XQuery**.

All three methods open the **New Scenario** dialog box.

The upper part of the dialog box allows you to specify the **Name** of the transformation scenario and the following **Storage** options:

- ***Project Options*** - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- ***Global Options*** - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.

XQuery Tab

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **XQuery** tab contains the following options:

XML URL

Specifies the source XML file. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list. You can also use the **Open in editor** button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, then the file is used directly from its remote location.

Note: If the transformer engine is Saxon 9.x and a custom URI resolver is configured in the *advanced Saxon preferences page*, the XML input of the transformation is passed to that URI resolver.

XQuery URL

Specifies the source XQuery file to be used for the transformation. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list. You can also use the **Open in editor** button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, the file is used directly from its remote location.

Transformer

This drop-down menu presents all the transformation engines available to Oxygen XML Editor for performing a transformation. These include the built-in engines and *the external engines defined in the Custom Engines preferences page*. The engine you choose is used as the default transformation engine. Also, if an XSLT or XQuery document does not have an associated validation scenario, this transformation engine is used in the validation process (if it provides validation support).

Advanced options

Allows you to configure the *advanced options of the Saxon HE/PE/EE engine* for the current transformation scenario. To configure the same options globally, go to the *Saxon-HE/PE/EE preferences page*. For the current transformation scenario, these **Advanced options** override the options configured in that preferences page.

Parameters

Opens the ***Configure parameters*** *dialog box* for configuring the XQuery parameters. You can use the buttons in this dialog box to add, edit, or remove parameters. If the XQuery transformation engine is custom-defined, you can not use this dialog box to set parameters. Instead, you can copy all parameters from the dialog box using contextual menu actions and edit the custom XQuery engine to include the necessary parameters in the command line that starts the transformation process.

Extensions

Opens a *dialog box for configuring the XQuery extension JARS or classes* that define extension Java functions or extension XSLT elements used in the transformation.

XQuery Parameters

The global parameters of the XQuery file used in a transformation scenario can be configured by using the **Parameters** button in the **XQuery** tab.

The resulting dialog box includes a table that displays all the parameters of the current XQuery file, along with their descriptions and current values. You can also add, edit, and remove parameters, and use the **Filter** text box to search for a specific term in the entire parameters collection. Note that edited parameters are displayed with their name in bold.

If the **XPath** column is selected, the parameter value is evaluated as an XPath expression before starting the XQuery transformation.

Example:

For example, you can use expressions such as:

```
doc('test.xml')//entry
//person[@atr='val']
```

Note:

1. The **doc** function solves the argument relative to the XQuery file location. You can use full paths or *editor variables* (such as *${cfdu}* [current file directory]) to specify other locations: `doc('${cfdu}/test.xml')//*`
2. Only XPath functions are allowed.

Below the table, the following actions are available for managing the parameters:

New

Opens the **Add Parameter** dialog box that allows you to add a new parameter to the list. An *editor variable* can be inserted in the text box using the **Insert Editor Variables** button. If the **Evaluate as XPath** option is selected, the parameter will be evaluated as an XPath expression.

Edit

Opens the **Edit Parameter** dialog box that allows you to edit the selected parameter. An *editor variable* can be inserted in the text box using the **Insert Editor Variables** button. If the **Evaluate as XPath** option is selected, the parameter will be evaluated as an XPath expression.

Unset

Resets the selected parameter to its default value. Available only for edited parameters with set values.

Delete

Removes the selected parameter from the list. It is available only for new parameters that have been added to the list.

The bottom panel presents the following:

- The default value of the parameter selected in the table.
- A description of the parameter, if available.
- The system ID of the stylesheet that declares it.

Related Information:

Editor Variables on page 169

XQuery Extensions

The **Extensions** button is used to specify the *JAR* and classes that contain extension functions called from the XQuery file of the current transformation scenario. You can use the **Add**, **Edit**, and **Remove** buttons to configure the extensions.

An extension function called from the XQuery file of the current transformation scenario will be searched, in the specified extensions, in the order displayed in this dialog box. To change the order of the items, select the item to be moved and press the **Move up** or **Move down** buttons.

Advanced Saxon HE/PE/EE XQuery Transformation Options

The XQuery transformation scenario allows you to configure advanced options that are specific for the Saxon HE (Home Edition), PE (Professional Edition), and EE (Enterprise Edition) engines. They are the same options as *those in the **Saxon HE/PE/EE** preferences page* but they are configured as a specific set of transformation options for each transformation scenario, while the values set in the preferences page apply as global options. The advanced options configured in a transformation scenario override the *global options* defined in the preferences page.

The advanced options for Saxon 9.7.0.15 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE) are as follows:

Recoverable errors ("-warnings")

Allows you to choose how dynamic errors are handled. The following options can be selected:

- **Recover silently ("silent")** - Continues processing without reporting the error.

- **Recover with warnings ("recover")** - Issues a warning but continues processing.
- **Signal the error and do not attempt recovery ("fatal")** - Issues an error and stops processing.

Strip whitespaces ("-strip")

Allows you to choose how the *strip whitespaces* operation is handled. You can choose one of the following values:

- **All ("all")** - Strips *all* whitespace text nodes from source documents before any further processing, regardless of any `xml:space` attributes in the source document.
- **Ignore ("ignorable")** - Strips all *ignorable* whitespace text nodes from source documents before any further processing, regardless of any `xml:space` attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.
- **None ("none")** - Strips *no* whitespace before further processing.

Optimization level ("-opt")

Allows you to set the optimization level. It is the value is an integer in the range of 0 (no optimization) to 10 (full optimization). This option allows optimization to be suppressed when reducing the compiling time is important, optimization conflicts with debugging, or optimization causes extension functions with side-effects to behave unpredictably.

Use linked tree model ("-tree:linked")

This option activates the linked tree model.

Enable XQuery 3.0 support ("-qversion:(1.0|3.0)")

If selected (default value), Saxon runs the XQuery transformation with the XQuery 3.0 support.

Initializer class

Equivalent to the `-init` Saxon command-line argument. The value is the name of a user-supplied class that implements the `net.sf.saxon.lib.Initializer` interface. This initializer is called during the initialization process, and may be used to set any options required on the configuration programmatically. It is particularly useful for tasks such as registering extension functions, collations, or external object models, especially in Saxon-HE where the option cannot be set via a configuration file. Saxon only calls the initializer when running from the command line, but the same code may be invoked to perform initialization when running user application code.

The following advanced options are specific for Saxon 9.7.0.15 Professional Edition (PE) and Enterprise Edition (EE) only:

Use a configuration file ("-config")

Sets a Saxon 9.7.0.15 configuration file that is used for XQuery transformation and validation scenarios.

Allow calls on extension functions ("-ext")

If selected, calls on external functions are allowed. Selecting this option is recommended in an environment where untrusted stylesheets may be executed. It also disables user-defined extension elements and the writing of multiple output files, both of which carry similar security risks.

The advanced options that are specific for Saxon 9.7.0.15 Enterprise Edition (EE) are as follows:

Validation of the source file ("-val")

Requests schema-based validation of the source file and of any files read using `document()` or similar functions. It can have the following values:

- **Schema validation ("strict")** - This mode requires an XML Schema and allows for parsing the source documents with strict schema-validation enabled.
- **Lax schema validation ("lax")** - If an XML Schema is provided, this mode allows for parsing the source documents with schema-validation enabled but the validation will not fail if, for example, element declarations are not found.
- **Disable schema validation** - This specifies that the source documents should be parsed with schema-validation disabled.

Validation errors in the result tree treated as warnings ("-outval")

Normally, if validation of result documents is requested, a validation error is fatal. Selecting this option causes such validation failures to be treated as warnings.

Write comments for non-fatal validation errors of the result document

The validation messages for non-fatal errors are written (wherever possible) as a comment in the result document itself.

Generate bytecode ("--generateByteCode:(on|off)")

If you select this option, Saxon-EE attempts to generate Java bytecode for evaluation of parts of a query or stylesheet that are amenable to such an action. For further details regarding this option, go to *http://www.saxonica.com/documentation9.5/index.html#!javadoc*.

Enable XQuery update ("-update:(on|off)")

This option controls whether or not XQuery update syntax is accepted. The default value is off.

Backup files updated by XQuery ("-backup:(on|off)")

If selected, backup versions for any XML files updated with an XQuery Update are generated. This option is available when the **Enable XQuery update** option is selected.

FO Processor Tab (XQuery Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **FO Processor** tab contains the following options:

Perform FO Processing

Specifies whether or not an FO processor is applied (either the built-in Apache FOP engine or an external engine defined in **Preferences**) during the transformation.

Input

Choose between the following options to specify which input file to use:

- **XQuery result as input** - The FO processor is applied to the result of the XQuery transformation that is defined in the **XQuery** tab.
- **XML URL as input** - The FO processor is applied to the input XML file.

Method

The output format of the FO processing. The available options depend on the selected processor type.

Processor

Specifies the FO processor to be used for the transformation. It can be the built-in Apache FOP processor or an *external processor*.

Output Tab (XQuery Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Output** tab contains the following options:

Present as a sequence

Selecting this option will reduce the time necessary to fetch the full results, as it will only fetch the first chunk of the results.

Prompt for file

At the end of the transformation, a file browser dialog box is displayed for specifying the path and name of the file that stores the transformation result.

Save As

The path of the file where the result of the transformation is stored. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Open in Browser/System Application

If selected, Oxygen XML Editor automatically opens the result of the transformation in a system application associated with the file type of the result (for example, in Windows *PDF* files are often opened in *Acrobat Reader*).

Note: To set the web browser that is used for displaying HTML/XHTML pages, go to **Options** > **Preferences** > **Global**, and set it in the **Default Internet browser** field.

- **Output file** - When **Open in Browser/System Application** is selected, you can use this button to automatically open the default output file at the end of the transformation.
- **Other location** - When **Open in Browser/System Application** is selected, you can use this option to open the file specified in this field at the end of the transformation. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Open in editor

When this is option is selected, at the end of the transformation, the default output file is opened in a new editor panel with the appropriate built-in editor type (for example, if the result is an XML file it is opened in the built-in XML editor, or if it is an XSL-FO file it is opened with the built-in FO editor).

Show in results view as

You can choose to view the results in one of the following:

- **XML** - If this is selected, Oxygen XML Editor displays the transformation result in an XML viewer panel at the bottom of the application window with *syntax highlighting*.
- **SVG** - If this is selected, Oxygen XML Editor displays the transformation result in an *integrated SVG viewer in the **Results** panel* at the bottom of the application window and renders the result as an SVG image.
- **XHTML** - This option is only available if **Open in Browser/System Application** is not selected. If selected, Oxygen XML Editor displays the transformation result in a built-in XHTML browser panel at the bottom of the application window.

  Important: When transforming very large documents, you should be aware that selecting this option may result in very long processing times. This drawback is due to the built-in Java XHTML browser implementation. To avoid delays for large documents, if you want to see the XHTML result of the transformation, you should use an external browser by selecting the **Open in Browser/System Application** option instead.

  - **Image URLs are relative to** - If **Show in results view as XHTML** is selected, this option specifies the path used to resolve image paths contained in the transformation result. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

DITA OT Transformation

This type of transformation specifies the parameters for an Ant transformation that executes a DITA-OT build script. Oxygen XML Editor includes a built-in version of Ant and a built-in version of DITA-OT, but other versions can be set in the scenario.

To create a **DITA OT Transformation** scenario, use one of the following methods:

- Use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **DITA OT Transformation**.
- Use the **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **DITA OT Transformation**.

  Note: If a scenario is already associated with the edited document, selecting **Apply Transformation Scenario(s)** runs the associated scenario automatically. You can check to see if transformation scenarios are associated with the edited document by hovering your cursor over the **Apply Transformation Scenario** button.
- Go to **Window** > **Show View** and select **Transformation Scenarios** to display *this view*. Click the **New** button and select **DITA OT Transformation**.

All three methods open the **DITA Transformation Type** dialog box that presents the list of possible outputs.

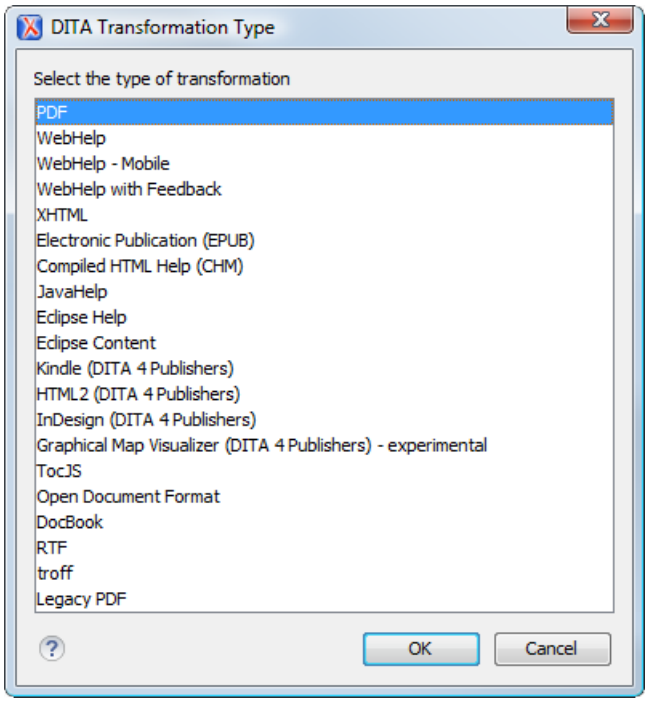


Figure 463: DITA Transformation Type Dialog Box

Select the desired type of output and click **OK**. This opens the **New Scenario** dialog box.

The upper part of the dialog box allows you to specify the **Name** of the transformation scenario and the following **Storage** options:

- ***Project Options*** - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- ***Global Options*** - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation. Some of these tabs are only available for certain output types (for example, a **Skins** tab is only available for **WebHelp Classic** and **WebHelp Classic with Feedback** output types, a **Templates** tab is available only for **WebHelp Responsive** and **WebHelp Responsive with Feedback**, and a **FO Processor** tab is available for PDF output).

Skins Tab (DITA OT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Skins** tab is available for DITA OT transformations with **WebHelp Classic** or **WebHelp Classic with Feedback** output types and it provides a set of predefined skins that you can use as a base for your WebHelp system output.

A *skin* is a collection of CSS properties that can alter the look of the output by changing colors, font types, borders, margins, and paddings. This allows you to rapidly adapt the look and feel of your output.

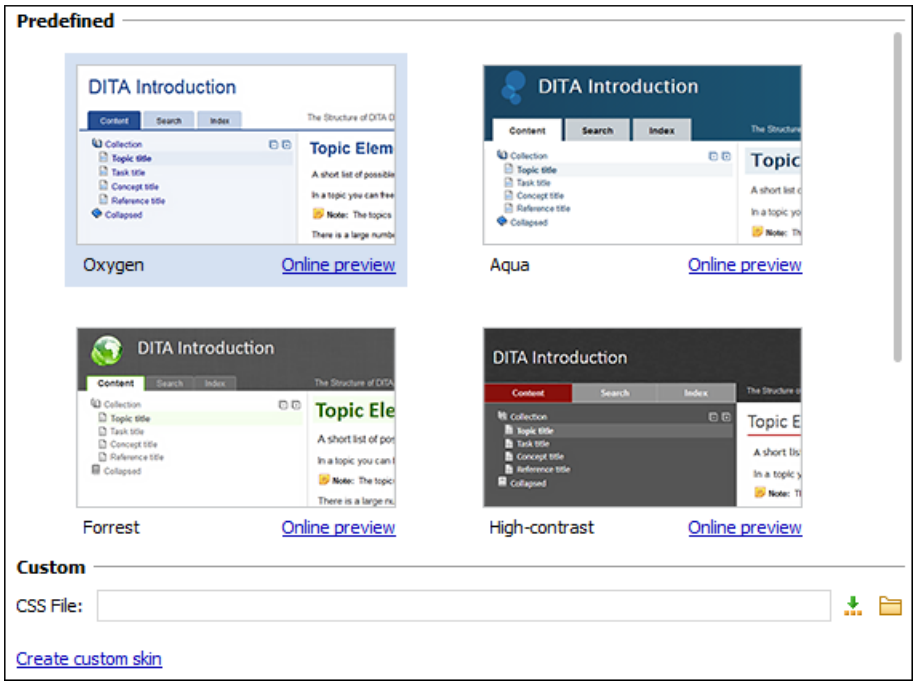


Figure 464: Skins Tab

The **Skins** tab includes the following sections:

Predefined Skins

This sections presents the predefined skins that are included in Oxygen XML Editor. The predefined skins cover a wide range of chromatic themes, ranging from a very light one to a high-contrast variant. To see how the *skin* looks when applied on a sample documentation project that is stored on the Oxygen XML Editor website, press the **Online preview** link.

Custom Skins

You can use this section to customize the look of the output.

CSS File

You can set this field to point to a custom CSS stylesheet or customized skin. A custom CSS file will overwrite a skin selection.

Note: The output can also be styled by setting the `args.css` parameter in the **Parameters tab**. The properties taken from the stylesheet referenced in this parameter take precedence over the properties declared in the skin set in the **Skins tab**.

Create custom skin

Use this link to open the ***WebHelp Skin Builder*** tool.

Templates Tab (DITA OT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Templates** tab is available for DITA OT transformations with **WebHelp Responsive** or **WebHelp Responsive with Feedback** output types and it provides a set of predefined *skins* that you can use as a base for the layout of your WebHelp system output.

A *skin* is a collection of CSS properties that can alter the look of the output by changing colors, font types, borders, margins, and paddings. This allows you to rapidly adapt the look and feel of your output. You can choose predefined skins in a *tile* style of layout or a *tree* style of layout, and you can also *add your own customized skins*.

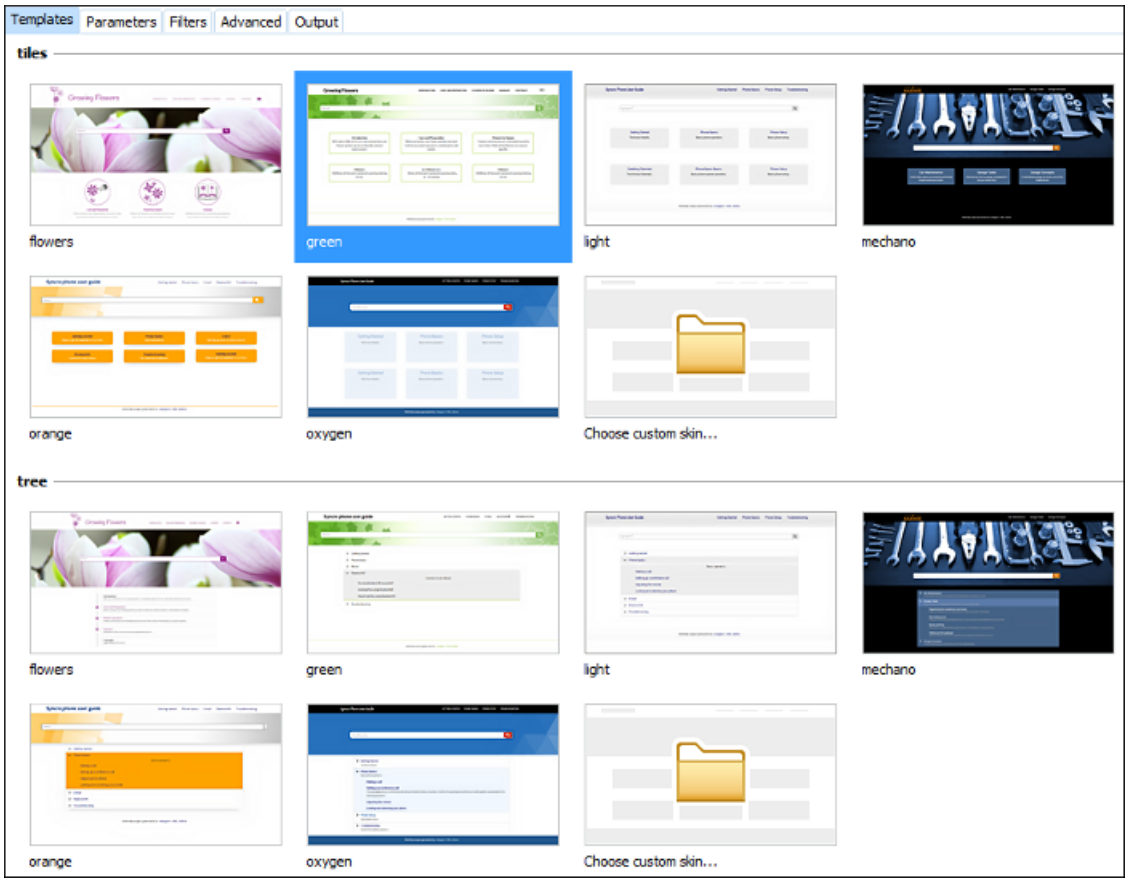


Figure 465: Templates Tab

The **Templates** tab comes by default with the following predefined collections of skins:

Tiles

This sections presents the predefined skins that are arranged in a *tiles* style of layout. These predefined skins include a variety of themes, ranging from a very light one to a high-contrast variant, and various styles. If you select **Choose custom skin**, you can select a custom CSS stylesheet to be used as your template.

Tree

This sections presents the predefined skins that are arranged in a *tree* style of layout. These predefined skins include a variety of themes, ranging from a very light one to a high-contrast variant, and various styles. If you select **Choose custom skin**, you can select a custom CSS stylesheet to be used as your template.

When you add a new collection of skins, this tab will list them.

FO Processor Tab (DITA OT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **FO Processor** tab is available for DITA OT transformations with a **PDF** output type.

This tab allows you to select an FO Processor to be used for the transformation.

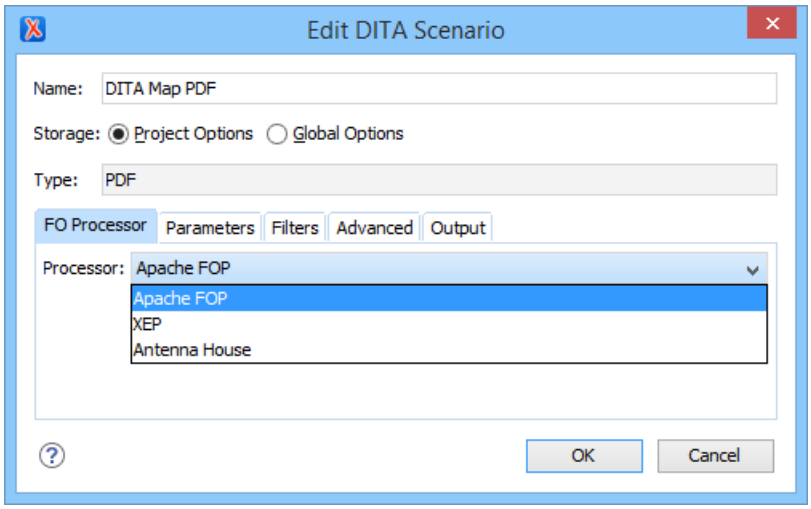


Figure 466: FO Processor Configuration Tab

You can choose one of the following processors:

Apache FOP

The default processor that comes bundled with Oxygen XML Editor.

XEP

The *RenderX* XEP processor. If XEP is already installed, Oxygen XML Editor displays the detected installation path under the drop-down menu. XEP is considered installed if it was detected in one of the following sources:

- XEP was configured as an external FO Processor in the ***FO Processors** option page*.
- The system property *com.oxygenxml.xep.location* was set to point to the XEP executable file for the platform (for example: `xep.bat` on Windows).
- XEP was installed in the *`DITA-OT-DIR`*`/plugins/org.dita.pdf2/lib` directory of the Oxygen XML Editor installation directory.

Antenna House

The *Antenna House* (AH Formatter) processor. If Antenna House is already installed, Oxygen XML Editordisplays the detected installation path under the drop-down menu. Antenna House is considered installed if it was detected in one of the following sources:

- Environment variable set by Antenna House installation (the newest installation version will be used).
- Antenna House was added as an external FO Processor in the Oxygen XML Editor preferences pages.

To further customize the PDF output obtained from the Antenna House processor, follow these steps:

1. **Edit** the transformation scenario.
2. Open the ***Parameters** tab*.
3. Add the `env.AXF_OPT` parameter and point to the Antenna House configuration file.

Related Information:

FO Processors Preferences on page 129

XSL-FO Processors

Parameters Tab (DITA OT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Parameters** tab allows you to configure the parameters sent to the DITA-OT build file.

The table in this tab displays all the parameters that the DITA-OT documentation specifies as available for each chosen type of transformation (for example, XHTML or PDF), along with their description and current values. You can find more information about each parameter in the *DITA OT Documentation*. You can also add, edit, and

remove parameters, and you can use the text box to filter or search for a specific term in the entire parameters collection. Note that edited parameters are displayed with their name in bold.

Depending on the type of a parameter, its value can be one of the following:

- A simple text field for simple parameter values.
- A combo box with some predefined values.
- A file chooser and an *editor variable* selector to simplify setting a file path as the value of a parameter.

Note: To input parameter values at runtime, use the *ask editor variable* in the **Value** column.

Below the table, the following actions are available for managing parameters:

New

Opens the **Add Parameter** dialog box that allows you to add a new parameter to the list. You can specify the **Value** of the parameter by using the ***Insert Editor Variables*** button or the **Browse** button.

Unset

Resets the selected parameter to its default value. Available only for edited parameters with set values.

Edit

Opens the **Edit Parameter** dialog box that allows you to change the value of the selected parameter or its description.

Delete

Removes the selected parameter from the list. It is available only for new parameters that have been added to the list.

Related Information:

DITA Open Toolkit Documentation

Filters Tab (DITA Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Filters** tab allows you to add filters to remove certain content elements from the generated output.

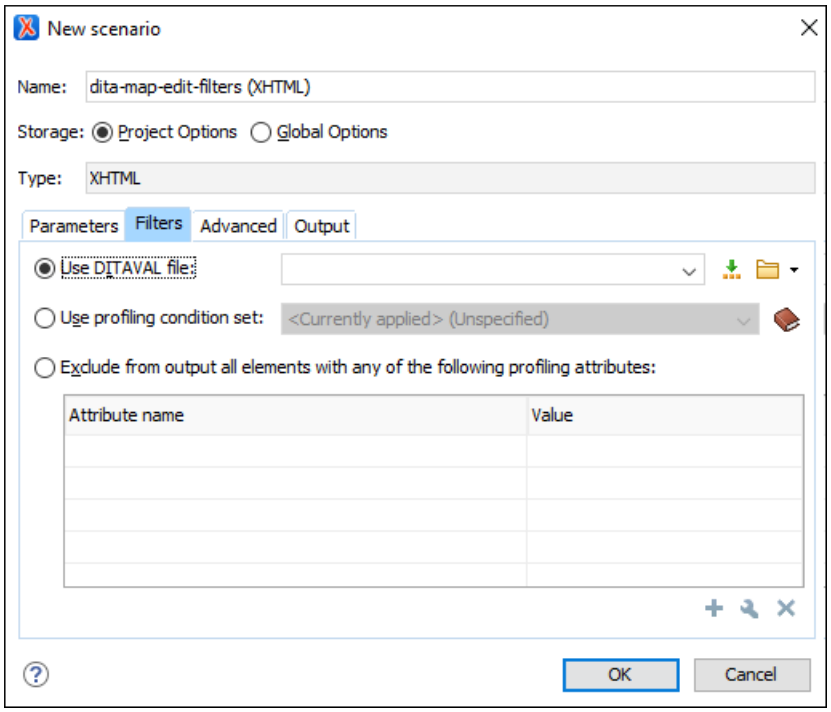


Figure 467: Edit Filters Tab

You can choose one of the following options to define filters:

Use DITAVAL file

If you already have a *DITAVAL* file associated with the *DITA map*, you can specify the file to be used when filtering content. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list. You can find out more about constructing a *DITAVAL* file in the *DITA Documentation*.

Attention: If a filter file is specified in the `args.filter` parameter (in *the **Parameters** tab*), that file takes precedence over a *DITAVAL* file specified here.

Use profiling condition set

Sets the *profiling condition set* that will be applied to your transformation.

Exclude from output all elements with any of the following attributes

By using the **New**, **Edit**, or **Delete** buttons at the bottom of the pane, you can configure a list of attributes (name and value) to exclude all elements that contain any of these attributes from the output.

Advanced Tab (DITA OT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Advanced** tab allows you to specify advanced options for the transformation scenario.

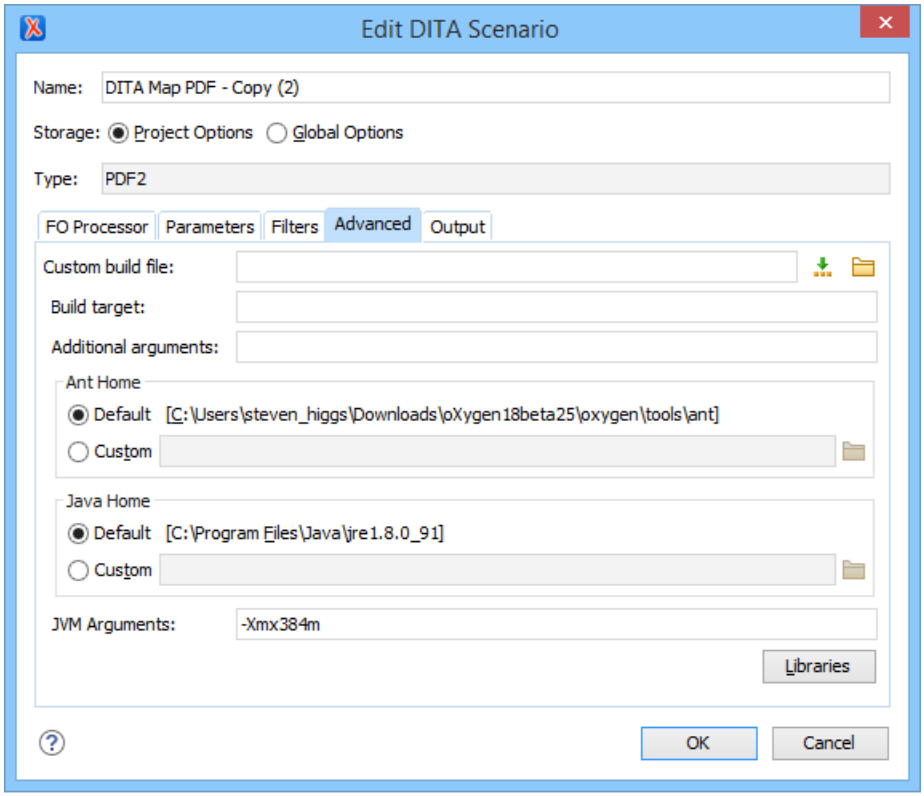


Figure 468: Advanced Settings Tab

You can specify the following parameters:

Custom build file

If you use a custom DITA-OT build file, you can specify the path to the customized build file. If empty, the `build.xml` file from the *dita.dir* parameter that is configured in the ***Parameters** tab* is used. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Build target

Optionally, you can specify a build target for the build file. If no target is specified, the default `init` target is used.

Additional arguments

You can specify additional command line arguments to be passed to the transformation (such as -verbose).

Ant Home

You can choose between the default or custom Ant installation to run the transformation. The default path can be configured in the *Ant preferences page*.

Java Home

You can choose between the default or custom Java installation to run the transformation. The default path is the Java installation that is used by Oxygen XML Editor.

Note: It may be possible that the used Java version is incompatible with the DITA Open Toolkit engine. For example, DITA OT 1.8 and older requires Java 1.6 or later, while DITA OT 2.0 and newer requires Java 1.7 or newer. Thus, if you encounter related errors running the transformation, consider installing a Java VM that is supported by the DITA OT publishing engine and using it in the **Java Home** text field.

JVM Arguments

This parameter allows you to set specific parameters for the Java Virtual Machine used by Ant. For example, if it is set to -Xmx384m, the transformation process is allowed to use 384 megabytes of memory. When performing a large transformation, you may want to increase the memory allocated to the Java Virtual Machine. This will help avoid *Out of Memory* error messages (**OutOfMemoryError**).

Libraries

By default, Oxygen XML Editor adds libraries (as high priority) that are not transformation-dependent and also patches for certain DITA Open Toolkit bugs. You can use this button to specify additional libraries (*JAR* files or additional class paths) to be used by the Ant transformer.

Tip: You can specify the path to the additional libraries using wildcards (for example, `${oxygenHome}/lib/*.jar`).

Output Tab (DITA OT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Output** tab allows you to configure options that are related to the location where the output is generated.

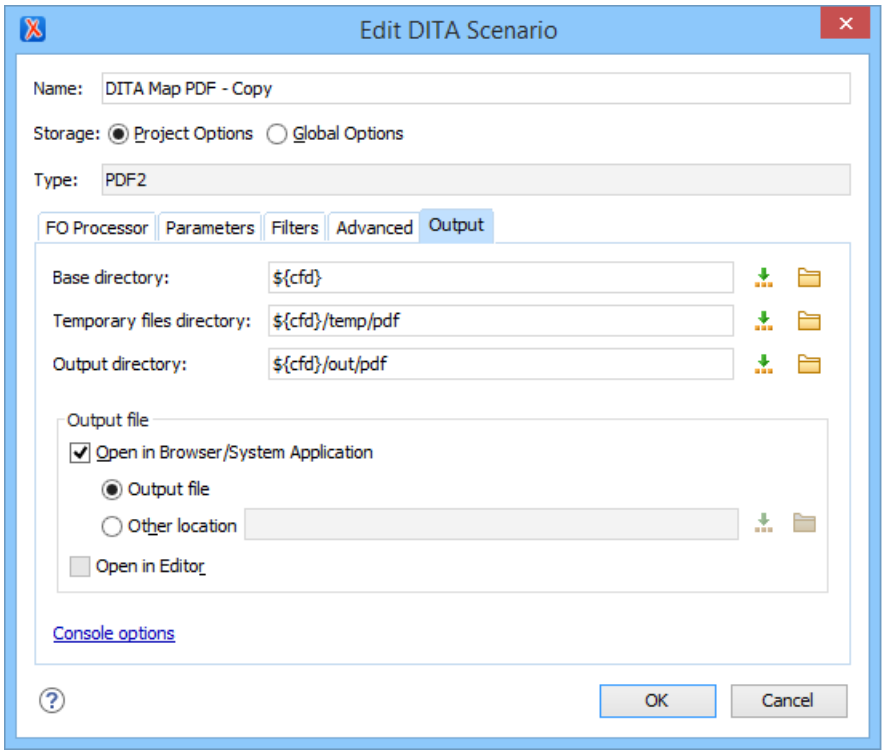


Figure 469: Output Settings Tab

You can specify the following parameters:

Base directory

All the relative paths that appear as values in parameters are considered relative to the base directory. The default value is the directory where the transformed map is located. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Temporary files directory

This directory is used to store pre-processed temporary files until the final output is obtained. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Output directory

The folder where the content of the final output is stored. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Note: If the *DITA map* or topic is opened from a remote location or a ZIP file, the parameters must specify absolute paths.

Open in Browser/System Application

If selected, Oxygen XML Editor automatically opens the result of the transformation in a system application associated with the file type of the result (for example, in Windows *PDF* files are often opened in *Acrobat Reader*).

Note: To set the web browser that is used for displaying HTML/XHTML pages, go to **Options** > **Preferences** > **Global**, and set it in the **Default Internet browser** field.

- **Output file** - When **Open in Browser/System Application** is selected, you can use this button to automatically open the default output file at the end of the transformation.
- **Other location** - When **Open in Browser/System Application** is selected, you can use this option to open the file specified in this field at the end of the transformation. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Open in editor

When this is option is selected, at the end of the transformation, the default output file is opened in a new editor panel with the appropriate built-in editor type (for example, if the result is an XML file it is opened in the built-in XML editor, or if it is an XSL-FO file it is opened with the built-in FO editor).

At the bottom of the pane there is a link to the *Console options* preferences page that contains options to control the display of the console output received from the publishing engine.

Troubleshooting DITA Transformation Errors

If a DITA transformation results in errors or warnings, the information is displayed in the message panel at the bottom of the editor. The information includes the severity, description of the problem, the name of the resource, and the path of the resource.

To help prevent and solve DITA transformation problems, follow these steps:

1. *Validate the DITA map* by using the **Validate and Check for Completeness** action that is available on the ***DITA Maps Manager*** toolbar and in the **DITA Maps** menu.
2. If this action results in validation errors, solve them prior to executing the transformation. Also, you should pay attention to the warning messages because they may identify problems in the transformation.
3. *Run the DITA transformation scenario*.
4. If the transformation results in errors or warnings, they are displayed in the ***Results** panel* at the bottom of the editor. The following information is presented to help you troubleshoot the problems:
   - **Severity** - The first column displays the following icons that indicate the severity of the problem:
     - **Informational** - The transformation encountered a condition of which you should be aware.
     - **Warning** - The transformation encountered a problem that should be corrected.
     - **Error** - The transformation encountered a more severe problem, and the output is affected or cannot be generated.
   - **Info** - You can click the **See More** icon to open a web page that contains details about DITA-OT error messages.
   - **Description** - A description of the problem.
   - **Resource** - The name of the transformation resource.
   - **System ID** - The path of the transformation resource.
5. Use this information or other resources from the online DITA-OT community to solve the transformation problems before re-executing the transformation scenario.
6. If you need to contact the Oxygen technical support team, they will need you to send the entire transformation scenario execution log. To obtain it:
   a. Go to the **Options** > **Preferences** > **DITA** preferences page and set the **Show console output** option to **Always**.
   b. Execute the transformation scenario again. The console output messages are displayed in the **DITA OT** view.
   c. Copy the entire log, save it in a text file, then send it to the Oxygen technical support team.
   d. After your issue has been solved, go back to the **Options** > **Preferences** > **DITA** preferences page and set the **Show console output** option to **When build fails**.

Ant Transformation

This type of transformation allows you to configure the options and parameters of an Ant build script.

An Ant transformation scenario is usually associated with an Ant build script. Oxygen XML Editor runs an Ant transformation scenario as an external process that executes the Ant build script with the built-in Ant distribution (*Apache Ant* version 1.8.2) that is included with the application, or optionally with a custom Ant distribution configured in the scenario.

To create an Ant transformation scenario, use one of the following methods:

- Use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **ANT transformation**.

- Use the **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **ANT transformation**.

  Note: If a scenario is already associated with the edited document, selecting **Apply Transformation Scenario(s)** runs the associated scenario automatically. You can check to see if transformation scenarios are associated with the edited document by hovering your cursor over the **Apply Transformation Scenario** button.
- Go to **Window** > **Show View** and select **Transformation Scenarios** to display *this view*. Click the **New** button and select **ANT transformation**.

All three methods open the **New Scenario** dialog box.

The upper part of the dialog box allows you to specify the **Name** of the transformation scenario and the following **Storage** options:

- ***Project Options*** - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- ***Global Options*** - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.

Options Tab (Ant Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Options** tab allows you to specify the following options:

Working directory

The path of the current directory of the Ant external process. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Build file

The Ant script file that is the input of the Ant external process. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Build target

Optionally, you can specify a build target for the Ant script file. If no target is specified, the Ant target that is specified as the default in the Ant script file is used.

Additional arguments

You can specify additional command line arguments to be passed to the transformation (such as -verbose).

Ant Home

You can choose between the default or custom Ant installation to run the transformation. The default path can be configured in the *Ant preferences page*.

Java Home

You can choose between the default or custom Java installation to run the transformation. The default path is the Java installation that is used by Oxygen XML Editor.

JVM Arguments

This parameter allows you to set specific parameters for the Java Virtual Machine used by Ant. For example, if it is set to -Xmx384m, the transformation process is allowed to use 384 megabytes of memory. When performing a large transformation, you may want to increase the memory allocated to the Java Virtual Machine. This will help avoid *Out of Memory* error messages (**OutOfMemoryError**).

Libraries

By default, Oxygen XML Editor adds libraries (as high priority) that are not transformation-dependent and also patches for certain DITA Open Toolkit bugs. You can use this button to specify additional libraries (*JAR* files or additional class paths) to be used by the Ant transformer.

Tip: You can specify the path to the additional libraries using wildcards (for example, `${oxygenHome}/lib/*.jar`).

Parameters Tab (Ant Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Parameters** tab allows you to configure the parameters that are accessible as Ant properties in the Ant build script.

The table displays all the parameters that are available in the Ant build script, along with their description and current values. You can also add, edit, and remove parameters, and use the **Filter** text box to search for a specific term in the entire parameters collection. Note that edited parameters are displayed with their name in bold.

Depending on the type of a parameter, its value can be one of the following:

- A simple text field for simple parameter values.
- A combo box with some predefined values.
- A file chooser and an *editor variable* selector to simplify setting a file path as the value of a parameter.

Note: To input parameter values at runtime, use the *ask editor variable* in the **Value** column.

Below the table, the following actions are available for managing parameters:

New

Opens the **Add Parameter** dialog box that allows you to add a new parameter to the list. You can specify the **Value** of the parameter by using the ***Insert Editor Variables*** button or the **Browse** button.

Edit

Opens the **Edit Parameter** dialog box that allows you to change the value of the selected parameter or its description.

Delete

Removes the selected parameter from the list. It is available only for new parameters that have been added to the list.

These parameters are also available for the built-in validation processor and the *Content Completion Assistant*.

Related Information:

Content Completion in Ant Build Files on page 545

Output Tab (Ant Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Output** tab contains the following options:

Open

Allows you to specify the file to open automatically when the transformation is finished. This is usually the output file of the Ant process. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

- **In System Application** - The file specified in the **Open** text box is opened in the system application that is set in the operating system as the default application for that type of file (for example, in Windows *PDF* files are often opened in *Acrobat Reader*).
- **In Editor** - The file specified in the **Open** text box is opened in a new editor panel with the appropriate built-in editor type (for example, if the result is an XML file it is opened in the built-in XML editor).

Show console output

Allows you to specify when to display the console output log. The following options are available:

- **When build fails** - displays the console output log if the build fails.
- **Always** - displays the console output log, regardless of whether or not the build fails.

XSLT Transformation
This type of transformation specifies the parameters and location of an XML document that the edited XSLT stylesheet is applied on. This scenario is useful when you develop an XSLT document and the XML document is in its final form.

To create an **XSLT transformation** scenario, use one of the following methods:

- Use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **XSLT transformation**.
- Use the **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **XSLT transformation**.

  Note: If a scenario is already associated with the edited document, selecting **Apply Transformation Scenario(s)** runs the associated scenario automatically. You can check to see if transformation scenarios are associated with the edited document by hovering your cursor over the **Apply Transformation Scenario** button.
- Go to **Window** > **Show View** and select **Transformation Scenarios** to display *this view*. Click the **New** button and select **XSLT transformation**.

All three methods open the **New Scenario** dialog box.

The upper part of the dialog box allows you to specify the **Name** of the transformation scenario and the following **Storage** options:

- ***Project Options*** - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- ***Global Options*** - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.

XSLT Tab

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **XSLT** tab contains the following options:

XML URL

Specifies the source XML file. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list. You can also use the **Open in editor** button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, then the file is used directly from its remote location.

Note: If the transformer engine is Saxon 9.x and a custom URI resolver is configured in the *advanced Saxon preferences page*, the XML input of the transformation is passed to that URI resolver. If the transformer engine is one of the built-in XSLT 2.0 / 3.0 engines and *the name of an initial template* is specified in the scenario, the **XML URL** field can be empty. The **XML URL** field can also be empty if you use *external XSLT processors*. Otherwise, a value is mandatory in this field.

XSL URL

Specifies the source XSL file that the transformation will use. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list. You can also use the **Open in editor** button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, the file is used directly from its remote location.

Use "xml-stylesheet" declaration

If selected, the scenario applies the stylesheet specified explicitly in the XML document with the `xml-stylesheet` processing instruction. By default, this option is deselected and the transformation applies the XSLT stylesheet that is specified in the **XSL URL** field.

Transformer

This drop-down menu presents all the transformation engines available to Oxygen XML Editor for performing a transformation. These include the built-in engines and *the external engines defined in the Custom Engines preferences page*. The engine you choose is used as the default transformation engine. Also, if an XSLT or XQuery document does not have an associated validation scenario, this transformation engine is used in the validation process (if it provides validation support).

Advanced options

Allows you to configure the *advanced options of the Saxon HE/PE/EE engine* for the current transformation scenario. To configure the same options globally, go to the *Saxon-HE/PE/EE preferences page*. For the current transformation scenario, these **Advanced options** override the options configured in that preferences page.

Parameters

Opens a ***Configure parameters*** *dialog box* that allows you to configure the XSLT parameters used in the current transformation. In this dialog box, you can also configure the parameters for *additional XSLT stylesheets*. If the XSLT transformation engine is custom-defined, you can not use this dialog box to configure the parameters sent to the custom engine. Instead, you can copy all parameters from the dialog box using contextual menu actions and edit the custom XSLT engine to include the necessary parameters in the command line that starts the transformation process.

Extensions

Opens a *dialog box for configuring the XSLT extension JARS or classes* that define extension Java functions or extension XSLT elements used in the transformation.

Additional XSLT stylesheets

Opens a *dialog box for adding XSLT stylesheets* that are applied on the main stylesheet specified in the **XSL URL** field. This is useful when a chain of XSLT stylesheets must be applied to the input XML document.

XSLT Parameters

The global parameters of the XSLT stylesheet used in a transformation scenario can be configured by using the **Parameters** button in the **XSLT** tab of a new or edited transformation scenario dialog box.

The resulting dialog box includes a table that displays all the parameters of the current XSLT stylesheet, all imported and included stylesheets, and all *additional stylesheets*, along with their descriptions and current values. You can also add, edit, and remove parameters, and you can use the **Filter** text box to search for a specific term in the entire parameters collection. Note that edited parameters are displayed with their name in bold.

If the **XPath** column is selected, the parameter value is evaluated as an XPath expression before starting the XSLT transformation.

Example:

For example, you can use expressions such as:

```
doc('test.xml')//entry
//person[@atr='val']
```

Note:

1. The **doc** function solves the argument relative to the XSL stylesheet location. You can use full paths or *editor variables* (such as *${cfdu}* [current file directory]) to specify other locations: `doc('${cfdu}/test.xml')//*`
2. You cannot use XSLT Functions. Only XPath functions are allowed.

Below the table, the following actions are available for managing the parameters:

New

Opens the **Add Parameter** dialog box that allows you to add a new parameter to the list. An *editor variable* can be inserted in the text box using the **Insert Editor Variables** button. If the **Evaluate as XPath** option is selected, the parameter will be evaluated as an XPath expression.

Edit

Opens the **Edit Parameter** dialog box that allows you to edit the selected parameter. An *editor variable* can be inserted in the text box using the **Insert Editor Variables** button. If the **Evaluate as XPath** option is selected, the parameter will be evaluated as an XPath expression.

Unset

Resets the selected parameter to its default value. Available only for edited parameters with set values.

Delete

Removes the selected parameter from the list. It is available only for new parameters that have been added to the list.

The bottom panel presents the following:

- The default value of the parameter selected in the table.
- A description of the parameter, if available.
- The system ID of the stylesheet that declares it.

Related Information:

Editor Variables on page 169

XSLT Extensions

The **Extensions** button is used to specify the *JARS* and classes that contain extension functions called from the XSLT file of the current transformation scenario. You can use the **Add**, **Edit**, and **Remove** buttons to configure the extensions.

Tip: You can specify the path to the resources using wildcards (for example, `${oxygenHome}/lib/*.jar`).

An extension function called from the XSLT file of the current transformation scenario will be searched, in the specified extensions, in the order displayed in this dialog box. To change the order of the items, select the item to be moved and press the **Move up** or **Move down** buttons.

Additional XSLT Stylesheets

Use the **Additional XSLT Stylesheets** button in the **XSLT** tab to display a list of additional XSLT stylesheets to be used in the transformation and you can add files to the list or edit existing entries. The following actions are available:

Add

Adds a stylesheet in the **Additional XSLT stylesheets** list using a file browser dialog box. You can type an *editor variable* in the file name field of the browser dialog box. The name of the stylesheet will be added in the list after the current selection.

Remove

Deletes the selected stylesheet from the **Additional XSLT stylesheets** list.

Open

Opens the selected stylesheet in a separate view.

Up

Moves the selected stylesheet up in the list.

Down

Moves the selected stylesheet down in the list.

Advanced Saxon HE/PE/EE XSLT Transformation Options

The XSLT transformation scenario allows you to configure advanced options that are specific for the Saxon HE (Home Edition), PE (Professional Edition), and EE (Enterprise Edition) engines. They are the same options as *those in the **Saxon HE/PE/EE** preferences page* but they are configured as a specific set of transformation options

for each transformation scenario, while the values set in the preferences page apply as global options. The advanced options configured in a transformation scenario override the *global options* defined in the preferences page.

Saxon-HE/PE/EE Options

The advanced options for Saxon 9.7.0.15 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE) are as follows:

Mode ("-im")

A Saxon-specific option that sets the initial mode for the transformation.

Template ("-it")

A Saxon-specific option that sets the name of the initial XSLT template to be executed.

Use a configuration file ("-config")

Sets a Saxon 9.7.0.15 configuration file that is executed for XSLT transformation and validation processes.

Debugger trace into XPath expressions (applies to debugging sessions)

Instructs the *XSLT Debugger* to *step into* XPath expressions.

Version warnings ("-versmsg")

Warns you when the transformation is applied to an XSLT 1.0 stylesheet.

Line numbering ("-l")

Line numbers where errors occur are included in the output messages.

Expand attributes defaults ("-expand")

Specifies whether or not the attributes defined in the associated DTD or XML Schema are expanded in the output of the transformation you are executing.

DTD validation of the source ("-dtd")

Specifies whether or not the source document will be validated against their associated DTD. You can choose from the following:

- **On** - Requests DTD validation of the source file and of any files read using the `document()` function.
- **Off** - (default setting) Suppresses DTD validation.
- **Recover** - Performs DTD validation but treats the errors as non-fatal.

  Note: Any external DTD is likely to be read even if not used for validation, since DTDs can contain definitions of entities.

Recoverable errors ("-warnings")

Allows you to choose how dynamic errors are handled. The following options can be selected:

- **Recover silently ("silent")** - Continues processing without reporting the error.
- **Recover with warnings ("recover")** - Issues a warning but continues processing.
- **Signal the error and do not attempt recovery ("fatal")** - Issues an error and stops processing.

Strip whitespaces ("-strip")

Allows you to choose how the *strip whitespaces* operation is handled. You can choose one of the following values:

- **All ("all")** - Strips *all* whitespace text nodes from source documents before any further processing, regardless of any `xml:space` attributes in the source document.
- **Ignore ("ignorable")** - Strips all *ignorable* whitespace text nodes from source documents before any further processing, regardless of any `xml:space` attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.
- **None ("none")** - Strips *no* whitespace before further processing.

Optimization level ("-opt")

Allows you to set the optimization level. It is the value is an integer in the range of 0 (no optimization) to 10 (full optimization). This option allows optimization to be suppressed when reducing the compiling time

is important, optimization conflicts with debugging, or optimization causes extension functions with side-effects to behave unpredictably.

Saxon-PE/EE Options

The following advanced options are specific for Saxon 9.7.0.15 Professional Edition (PE) and Enterprise Edition (EE) only:

Register Saxon-CE extension functions and instructions

Registers the Saxon-CE extension functions and instructions when compiling a stylesheet using the Saxon 9.7.0.15 processors.

Note: Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Editor only for developing the Saxon-CE stylesheet, leaving the execution part to a web browser. See more details about *executing such a stylesheet on Saxonica's website*.

Allow calls on extension functions ("-ext")

If selected, the stylesheet is allowed to call external Java functions. This does not affect calls on integrated extension functions, including Saxon and EXSLT extension functions. This option is useful when loading an untrusted stylesheet (such as from a remote site using `http://[URL]`). It ensures that the stylesheet cannot call arbitrary Java methods and thus gain privileged access to resources on your machine.

Enable assertions ("-ea")

In XSLT 3.0, you can use the **xsl:assert** element to make assertions in the form of XPath expressions, causing a dynamic error if the assertion turns out to be false. If this option is selected, XSLT 3.0 `xsl:assert` instructions are enabled. If it is not selected (default), the assertions are ignored.

Saxon-EE Options

The advanced options that are specific for Saxon 9.7.0.15 Enterprise Edition (EE) are as follows:

XML Schema version

Use this option to change the default XML Schema version for this transformation. To change the default XML Schema version globally, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XML Parser** > **XML Schema** and use the ***Default XML Schema version** option*.

Validation of the source file ("-val")

Requests schema-based validation of the source file and of any files read using `document()` or similar functions. It can have the following values:

- **Schema validation ("strict")** - This mode requires an XML Schema and allows for parsing the source documents with strict schema-validation enabled.
- **Lax schema validation ("lax")** - If an XML Schema is provided, this mode allows for parsing the source documents with schema-validation enabled but the validation will not fail if, for example, element declarations are not found.
- **Disable schema validation** - This specifies that the source documents should be parsed with schema-validation disabled.

Validation errors in the result tree treated as warnings ("-outval")

Normally, if validation of result documents is requested, a validation error is fatal. Selecting this option causes such validation failures to be treated as warnings.

Write comments for non-fatal validation errors of the result document

The validation messages for non-fatal errors are written (wherever possible) as a comment in the result document itself.

Generate bytecode ("--generateByteCode:(on|off)")

If you select this option, Saxon-EE attempts to generate Java bytecode for evaluation of parts of a query or stylesheet that are amenable to such an action. For further details regarding this option, go to *http://www.saxonica.com/documentation9.5/index.html#!javadoc*.

Enable streaming mode

Selecting this option will allow an XSLT to run in streaming mode. It is not selected by default.

Other Options

Initializer class

Equivalent to the `-init` Saxon command-line argument. The value is the name of a user-supplied class that implements the `net.sf.saxon.lib.Initializer` interface. This initializer is called during the initialization process, and may be used to set any options required on the configuration programmatically. It is particularly useful for tasks such as registering extension functions, collations, or external object models, especially in Saxon-HE where the option cannot be set via a configuration file. Saxon only calls the initializer when running from the command line, but the same code may be invoked to perform initialization when running user application code.

FO Processor Tab (XSLT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **FO Processor** tab contains the following options:

Perform FO Processing

Specifies whether or not an FO processor is applied (either the built-in Apache FOP engine or an external engine defined in **Preferences**) during the transformation.

Input

Choose between the following options to specify which input file to use:

- **XSLT result as input** - The FO processor is applied to the result of the XSLT transformation that is defined in the **XSLT** tab.
- **XML URL as input** - The FO processor is applied to the input XML file.

Method

The output format of the FO processing. The available options depend on the selected processor type.

Processor

Specifies the FO processor to be used for the transformation. It can be the built-in Apache FOP processor or an *external processor*.

Output Tab (XSLT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Output** tab contains the following options:

Prompt for file

At the end of the transformation, a file browser dialog box is displayed for specifying the path and name of the file that stores the transformation result.

Save As

The path of the file where the result of the transformation is stored. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Open in Browser/System Application

If selected, Oxygen XML Editor automatically opens the result of the transformation in a system application associated with the file type of the result (for example, in Windows *PDF* files are often opened in *Acrobat Reader*).

Note: To set the web browser that is used for displaying HTML/XHTML pages, go to **Options** > **Preferences** > **Global**, and set it in the **Default Internet browser** field.

- **Output file** - When **Open in Browser/System Application** is selected, you can use this button to automatically open the default output file at the end of the transformation.
- **Other location** - When **Open in Browser/System Application** is selected, you can use this option to open the file specified in this field at the end of the transformation. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Open in editor

When this is option is selected, at the end of the transformation, the default output file is opened in a new editor panel with the appropriate built-in editor type (for example, if the result is an XML file it is opened in the built-in XML editor, or if it is an XSL-FO file it is opened with the built-in FO editor).

Show in results view as

You can choose to view the results in one of the following:

- **XML** - If this is selected, Oxygen XML Editor displays the transformation result in an XML viewer panel at the bottom of the application window with *syntax highlighting*.
- **SVG** - If this is selected, Oxygen XML Editor displays the transformation result in an *integrated SVG viewer in the **Results** panel* at the bottom of the application window and renders the result as an SVG image.
- **XHTML** - This option is only available if **Open in Browser/System Application** is not selected. If selected, Oxygen XML Editor displays the transformation result in a built-in XHTML browser panel at the bottom of the application window.

  Important: When transforming very large documents, you should be aware that selecting this option may result in very long processing times. This drawback is due to the built-in Java XHTML browser implementation. To avoid delays for large documents, if you want to see the XHTML result of the transformation, you should use an external browser by selecting the **Open in Browser/System Application** option instead.

  - **Image URLs are relative to** - If **Show in results view as XHTML** is selected, this option specifies the path used to resolve image paths contained in the transformation result. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

XProc Transformation

This type of transformation specifies the parameters and location of an XProc script.

A sequence of transformations described by an XProc script can be executed with an XProc transformation scenario. To create an **XProc transformation** scenario, use one of the following methods:

- Use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **XProc transformation**.
- Use the **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **XProc transformation**.

  Note: If a scenario is already associated with the edited document, selecting **Apply Transformation Scenario(s)** runs the associated scenario automatically. You can check to see if transformation scenarios are associated with the edited document by hovering your cursor over the **Apply Transformation Scenario** button.
- Go to **Window** > **Show View** and select **Transformation Scenarios** to display *this view*. Click the **New** button and select **XProc transformation**.

All three methods open the **New Scenario** dialog box.

The upper part of the dialog box allows you to specify the **Name** of the transformation scenario and the following **Storage** options:

- ***Project Options*** - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- ***Global Options*** - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.

XProc Tab

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **XProc** tab contains the following options:

XProc URL

Specify the source XSL file to be used by the transformation. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, the file is used directly from its remote location.

Processor

Allows you to select the XProc engine to be used for the transformation. You can select the built-in *Calabash* engine or a custom engine that is *configured in the **Preferences** dialog box*.

Inputs Tab (XProc Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Inputs** tab contains a list with the ports that the XProc script uses to read input data. Use the **Filter** text box to search for a specific term in the entire ports collection.

Each input port has an assigned name in the XProc script. The XProc engine reads data from the URL specified in the **URL** column.

The following actions are available for managing the input ports:

New

Opens an **Edit** dialog box that allows you to add a new port and its URL. The *built-in editor variables* and *custom editor variables* can be used to specify the URL.

Edit

Opens an **Edit** dialog box that allows you to modify the selected port and its URL. The *built-in editor variables* and *custom editor variables* can be used to specify the URL.

Delete

Removes the selected port from the list. It is available only for new ports that have been added to the list.

Parameters Tab (XProc Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Parameters** tab presents a list of ports and parameters collected from the XProc script. The tab is divided into three sections:

List of Ports

In this section, you can use the **New** and **Delete** buttons to add or remove ports.

List of Parameters

This section presents a list of parameters for each port and includes columns for the parameter name, namespace URI, and its value. Use the **Filter** text box to search for a specific term in the entire parameters collection. You can use the **New** and **Delete** buttons to add or remove parameters. You can edit the value of each cell in this table by double-clicking the cell. You can also sort the parameters by clicking the column headers.

Editor Variable Information

The *built-in editor variables* and *custom editor variables* can be used for specifying the URI. The message pane at the bottom of the dialog box provides more information about the editor variables that can be used.

Outputs Tab (XProc Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Outputs** tab displays a list of output ports (along with the URL) collected from the XProc script. Use the **Filter** text box to search for a specific term in the entire ports collection. You can also sort the columns by clicking the column headers.

The following actions are available for managing the output ports:

New

Opens an **Edit** dialog box that allows you to add a new output port and its URL. An *editor variable* can be inserted for the URL by using the **Insert Editor Variables** button. There is also a **Show in transformation results view** option that allows you to select whether or not the results will be displayed in the output ***Results view***.

Edit

Opens an **Edit** dialog box that allows you to edit an existing output port and its URL. An *editor variable* can be inserted for the URL by using the **Insert Editor Variables** button. There is also a **Show in transformation results view** option that allows you to select whether or not the results will be displayed in the output ***Results view***.

Delete

Removes the selected output port from the list. It is available only for new ports that have been added to the list.

Additional options that are available at the bottom of this tab include:

Open in Editor

If this option is selected, the XProc transformation result is automatically opened in an editor panel.

Open in Browser/System Application

If this option is selected, you can specify a file to be opened at the end of the XProc transformation in the browser or system application that is associated with the file type. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.

Results

The result of the XProc transformation can be displayed as a sequence in an output view with two sections:

- A list with the output ports on the left side.
- The content that correspond to the selected output port on the right side.

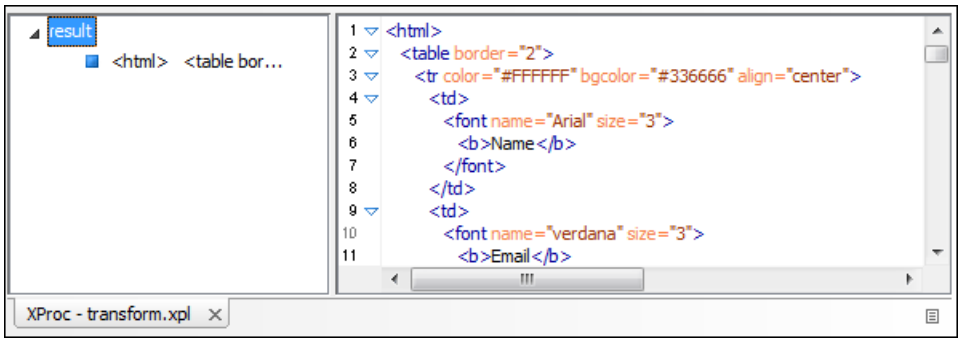


Figure 470: XProc Transformation Results View

Options Tab (XProc Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Options** tab displays a list of the options collected from the XProc script. The tab is divided into two sections:

List of Options

This section presents a list of options and includes columns for the option name, namespace URI, and its value. Use the **Filter** text box to search for a specific term in the entire options collection. You can use the **New** and **Delete** buttons to add or remove options. You can edit the value of each cell in this table by double-

clicking the cell. You can also sort the parameters by clicking the column headers. The names of edited options are displayed in bold.

Editor Variable Information

The *built-in editor variables* and *custom editor variables* can be used for specifying the URI. This section provides more information about the editor variables that can be used.

XQuery Transformation

This type of transformation specifies the parameters and location of an XML source that the edited XQuery file is applied on.

Note: When the XML source is a native XML database, the source field of the scenario is empty because the XML data is read with XQuery-specific functions, such as `document()`. When the XML source is a local XML file, the URL of the file is specified in the input field of the scenario.

To create an **XQuery transformation** scenario, use one of the following methods:

- Use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **XQuery transformation**.
- Use the **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **XQuery transformation**.

  Note: If a scenario is already associated with the edited document, selecting **Apply Transformation Scenario(s)** runs the associated scenario automatically. You can check to see if transformation scenarios are associated with the edited document by hovering your cursor over the **Apply Transformation Scenario** button.
- Go to **Window** > **Show View** and select **Transformation Scenarios** to display *this view*. Click the **New** button and select **XQuery transformation**.

All three methods open the **New Scenario** dialog box.

The upper part of the dialog box allows you to specify the **Name** of the transformation scenario and the following **Storage** options:

- ***Project Options*** - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- ***Global Options*** - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.

XQuery Tab

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **XQuery** tab contains the following options:

XML URL

Specifies the source XML file. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list. You can also use the **Open in editor** button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, then the file is used directly from its remote location.

Note: If the transformer engine is Saxon 9.x and a custom URI resolver is configured in the *advanced Saxon preferences page*, the XML input of the transformation is passed to that URI resolver.

XQuery URL

Specifies the source XQuery file to be used for the transformation. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list. You can also use the **Open in editor** button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, the file is used directly from its remote location.

Transformer

This drop-down menu presents all the transformation engines available to Oxygen XML Editor for performing a transformation. These include the built-in engines and *the external engines defined in the Custom Engines preferences page*. The engine you choose is used as the default transformation engine. Also, if an XSLT or XQuery document does not have an associated validation scenario, this transformation engine is used in the validation process (if it provides validation support).

Advanced options

Allows you to configure the *advanced options of the Saxon HE/PE/EE engine* for the current transformation scenario. To configure the same options globally, go to the *Saxon-HE/PE/EE preferences page*. For the current transformation scenario, these **Advanced options** override the options configured in that preferences page.

Parameters

Opens the ***Configure parameters*** *dialog box* for configuring the XQuery parameters. You can use the buttons in this dialog box to add, edit, or remove parameters. If the XQuery transformation engine is custom-defined, you can not use this dialog box to set parameters. Instead, you can copy all parameters from the dialog box using contextual menu actions and edit the custom XQuery engine to include the necessary parameters in the command line that starts the transformation process.

Extensions

Opens a *dialog box for configuring the XQuery extension JARS or classes* that define extension Java functions or extension XSLT elements used in the transformation.

XQuery Parameters

The global parameters of the XQuery file used in a transformation scenario can be configured by using the **Parameters** button in the **XQuery** tab.

The resulting dialog box includes a table that displays all the parameters of the current XQuery file, along with their descriptions and current values. You can also add, edit, and remove parameters, and use the **Filter** text box to search for a specific term in the entire parameters collection. Note that edited parameters are displayed with their name in bold.

If the **XPath** column is selected, the parameter value is evaluated as an XPath expression before starting the XQuery transformation.

Example:

For example, you can use expressions such as:

```
doc('test.xml')//entry
//person[@atr='val']
```

Note:

1. The **doc** function solves the argument relative to the XQuery file location. You can use full paths or *editor variables* (such as *${cfdu}* [current file directory]) to specify other locations: `doc('${cfdu}/test.xml')//*`
2. Only XPath functions are allowed.

Below the table, the following actions are available for managing the parameters:

New

Opens the **Add Parameter** dialog box that allows you to add a new parameter to the list. An *editor variable* can be inserted in the text box using the **Insert Editor Variables** button. If the **Evaluate as XPath** option is selected, the parameter will be evaluated as an XPath expression.

Edit

Opens the **Edit Parameter** dialog box that allows you to edit the selected parameter. An *editor variable* can be inserted in the text box using the **Insert Editor Variables** button. If the **Evaluate as XPath** option is selected, the parameter will be evaluated as an XPath expression.

Unset

Resets the selected parameter to its default value. Available only for edited parameters with set values.

Delete

Removes the selected parameter from the list. It is available only for new parameters that have been added to the list.

The bottom panel presents the following:

- The default value of the parameter selected in the table.
- A description of the parameter, if available.
- The system ID of the stylesheet that declares it.

Related Information:

Editor Variables on page 169

XQuery Extensions

The **Extensions** button is used to specify the *JAR* and classes that contain extension functions called from the XQuery file of the current transformation scenario. You can use the **Add**, **Edit**, and **Remove** buttons to configure the extensions.

An extension function called from the XQuery file of the current transformation scenario will be searched, in the specified extensions, in the order displayed in this dialog box. To change the order of the items, select the item to be moved and press the **Move up** or **Move down** buttons.

Advanced Saxon HE/PE/EE XQuery Transformation Options

The XQuery transformation scenario allows you to configure advanced options that are specific for the Saxon HE (Home Edition), PE (Professional Edition), and EE (Enterprise Edition) engines. They are the same options as *those in the **Saxon HE/PE/EE** preferences page* but they are configured as a specific set of transformation options for each transformation scenario, while the values set in the preferences page apply as global options. The advanced options configured in a transformation scenario override the *global options* defined in the preferences page.

The advanced options for Saxon 9.7.0.15 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE) are as follows:

Recoverable errors ("-warnings")

Allows you to choose how dynamic errors are handled. The following options can be selected:

- **Recover silently ("silent")** - Continues processing without reporting the error.
- **Recover with warnings ("recover")** - Issues a warning but continues processing.
- **Signal the error and do not attempt recovery ("fatal")** - Issues an error and stops processing.

Strip whitespaces ("-strip")

Allows you to choose how the *strip whitespaces* operation is handled. You can choose one of the following values:

- **All ("all")** - Strips *all* whitespace text nodes from source documents before any further processing, regardless of any `xml:space` attributes in the source document.
- **Ignore ("ignorable")** - Strips all *ignorable* whitespace text nodes from source documents before any further processing, regardless of any `xml:space` attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.
- **None ("none")** - Strips *no* whitespace before further processing.

Optimization level ("-opt")

Allows you to set the optimization level. It is the value is an integer in the range of 0 (no optimization) to 10 (full optimization). This option allows optimization to be suppressed when reducing the compiling time

is important, optimization conflicts with debugging, or optimization causes extension functions with side-effects to behave unpredictably.

Use linked tree model ("-tree:linked")

This option activates the linked tree model.

Enable XQuery 3.0 support ("-qversion:(1.0|3.0)")

If selected (default value), Saxon runs the XQuery transformation with the XQuery 3.0 support.

Initializer class

Equivalent to the `-init` Saxon command-line argument. The value is the name of a user-supplied class that implements the `net.sf.saxon.lib.Initializer` interface. This initializer is called during the initialization process, and may be used to set any options required on the configuration programmatically. It is particularly useful for tasks such as registering extension functions, collations, or external object models, especially in Saxon-HE where the option cannot be set via a configuration file. Saxon only calls the initializer when running from the command line, but the same code may be invoked to perform initialization when running user application code.

The following advanced options are specific for Saxon 9.7.0.15 Professional Edition (PE) and Enterprise Edition (EE) only:

Use a configuration file ("-config")

Sets a Saxon 9.7.0.15 configuration file that is used for XQuery transformation and validation scenarios.

Allow calls on extension functions ("-ext")

If selected, calls on external functions are allowed. Selecting this option is recommended in an environment where untrusted stylesheets may be executed. It also disables user-defined extension elements and the writing of multiple output files, both of which carry similar security risks.

The advanced options that are specific for Saxon 9.7.0.15 Enterprise Edition (EE) are as follows:

Validation of the source file ("-val")

Requests schema-based validation of the source file and of any files read using `document()` or similar functions. It can have the following values:

- **Schema validation ("strict")** - This mode requires an XML Schema and allows for parsing the source documents with strict schema-validation enabled.
- **Lax schema validation ("lax")** - If an XML Schema is provided, this mode allows for parsing the source documents with schema-validation enabled but the validation will not fail if, for example, element declarations are not found.
- **Disable schema validation** - This specifies that the source documents should be parsed with schema-validation disabled.

Validation errors in the result tree treated as warnings ("-outval")

Normally, if validation of result documents is requested, a validation error is fatal. Selecting this option causes such validation failures to be treated as warnings.

Write comments for non-fatal validation errors of the result document

The validation messages for non-fatal errors are written (wherever possible) as a comment in the result document itself.

Generate bytecode ("--generateByteCode:(on|off)")

If you select this option, Saxon-EE attempts to generate Java bytecode for evaluation of parts of a query or stylesheet that are amenable to such an action. For further details regarding this option, go to *http://www.saxonica.com/documentation9.5/index.html#!javadoc*.

Enable XQuery update ("-update:(on|off)")

This option controls whether or not XQuery update syntax is accepted. The default value is off.

Backup files updated by XQuery ("-backup:(on|off)")

If selected, backup versions for any XML files updated with an XQuery Update are generated. This option is available when the **Enable XQuery update** option is selected.

FO Processor Tab (XQuery Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **FO Processor** tab contains the following options:

Perform FO Processing

Specifies whether or not an FO processor is applied (either the built-in Apache FOP engine or an external engine defined in **Preferences**) during the transformation.

Input

Choose between the following options to specify which input file to use:

- **XQuery result as input** - The FO processor is applied to the result of the XQuery transformation that is defined in the **XQuery** tab.
- **XML URL as input** - The FO processor is applied to the input XML file.

Method

The output format of the FO processing. The available options depend on the selected processor type.

Processor

Specifies the FO processor to be used for the transformation. It can be the built-in Apache FOP processor or an *external processor*.

Output Tab (XQuery Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Output** tab contains the following options:

Present as a sequence

Selecting this option will reduce the time necessary to fetch the full results, as it will only fetch the first chunk of the results.

Prompt for file

At the end of the transformation, a file browser dialog box is displayed for specifying the path and name of the file that stores the transformation result.

Save As

The path of the file where the result of the transformation is stored. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Open in Browser/System Application

If selected, Oxygen XML Editor automatically opens the result of the transformation in a system application associated with the file type of the result (for example, in Windows *PDF* files are often opened in *Acrobat Reader*).

Note: To set the web browser that is used for displaying HTML/XHTML pages, go to **Options** > **Preferences** > **Global**, and set it in the **Default Internet browser** field.

- **Output file** - When **Open in Browser/System Application** is selected, you can use this button to automatically open the default output file at the end of the transformation.
- **Other location** - When **Open in Browser/System Application** is selected, you can use this option to open the file specified in this field at the end of the transformation. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Open in editor

When this is option is selected, at the end of the transformation, the default output file is opened in a new editor panel with the appropriate built-in editor type (for example, if the result is an XML file it is opened in the built-in XML editor, or if it is an XSL-FO file it is opened with the built-in FO editor).

Show in results view as

You can choose to view the results in one of the following:

- **XML** - If this is selected, Oxygen XML Editor displays the transformation result in an XML viewer panel at the bottom of the application window with *syntax highlighting*.
- **SVG** - If this is selected, Oxygen XML Editor displays the transformation result in an *integrated SVG viewer in the **Results** panel* at the bottom of the application window and renders the result as an SVG image.
- **XHTML** - This option is only available if **Open in Browser/System Application** is not selected. If selected, Oxygen XML Editor displays the transformation result in a built-in XHTML browser panel at the bottom of the application window.

  Important: When transforming very large documents, you should be aware that selecting this option may result in very long processing times. This drawback is due to the built-in Java XHTML browser implementation. To avoid delays for large documents, if you want to see the XHTML result of the transformation, you should use an external browser by selecting the **Open in Browser/System Application** option instead.

  - **Image URLs are relative to** - If **Show in results view as XHTML** is selected, this option specifies the path used to resolve image paths contained in the transformation result. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

SQL Transformation

This type of transformation specifies a database connection for the database server that runs the SQL file associated with the scenario. The data processed by the SQL script is located in the database.

To create an **SQL transformation** scenario, use one of the following methods:

- Use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **SQL transformation**.
- Use the **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **SQL transformation**.

  Note: If a scenario is already associated with the edited document, selecting **Apply Transformation Scenario(s)** runs the associated scenario automatically. You can check to see if transformation scenarios are associated with the edited document by hovering your cursor over the **Apply Transformation Scenario** button.
- Go to **Window** > **Show View** and select **Transformation Scenarios** to display *this view*. Click the **New** button and select **SQL transformation**.

All three methods open the **New Scenario** dialog box. This dialog box allows you to configure the following options that control the transformation:

Name

The unique name of the SQL transformation scenario.

Storage

Allows you to select one of the following storage options:

- ***Project Options*** - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- ***Global Options*** - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

SQL URL

Allows you to specify the URL of the SQL script. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list. You can also use the **Open in editor** button to open the specified file in the editor panel.

Connection

Allows you to select a connection from a drop-down list. To configure a connection, use the **Advanced options** button to open the ***Data Source** preferences page*.

Parameters

Allows you to add or configure parameters for the transformation.

Editing a Transformation Scenario

Editing a transformation scenario is useful if you need to configure some of its parameters.

Note: Since transformation scenarios that are associated with predefined *frameworks* are read-only, to edit one of these scenarios you will need to *duplicate it and edit the duplicated scenario*.

To configure an existing transformation scenario, follow these steps:

1. Select the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu.

   Step Result: The ***Configure Transformation Scenario(s)*** *dialog box* is opened.
2. Select the particular transformation scenario and click the **Edit** button at the bottom of the dialog box or from the contextual menu.

   Tip: You could also select the scenario and the **Edit** button in the ***Transformation Scenarios*** *view* to achieve the same result.

Result: This will open an **Edit scenario** configuration dialog box that allows you to configure various options in several tabs, depending on the type of transformation scenario that was selected.

Transformation Types

The ***Configure Transformation Scenario(s)*** *dialog box* contains a **Type** column that shows you the transformation type for each of the listed scenarios. Each type of transformation contains includes some tabs with various configuration options.

The following is a list of the transformation types and their particular tabs (click the name of each tab below to see details about all the options that are available):

- **DITA OT** - This type of transformation includes configurable options in the following tabs:
  - *Skins Tab* (Available for **WebHelp Classic** and **WebHelp Classic with Feedback**)
  - *Templates Tab* (Available for **WebHelp Responsive** and **WebHelp Responsive with Feedback**)
  - *FO Processor Tab* (Available for PDF output)
  - *Parameters Tab*
  - *Filters Tab*
  - *Advanced Tab*
  - *Output Tab*
- **ANT** - This type of transformation includes configurable options in the following tabs:
  - *Options Tab*
  - *Parameters Tab*
  - *Output Tab*
- **XSLT** - This type of transformation includes configurable options in the following tabs:
  - *XSLT Tab*
  - *FO Processor Tab*
  - *Output Tab*
- **XProc** - This type of transformation includes configurable options in the following tabs:
  - *XProc Tab*
  - *Inputs Tab*
  - *Parameters Tab*
  - *Outputs Tab*
  - *Options Tab*
- **XQuery** - This type of transformation includes configurable options in the following tabs:

- *XQuery Tab*
- *FO Processor Tab*
- *Output Tab*

Related Information:

Creating New Transformation Scenarios on page 886
Duplicating a Transformation Scenario on page 926
Configure Transformation Scenario(s) Dialog Box on page 926

Duplicating a Transformation Scenario

Duplicating a transformation scenario is useful for creating a scenario that is similar to an existing one or to edit a predefined transformation scenario.

To configure an existing transformation scenario, follow these steps:

1. Select the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu.

   Step Result: The ***Configure Transformation Scenario(s)*** *dialog box* is opened.
2. Select the particular transformation scenario and click the **Duplicate** button at the bottom of the dialog box or from the contextual menu.

   Tip: You could also select the scenario and the **Duplicate** button in the ***Transformation Scenarios*** *view* to achieve the same result.

Result: This will open an **Edit scenario** configuration dialog box that allows you to configure various options in several tabs, depending on the type of transformation scenario that was selected. For information about all the specific options in the various tabs, see the *Transformation Types section*.

Related Information:

Creating New Transformation Scenarios on page 886
Editing a Transformation Scenario on page 925

Configure Transformation Scenario(s) Dialog Box

You can use the **Configure Transformation Scenarios(s)** dialog box for *editing exiting transformation scenarios* or *creating new ones*.

To open this dialog box, use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu.

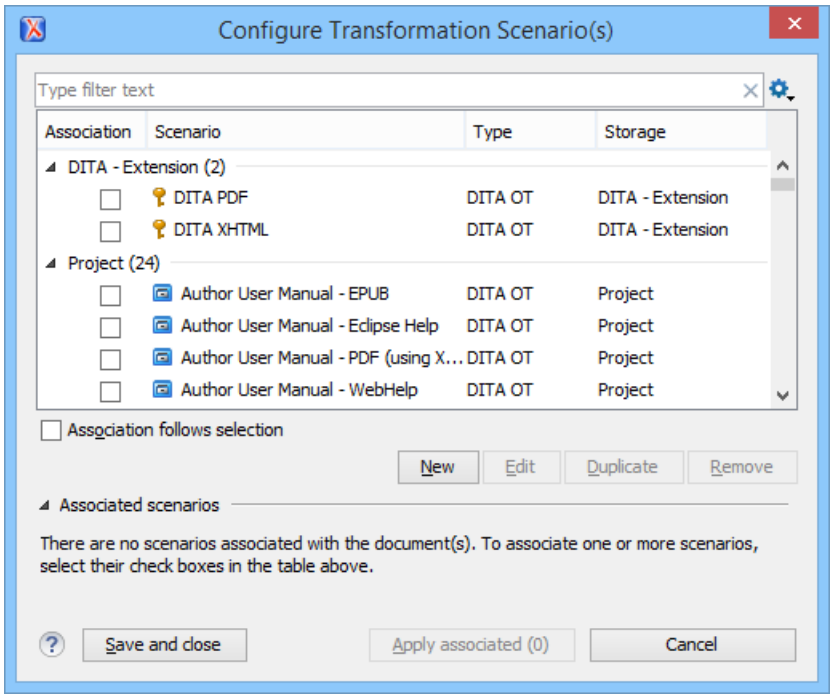


Figure 471: Configure Transformation Scenario(s) Dialog Box

The top section of the dialog box contains a filter that allows you to search through the scenarios list and the **Settings** button allows you to configure the following options:

Show all scenarios

Select this option to display all the available scenarios, regardless of the document they are associated with.

Show only the scenarios available for the editor

Select this option to only display the scenarios that Oxygen XML Editor can apply for the current document type.

Show associated scenarios

Select this option to only display the scenarios associated with the document you are editing.

Import scenarios

This option opens the **Import scenarios** dialog box that allows you to select the `scenarios` file that contains the scenarios you want to import. If one of the scenarios you import is identical to an existing scenario, Oxygen XML Editor ignores it. If a conflict appears (an imported scenario has the same name as an existing one), you can choose between two options:

- Keep or replace the existing scenario.
- Keep both scenarios.

  Note: When you keep both scenarios, Oxygen XML Editor adds `imported` to the name of the imported scenario.

Export selected scenarios

Use this option to export selected scenarios individually. Oxygen XML Editor creates a `scenarios` file that contains the scenarios that you export. This is useful if you want to share scenarios with others or export them to another computer.

The middle section of the dialog box displays the scenarios that you can apply to the current document. You can view both the scenarios associated with the current document type and the scenarios defined at *project level*. The following columns are used to display the transformation scenarios:

- **Association** - The checkboxes in this column mark whether or not a transformation scenario is associated with the current document.
- **Scenario** - This column presents the names of the transformation scenarios.

- **Type** - If the **Show Type** contextual menu option is selected, this column displays the type of the transformation scenario. For further details about the types of transformation scenarios that are available in Oxygen XML Editor, see the *Transformation Types section*.
- **Storage** - If the **Show Storage** contextual menu option is selected, this column displays where a transformation scenario is stored.

To sort each column you can left-click its header. The contextual menu of each header also includes the following actions:

Show Type

Use this option to display the transformation type of each scenario.

Show Storage

Use this option to display the storage location of the scenarios.

Group by Association

Select this option to group the scenarios depending on whether or not they are associated with the current document.

Group by Type

Select this option to group the scenarios by their type.

Group by Storage

Select this option to group the scenarios by their storage location.

Ungroup all

Select this option to ungroup all the scenarios.

Reset Layout

Select this option to restore the default settings of the layout.

The bottom section of the dialog box contains the following actions:

Association follows selection

Select this checkbox to automatically associate selected transformation scenarios with the current document. This option can also be used for multiple selections.

Note: When this option is selected, the **Association** column is hidden.

New

This button allows you to *create a new transformation scenario*.

Edit

This button opens the **Edit Scenario** dialog box that allows you to configure the options of the transformations scenario. For information about all the specific options in the various tabs, see the *Transformation Types section*.

Note: If you try to edit a transformation scenario associated with a defined document type, Oxygen XML Editor displays a warning message to inform you that this is not possible and gives you the option to create a *duplicate transformation scenario* to edit instead.

Duplicate

Use this button to create a *duplicate transformation scenario*.

Remove

Use this button to remove transformation scenarios.

Note: Removing scenarios associated with a defined document type is not allowed.

The **Edit**, **Duplicate**, and **Remove** actions are also available in the contextual menu of the transformation scenarios listed in the middle section of the dialog box (along with **Import scenarios** and **Export selected scenarios**).

This contextual menu also contains a **Change storage** action that allows you to change the storage location of a transformation scenario to ***Project Options*** or ***Global Options***. You are also able to keep the original storage location and make a copy of the selected scenario in the new storage location.

Related Information:

Editing a Transformation Scenario on page 925
Duplicating a Transformation Scenario on page 926

Apply Batch Transformations

A transformation action can be applied on a batch of selected files *from the contextual menu of the **Project** view* without having to open the files involved in the transformation. You can apply the same scenario to a batch of files or multiple scenarios to a single file or batch of files.

1. (Optional, but recommended) Organize the files you want to transform in logical folders.
   a) Create a logical folder in the ***Project** view* by using the **New** > **Logical Folder** action from the contextual menu of the root file.
   b) Add files you want to transform to the logical folder by using the **Add Files** or **Add Edited File** actions from the contextual menu of the logical folder.

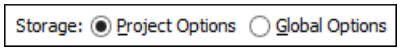
      Note: You can skip this step if the files are already in a dedicated folder that does not include any additional files or folders. You can also manually select the individual files in the ***Project** view* each time you want to transform them, but this can be tedious.
2. Select the files you want to transform (or the newly created logical folder) and from the contextual menu, select **Transform** > **Configure Transformation Scenario(s)** to choose one or more transformation scenarios to be applied on all the files in the logical folder.
3. Use Oxygen XML Editor *editor variables* to specify the input and output files. This ensures that each file from the selected set of resources is processed and that the output is not overwritten by the subsequent processing.
   a) Edit the transformation scenario to make sure the appropriate *editor variable* is assigned for the input file. For example, for a *DocBook PDF transformation* make sure the **XML URL** input box is set to the *`${currentFileURL}` editor variable*. For a DITA PDF transformation make sure the `args.input` parameter is set to the *`${cf}` editor variable*.
   b) Edit the transformation scenario to make sure the appropriate editor variable is assigned for the output file. For example, for an **XML transformation with XSLT**, switch to the **Output** tab and set the path of the output file using a construct of *editor variables*, such as `${cfd}/${cfn}.html`.
4. Now that logical folder has been associated with one or more transformation scenarios, whenever you want to apply the same batch transformation you can select **Transform** > **Transform with** from the contextual menu and the same previously associated scenario(s) will be applied.
5. If you want a different type of transformation to be applied to each file inside the logical folder, associate individual scenarios for each file and select **Transform** > **Apply Transformation Scenario(s)** from the contextual menu of the logical folder.

Related Information:

Editor Variables on page 169

Sharing Transformation Scenarios

The transformation scenarios and their settings can be shared with other users by saving them at *project level* or by *exporting them to a specialized scenarios file* that can then be imported. When you create a new transformation scenario or edit an existing one, there is a **Storage** option to control whether the scenarios are stored in ***Project Options*** or ***Global Options***.



Selecting ***Project Options*** stores the scenario in the project file and can be shared with other users that have access to the project. If your project is saved on a source versioning system (CVS, SVN, Source Safe, etc.) then your team will have access to the scenarios that you define. When you create a scenario at the project level, the URLs from the scenario become relative to the project URL.

Selecting ***Global Options*** stores the scenario in the global options that are stored in the user home directory.

You can also change the storage options on existing transformation scenarios by using the ***Change storage** action* from the contextual menu of the list of scenarios.

Related Information:

Sharing Application Settings on page 163

Transformation Scenarios View

You can manage the transformation scenarios by using the **Transformation Scenarios** view. To open this view, select **Window** > **Show View** > **Transformation Scenarios**.

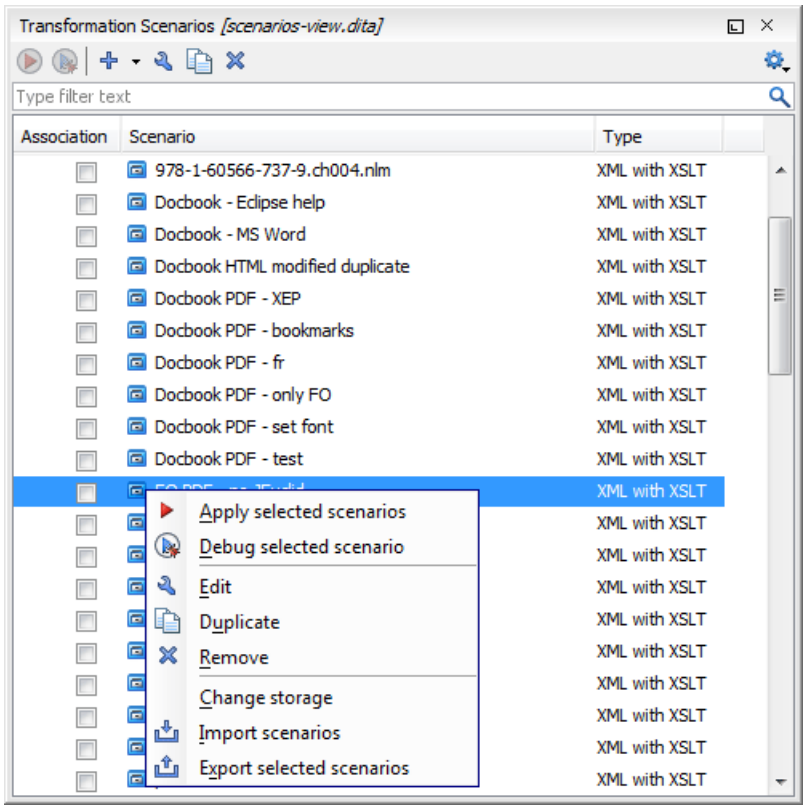


Figure 472: Transformation Scenarios view

Oxygen XML Editor supports multiple scenarios association. To associate multiple scenarios with a document, select the checkboxes in front of each scenario. You can also associate multiple scenarios with a document from the ***Configure Transformation Scenario(s)** dialog box*.

The **Transformation Scenarios** view presents both global and *project-level* scenarios. By default, Oxygen XML Editor presents the items in the following order:

1. Scenarios that match the current *framework*.
2. Scenarios that match the current project.
3. Scenarios that match other *frameworks*.

Toolbar/Contextual Menu Actions and Options

The following actions and options are available on the toolbar or in the contextual menu:

Apply selected scenarios

Select this option to run the current transformation scenario.

Debug selected scenario

Select this option to switch to the **Debugger** *perspective* and initialize it with the parameters from the scenario (the XML, XSLT, or XQuery input, the transformation engine, the XSLT parameters).

New

This drop-down menu contains a list of the *scenarios that you can create*. Oxygen XML Editor determines the most appropriate scenarios for the current type of file and displays them at the beginning of the list, followed by the rest of the scenarios.

Duplicate

Adds a new scenario to the list that is a duplicate of the current scenario. It is useful for creating a scenario that is similar to an existing one.

Edit

Opens the dialog box that allows you to configure various options in several tabs, depending on the type of transformation scenario that was selected. For information about all the specific options in the various tabs, see the *Transformation Types section*.

Remove

Removes the current scenario from the list. This action is also available by using the **Delete** key.

Change storage

Use this option to change the storage location of the selected scenario. You are also able to keep the original storage location and make a copy of the selected scenario in the target storage location.

Import scenarios

This option opens the **Import scenarios** dialog box that allows you to select the `scenarios` file that contains the scenarios you want to import. If one of the scenarios you import is identical to an existing scenario, Oxygen XML Editor ignores it. If a conflict appears (an imported scenario has the same name as an existing one), you can choose between two options:

- Keep or replace the existing scenario.
- Keep both scenarios.

  Note: When you keep both scenarios, Oxygen XML Editor adds `imported` to the name of the imported scenario.

Export selected scenarios

Use this option to export transformation and validation scenarios individually. Oxygen XML Editor creates a `scenarios` file that contains the scenarios that you export.

Settings

This drop-down menu allows you to configure the following options (many of these options are also available if you right-click the name of a column):

Show all scenarios

Select this option to display all the available scenarios, regardless of the document they are associated with.

Show only the scenarios available for the editor

Select this option to only display the scenarios that Oxygen XML Editor can apply for the current document type.

Show associated scenarios

Select this option to only display the scenarios associated with the document you are editing.

Change storage

Use this option to change the storage location of the selected scenario to ***Project Options*** or ***Global Options***. You are also able to keep the original storage location and make a copy of the selected scenario in the new storage location.

Import scenarios

This option opens the **Import scenarios** dialog box that allows you to select the `scenarios` file that contains the scenarios you want to import. If one of the scenarios you import is identical to an existing scenario, Oxygen XML Editor ignores it. If a conflict appears (an imported scenario has the same name as an existing one), you can choose between two options:

- Keep or replace the existing scenario.
- Keep both scenarios.

  Note: When you keep both scenarios, Oxygen XML Editor adds `imported` to the name of the imported scenario.

Export selected scenarios

Use this option to export selected scenarios individually. Oxygen XML Editor creates a `scenarios` file that contains the scenarios that you export. This is useful if you want to share scenarios with others or export them to another computer.

Show Type

Use this option to display the transformation type of each scenario.

Show Storage

Use this option to display the storage location of the scenarios.

Group by Association

Select this option to group the scenarios depending on whether or not they are associated with the current document.

Group by Type

Select this option to group the scenarios by their type.

Group by Storage

Select this option to group the scenarios by their storage location.

Ungroup all

Select this option to ungroup all the scenarios.

Reset Layout

Select this option to restore the default settings of the layout.

Related Information:

Editing a Transformation Scenario on page 925
Creating New Transformation Scenarios on page 886

Debugging PDF Transformations

To debug a DITA PDF transformation scenario using the XSLT Debugger follow these steps:

1. *Open the **Preferences** dialog box **(Options > Preferences)***, go to **XML** > **XML Catalog**, click **Add**, and select the file located at *`DITA-OT-DIR`*`\plugins\org.dita.pdf2\cfg\catalog.xml`.
2. Open the map in the ***DITA Maps Manager*** and create a **DITA Map PDF** transformation scenario.
3. Edit the scenario, go to the **Parameters** tab and change the value of the **clean.temp** parameter to **no**.
4. Run the transformation scenario.
5. Open the **stage1.xml** file located in the temporary directory and *format and indent* it.
6. Create a transformation scenario for this XML file by associating the `topic2fo_shell_fop.xsl` stylesheet located at *`DITA-OT-DIR`*`\plugins\org.dita.pdf2\xsl\fo\topic2fo_shell_fop.xsl`. If you are specifically using the RenderX XEP or Antenna House FO processors to build the PDF output, you should use the XSL stylesheets `topic2fo_shell_xep.xsl` or `topic2fo_shell_axf.xsl` located in the same folder.
7. In the transformation scenario edit the XSLT Processor combo box choose the Saxon EE XSLT processor (the same processor used when the DITA OT transformation is executed).
8. In the transformation scenario, edit the **Parameters** list and set the parameter *locale* with the value *en_GB* and the parameter *customizationDir.url* to point either to your customization directory or to the default DITA OT customization directory. Its value should have a URL syntax like this: `file://c:/path/to/`*`DITA-OT-DIR`*`/plugins/org.dita.pdf2/cfg`.
9. Debug the transformation scenario.

Related Information:

Working with the XSLT / XQuery Debugger on page 1155

Configuring Calabash with XEP

To generate PDF output from your XProc pipeline (when using the Calabash XProc processor), follow these steps:

1. Open the *[OXYGEN_INSTALL_DIR]*/lib/xproc/calabash/engine.xml file.
2. Uncomment the <system-property name="com.xmlcalabash.fo-processor" value="com.xmlcalabash.util.FoXEP"/> system property.
3. Uncomment the <system-property name="com.renderx.xep.CONFIG" file="../../../tools/xep/xep.xml"/> system property. Edit the file attribute to point to the configuration file that is usually located in the XEP installation folder.
4. Uncomment the references to the XEP libraries. Edit them to point to the matching library names from the XEP installation directory.
5. Restart Oxygen XML Editor.

Integration of an External XProc Engine

The Javadoc documentation of the XProc API is available for download from the application website as a zip file: *xprocAPI.zip*.

To create an XProc integration project, follow these steps:

1. Move the oxygen.jar file from *[OXYGEN_INSTALL_DIR]*/lib to the lib folder of your project.
2. Implement the ro.sync.xml.transformer.xproc.api.XProcTransformerInterface interface.
3. Create a *Java archive (JAR)* from the classes you created.
4. Create a engine.xml file according with the engine.dtd file. The attributes of the engine element are as follows:
   1. name - The name of the XProc engine.
   2. description - A short description of the XProc engine.
   3. class - The complete name of the class that implements ro.sync.xml.transformer.xproc.api.XProcTransformerInterface.
   4. version - The version of the integration.
   5. engineVersion - The version of the integrated engine.
   6. vendor - The name of the vendor / implementer.
   7. supportsValidation - true if the engine supports validation (otherwise, false).

   The engine element has only one child, runtime. The runtime element contains several library elements with the name attribute containing the relative or absolute location of the libraries necessary to run this integration.
5. Create a folder with the name of the integration in the *[OXYGEN_INSTALL_DIR]*/lib/xproc.
6. Place the engine.xml and all the libraries necessary to run the new integration in that folder.

XSLT Processors

This section explains how to configure an XSLT processor and extensions for such a processor in Oxygen XML Editor.

Supported XSLT Processors

Oxygen XML Editor includes the following XSLT processors:

- **Xalan 2.7.1** - *Xalan-Java* is an XSLT processor for transforming XML documents into HTML, text, or other XML document types. It implements XSL Transformations (XSLT) Version 1.0 and XML Path Language (XPath) Version 1.0.
- **Saxon 6.5.5** - *Saxon 6.5.5* is an XSLT processor that implements the Version 1.0 XSLT and XPath with a number of powerful extensions. This version of Saxon also includes many of the new features that were first defined in the XSLT 1.1 working draft, but for conformance and portability reasons these are not available if the stylesheet header specifies version="1.0".
- **Saxon 9.7.0.15 Home Edition (HE), Professional Edition (PE)** - *Saxon-HE/PE* implements the basic conformance level for XSLT 2.0 / 3.0 and XQuery 1.0. The term *basic XSLT 2.0 / 3.0 processor* is defined in

the draft XSLT 2.0 / 3.0 specifications. It is a conformance level that requires support for all features of the language other than those that involve schema processing. The HE product remains open source, but removes some of the more advanced features that are present in Saxon-PE.

- **Saxon 9.7.0.15 Enterprise Edition (EE)** - *Saxon EE* is the schema-aware edition of Saxon and it is one of the built-in processors included in Oxygen XML Editor. Saxon EE includes an XML Schema processor, and schema-aware XSLT, XQuery, and XPath processors.

  The validation in schema aware transformations is done according to the W3C XML Schema 1.0 or 1.1. This can be *configured in Preferences*.

  Note: Oxygen XML Editor implements a Saxon *framework* that allows you to create Saxon configuration files. Two templates are available: **Saxon collection catalog** and **Saxon configuration**. Both of these templates support content completion, element annotation, and attribute annotation.

  Note: Saxon can use the *ICU-J localization library* (`saxon9-icu.jar`) to add support for sorting and date/number formatting in a wide variety of languages. This library is not included in the Oxygen XML Editor installation kit. However, Saxon will use the default collation and localization support available in the currently used JRE. To enable this capability, follow these steps:

  1. Download Saxon 9.7.0.15 Professional Edition (PE) or Enterprise Edition (EE) from *http://www.saxonica.com*.
  2. Unpack the downloaded archive.
  3. Create a new XSLT transformation scenario (or edit an existing one). In the **XSLT** tab, click the **Extensions** button to open the list of additional libraries used by the transformation process.
  4. Click **Add** and browse to the folder where you unpacked the downloaded archive and choose the `saxon9-icu.jar` file.

  Note that the `saxon9-icu.jar` should NOT be added to the application library folder because it will conflict with another version of the ICU-J library that comes bundled with Oxygen XML Editor.

- **Saxon-CE (Client Edition)** is Saxonica's implementation of XSLT 2.0 for use on web browsers. Oxygen XML Editor provides support for editing stylesheets that contain Saxon-CE extension functions and instructions. This support improves the validation, content completion, and syntax highlighting.

  Note: Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Editor only for developing the Saxon-CE stylesheet, leaving the execution part to a web browser. See more details about *executing such a stylesheet on Saxonica's website*.

  Note: A specific template, named **Saxon-CE stylesheet**, is available in the ***New** document wizard*.

- **Xsltproc (libxslt)** - *Libxslt* is the XSLT C library developed for the Gnome project. `Libxslt` is based on *libxml2*, the XML C library developed for the Gnome project. It also implements most of the EXSLT set of processor-portable extensions, functions, and some of Saxon's evaluate and expression extensions. The *libxml2* version included in Oxygen XML Editor is 2.7.6 and the `Libxslt` version is 1.1.26.

  Oxygen XML Editor uses `Libxslt` through its command line tool (`Xsltproc`). The XSLT processor is included in the distribution kit of the stand-alone version for Windows and Mac OS X. Since there are differences between various Linux distributions, on Linux you must install `Libxslt` on your machine as a separate application and set the PATH variable to contain the `Xsltproc` executable.

  Note: The Xsltproc processor can be configured from the ***XSLTPROC** options page*.

  CAUTION: There is a known problem where file paths that contain spaces are not handled correctly in the LIBXML processor. For example, the built-in *XML Catalog* files of the predefined document types (DocBook, TEI, DITA, etc.) are not handled properly by LIBXML if Oxygen XML Editor is installed in the default location on Windows (C:\Program Files). This is because the built-in *XML catalog* files are stored in the `[OXYGEN_INSTALL_DIR]/frameworks` subdirectory of the installation directory, and in this case it contains a space character.

- **MSXML 4.0** - *MSXML 4.0* is available only on Windows platforms. It can be used for *transformation* and *validation of XSLT stylesheets*.

  Oxygen XML Editor uses the Microsoft XML parser through its command line tool *`msxsl.exe`*.

  Since `msxsl.exe` is only a wrapper, Microsoft Core XML Services (MSXML) must be installed on the computer. Otherwise, you will get a corresponding warning. You can get the latest Microsoft XML parser from *Microsoft web-site*.

- **MSXML .NET** - *MSXML .NET* is available only on Windows platforms. It can be used for *transformation* and *validation of XSLT stylesheets*.

  Oxygen XML Editor performs XSLT transformations and validations using the .NET *Framework* XSLT implementation (`System.Xml.Xsl.XslTransform` class) through the **nxslt** command line utility. The **nxslt** version included in Oxygen XML Editor is 1.6.

  You should have the .NET *Framework* version 1.0 already installed on your system. Otherwise, you will get the following warning: `MSXML.NET requires .NET Framework version 1.0 to be installed. Exit code: 128`.

  You can get the .NET *Framework* version 1.0 from the *Microsoft website*.
- **.NET 1.0** - A transformer based on the `System.Xml` 1.0 library available in the .NET 1.0 and .NET 1.1 *frameworks* from Microsoft (*http://msdn.microsoft.com/xml/*). It is available only on Windows.

  You should have the .NET *Framework* version 1.0 or 1.1 already installed on your system. Otherwise, you will get the following warning: `MSXML.NET requires .NET Framework version 1.0 to be installed. Exit code: 128`.

  You can get the .NET *Framework* version 1.0 from the *Microsoft website*.
- **.NET 2.0** - A transformer based on the `System.Xml` 2.0 library available in the .NET 2.0 *Framework* from *Microsoft*. It is available only on Windows.

  You should have the .NET *Framework* version 2.0 already installed on your system. Otherwise, you will get the following warning: `MSXML.NET requires .NET Framework version 2.0 to be installed. Exit code: 128`.

  You can get the .NET *Framework* version 2.0 from the *Microsoft website*.

For information about configuring the XSLT preferences, see the *XSLT options* section.

Configuring Custom XSLT Processors

Oxygen XML Editor allows you to configure custom processors to be used for running XSLT and XQuery transformations.

To add a new custom processor, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **XML** > **XSLT-FO-XQuery** > **Custom Engines**.
2. Click the **New** button at the bottom of the dialog box.
3. Configure the *parameters for the custom engine*.
4. Click **OK**.

Note: You can not use these custom engines in *the Debugger perspective*.

The output messages of a custom processor are displayed in an output view at the bottom of the application window. If an output message follows *the format of an Oxygen XML Editor linked message*, clicking it highlights the location of the message in an editor panel containing the file referenced in the message.

Related Information:

Custom Engines Preferences on page 133

Configuring the XSLT Processor Extensions Paths

The Xalan and Saxon processors support the use of extension elements and extension functions. Unlike a literal result element, which the stylesheet simply transfers to the result tree, an extension element performs an action. The extension is usually used because the XSLT stylesheet fails in providing adequate functions for accomplishing a more complex task.

The DocBook extensions for Xalan and Saxon are included in the `[OXYGEN_INSTALL_DIR]\frameworks\docbook\xsl\extensions` folder.

For more information about how to use extensions, see the following links:

- **Xalan** - *http://xml.apache.org/xalan-j/extensions.html*
- **Saxon 6.5.5** - *http://saxon.sourceforge.net/saxon6.5.5/extensions.html*
- **Saxon 9.7.0.15** - *http://www.saxonica.com/documentation9.5/index.html#!extensibility*

To set an XSLT processor extension (a directory or a `jar` file), use *the **Extensions** button* in the **Edit scenario** dialog box.

Note: The old way of setting an extension (using the parameter -Dcom.oxygenxml.additional.classpath) was deprecated, and instead you should use the extension mechanism of the XSLT transformation scenario.

XSL-FO Processors

This section explains how to apply XSL-FO processors when transforming XML documents to various output formats in Oxygen XML Editor.

Built-in XSL-FO Processor

The Oxygen XML Editor installation package is distributed with the *Apache FOP* that is a Formatting Objects processor for rendering your XML documents to PDF. *FOP* is a print and output independent formatter driven by XSL Formatting Objects. *FOP* is implemented as a Java application that reads a formatting object tree and renders the resulting pages to a specified output.

Other FO processors can be configured in *the **Preferences** dialog box*.

Add a Font to the Built-in FO Processor - Simple Version

If the font that must be set to Apache FOP is one of the fonts that are installed in the operating system you should follow the next steps for creating and setting a FOP configuration file that looks for the font that it needs in the system fonts. It is a simplified version of *the procedure for setting a custom font in Apache FOP*.

1. Register the font in FOP configuration. (This is not necessary for DITA PDF transformations, skip to the next step)
   a) Create a FOP configuration file that specifies that FOP should look for fonts in the installed fonts of the operating system.

```
<fop version="1.0">
  <renderers>
    <renderer mime="application/pdf">
      <fonts>
        <auto-detect/>
      </fonts>
    </renderer>
  </renderers>
</fop>
```

   b) *Open the **Preferences** dialog box **(Options** > **Preferences)***, go to **XML** > **XSLT/FO/XQuery** > **FO Processors**, and enter the path of the FOP configuration file in the **Configuration file** text field.
2. Set the font on the document content.

   This is done usually with XSLT stylesheet parameters and depends on the document type processed by the stylesheet.

   - For DocBook documents you can start with the predefined scenario called **DocBook PDF**, *edit the XSLT parameters* and set the font name (in our example the font family name is **Arial Unicode MS**) to the parameters body.font.family and title.font.family.
   - For TEI documents you can start with the predefined scenario called **TEI PDF**, *edit the XSLT parameters* and set the font name (in our example **Arial Unicode MS**) to the parameters bodyFont and sansFont.
   - For DITA transformations to PDF using DITA-OT you should modify the following two files:
     - *`DITA-OT-DIR`*`/plugins/org.dita.pdf2/cfg/fo/font-mappings.xml` - The `font-face` element included in each element `physical-font` having the `attribute char-set="default"` must contain the name of the font (**Arial Unicode MS** in our example)
     - *`DITA-OT-DIR`*`/plugins/org.dita.pdf2/fop/conf/fop.xconf` - An element `auto-detect` must be inserted in the element `fonts`, which is inside the element `renderer` that has the attribute `mime="application/pdf"`:

```
<renderer mime="application/pdf">
  . . .
   <fonts>
       <auto-detect/>
   </fonts>
  . . .
</renderer>
```

Add a Font to the Built-in FO Processor

If an XML document is transformed to PDF using the built-in Apache FOP processor but it contains some Unicode characters that cannot be rendered by the default PDF fonts, then a special font that is capable to render these characters must be configured and embedded in the PDF result.

Important: If this special font is installed in the operating system, there is a simple way of telling FOP to look for it. See *the simplified procedure for adding a font to FOP*.

1. Locate the font.

   First, find out the name of a font that has the glyphs for the special characters you used. One font that covers most characters, including Japanese, Cyrillic, and Greek, is Arial Unicode MS.

   On Windows the fonts are located into the `C:\Windows\Fonts` directory. On Mac, they are placed in `/Library/Fonts`. To install a new font on your system, is enough to copy it in the `Fonts` directory.

2. Generate a font metrics file from the font file.
   a) Open a terminal.
   b) Change the working directory to the Oxygen XML Editor install directory.
   c) Create the following script file in the Oxygen XML Editor installation directory.

      For OS X and Linux create a file `ttfConvert.sh`:

      ```
      #!/bin/sh

      export LIB=lib
      export CP=$LIB/fop.jar
      export CP=$CP:$LIB/avalon-framework-4.2.0.jar
      export CP=$CP:$LIB/xercesImpl.jar
      export CP=$CP:$LIB/commons-logging-1.1.3.jar
      export CP=$CP:$LIB/commons-io-1.3.1.jar
      export CP=$CP:$LIB/xmlgraphics-commons-1.5.jar
      export CP=$CP:$LIB/xml-apis.jar
      export CMD="java -cp $CP org.apache.fop.fonts.apps.TTFReader"
      export FONT_DIR='.'

      $CMD $FONT_DIR/Arialuni.ttf Arialuni.xml
      ```

      For Windows create a file `ttfConvert.bat`:

      ```
      @echo off
      set LIB=lib
      set CP=%LIB%\fop.jar
      set CP=%CP%;%LIB%\avalon-framework-4.2.0.jar
      set CP=%CP%;%LIB%\xercesImpl.jar
      set CP=%CP%;%LIB%\commons-logging-1.1.3.jar
      set CP=%CP%;%LIB%\commons-io-1.3.1.jar
      set CP=%CP%;%LIB%\xmlgraphics-commons-1.5.jar
      set CP=%CP%;%LIB%\xml-apis.jar
      set CMD=java -cp "%CP%" org.apache.fop.fonts.apps.TTFReader
      set FONT_DIR=C:\Windows\Fonts
      %CMD% %FONT_DIR%\Arialuni.ttf Arialuni.xml
      ```

      The paths specified in the file are relative to the Oxygen XML Editor installation directory. If you decide to create it in other directory, change the file paths accordingly.

      The *FONT_DIR* can be something different on your system. Check that it points to the correct font directory. If the Java executable is not in the *PATH*, specify the full path of the executable.

      If the font has bold and italic variants, convert them too by adding two more lines to the script file:

      - for OS X and Linux:

        ```
        $CMD $FONT_DIR/Arialuni-Bold.ttf Arialuni-Bold.xml
        $CMD $FONT_DIR/Arialuni-Italic.ttf Arialuni-Italic.xml
        ```

      - for Windows:

        ```
        %CMD% %FONT_DIR%\Arialuni-Bold.ttf Arialuni-Bold.xml
        %CMD% %FONT_DIR%\Arialuni-Italic.ttf Arialuni-Italic.xml
        ```

   d) Run the script.

      On Linux and OS X, run the command `sh ttfConvert.sh` from the command line. On Windows, run the command `ttfConvert.bat` from the command line or double-click the file `ttfConvert.bat`.

3. Register the font in FOP configuration. (This is not necessary for DITA PDF transformations, skip to the next step)
   a) Create a FOP configuration file that specifies the font metrics file for your font.

```
<fop version="1.0">
  <base>./</base>
  <font-base>file:/C:/path/to/FOP/font/metrics/files/</font-base>
  <source-resolution>72</source-resolution>
  <target-resolution>72</target-resolution>
  <default-page-settings height="11in" width="8.26in"/>
  <renderers>
    <renderer mime="application/pdf">
      <filterList>
        <value>flate</value>
      </filterList>
      <fonts>
          <font metrics-url="Arialuni.xml" kerning="yes"
                embed-url="file:/Library/Fonts/Arialuni.ttf">
              <font-triplet name="Arialuni" style="normal"
                    weight="normal"/>
          </font>
      </fonts>
    </renderer>
  </renderers>
</fop>
```

   The `embed-url` attribute points to the font file to be embedded. Specify it using the URL convention. The `metrics-url` attribute points to the font metrics file with a path relative to the `base` element. The triplet refers to the unique combination of name, weight, and style (italic) for each variation of the font. In our case is just one triplet, but if the font had variants, you would have to specify one for each variant. Here is an example for Arial Unicode if it had italic and bold variants:

```
<fop version="1.0">
  ...
    <fonts>
        <font metrics-url="Arialuni.xml" kerning="yes"
              embed-url="file:/Library/Fonts/Arialuni.ttf">
            <font-triplet name="Arialuni" style="normal"
                  weight="normal"/>
        </font>
        <font metrics-url="Arialuni-Bold.xml" kerning="yes"
              embed-url="file:/Library/Fonts/Arialuni-Bold.ttf">
            <font-triplet name="Arialuni" style="normal"
                  weight="bold"/>
        </font>
        <font metrics-url="Arialuni-Italic.xml" kerning="yes"
              embed-url="file:/Library/Fonts/Arialuni-Italic.ttf">
            <font-triplet name="Arialuni" style="italic"
                  weight="normal"/>
        </font>
    </fonts>
  ...
</fop>
```

   More details about the FOP configuration file are available on the FOP website.
   b) *Open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **XML** > **XSLT/FO/XQuery** > **FO Processors**, and enter the path of the FOP configuration file in the **Configuration file** text field.
4. Set the font on the document content.

   This is usually done with XSLT stylesheet parameters and depends on the document type processed by the stylesheet.

   DocBook Example: For DocBook documents, you can start with the predefined scenario called **DocBook PDF**, *edit the XSLT parameters*, and set the font name (in our example **Arialuni**) to the parameters body.font.family and title.font.family.

   TEI Example: For TEI documents, you can start with the predefined scenario called **TEI PDF**, *edit the XSLT parameters*, and set the font name (in our example **Arialuni**) to the parameters bodyFont and sansFont.

   DITA Example: For DITA to PDF transformations using DITA-OT modify the following two files:

   - *`DITA-OT-DIR`*`/plugins/org.dita.pdf2/cfg/fo/font-mappings.xml` - The `font-face` element included in each element `physical-font` having the `attribute char-set="default"` must contain the name of the font (*Arialuni* in our example)

- *DITA-OT-DIR*/plugins/org.dita.pdf2/fop/conf/fop.xconf - An element font must be inserted in the element fonts, which is inside the element renderer that has the attribute mime="application/pdf":

```
<renderer mime="application/pdf">
  . . .
   <fonts>
       <font metrics-url="Arialuni.xml" kerning="yes"
           embed-url="file:/Library/Fonts/Arialuni.ttf">
           <font-triplet name="Arialuni" style="normal"
               weight="normal"/>
       </font>
   </fonts>
  . . .
</renderer>
```

Adding Libraries to the Built-in FO Processor (XML with XSLT and FO)

Adding Hyphenation Support for XML with XSLT Transformation Scenarios

You can extend the functionality of the built-in FO processor by dropping additional libraries in the *[OXYGEN_INSTALL_DIR]*/lib/fop directory.

To add support for hyphenation, follow these steps:

1. Create a folder called fop in the *[OXYGEN_INSTALL_DIR]*/lib folder.
2. Download the compiled *JAR* from *OFFO*.
3. Copy the fop-hyph.jar file into the *[OXYGEN_INSTALL_DIR]*/lib/fop folder.
4. Restart Oxygen XML Editor.

Adding Support for PDF Images

To add support for PDF images, follow these steps:

1. Create a folder called fop in the *[OXYGEN_INSTALL_DIR]*/lib folder.
2. Download the *fop-pdf-images* *JAR* libraries.
3. Copy the libraries into the *[OXYGEN_INSTALL_DIR]*/lib/fop folder.
4. Restart Oxygen XML Editor.

Adding Libraries to the Built-in FO Processor (DITA-OT)

To use additional libraries with the DITA-OT publishing engine, you need to edit the transformation scenario and add the path to the new libraries in the **Libraries** section of the **Advanced** tab.

Adding Hyphenation Support for DITA-OT Transformation Scenarios

1. Download the pre-compiled *JAR* from *OFFO*.
2. Edit the DITA-OT transformation scenario and switch to the **Advanced** tab. Click the **Libraries** button and add the path to the fop-hyph.jar library.

Adding Support for PDF Images

1. Download the *fop-pdf-images* *JAR* libraries.
2. Edit the DITA-OT transformation scenario and switch to the **Advanced** tab. Click the **Libraries** button and add the path to the libraries.

WebHelp System Output

Oxygen XML Editor allows you to obtain WebHelp Classic and WebHelp Responsive outputs. This section contains information about the WebHelp system, its variants, and ways to customize it to better fit your specific needs.

Table 38: WebHelp System Feature Matrix

Features	WebHelp System Variants				
	Responsive w/ Feedback	Responsive	Classic w/ Feedback	Classic	Classic Mobile
Desktop Systems	✓	✓	✓	✓	
Mobile Devices	✓	✓			✓
Predefined Skins	✓	✓	✓	✓	
Predefined Templates	✓	✓			
Search Capabilities	✓	✓	✓	✓	✓
Modern Layout	✓	✓			
Adaptable to Any Screen Size	✓	✓			
Comments Section	✓		✓		
DITA Documents	✓	✓	✓	✓	✓
DocBook Documents			✓	✓	✓
Tri-Pane Frames or Frameless Version			✓	✓	

WebHelp Responsive System

WebHelp is a form of online help that consists of a series of web pages (XHTML format). Its advantages include platform independence, ability to update content continuously, and it can be viewed using a regular web browser. The Oxygen XML Editor WebHelp system includes several variants to suit your specific needs. The **WebHelp Responsive** variant features a very flexible layout, is designed to adapt to any screen size, and is available for DITA document types.

This type of WebHelp system can be generated by using the ***DITA Map WebHelp Responsive*** *transformation scenario*.

Layout

The layout of the **WebHelp Responsive** system is platform independent and is able to adapt to any screen size. It is highly customizable and relies on a *template mechanism* that allows you to control the position of various functional *template components* to suit your particular requirements.

You can select from several different styles of layouts (for example, by default, you can select either a *tiles* or *tree* style of layout). Furthermore, each of these layouts include a collection of skins that you can choose from, or you can customize your own.

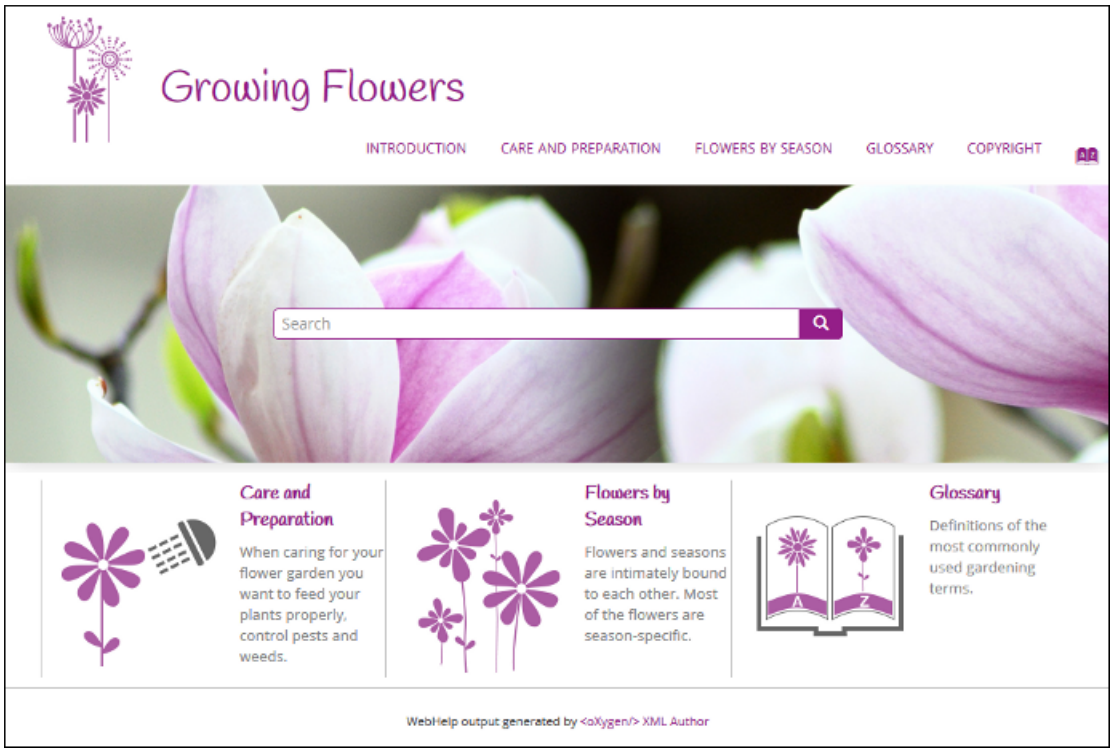


Figure 473: WebHelp Responsive Output on a Normal Screen

Figure 474: WebHelp Responsive Output on a Narrow Screen

WebHelp Responsive Search Engine
Search Rules

Rules that are applied during a search include:

- You can use quotes to perform an exact search for multiple word phrases (for example, *"grow flowers"* will only return results if both words are found consecutively and exactly as they are typed in the search field). This type of search is known as a *phrase search*.
- *Boolean Search* is supported using the following operators: *and*, *or*, *not*. When there are two adjacent search terms without an operator, *or* is used as the default search operator (for example, *grow flowers* is the same as *grow or flowers*).
- The space character separates keywords (an expression such as *grow flowers* counts as two separate keywords: *grow* and *flowers*).
- `indexterm` and `keywords` DITA elements are an effective way to increase the ranking of a page (for example, content inside a *keywords* element weighs more than an *H1* HTML element).
- Words composed by merging two or more words with colon (":"), minus ("-"), underline ("_"), or dot (".") characters count as a single word.
- Always search for words containing three or more characters (shorter words, such as *to* or *of* are ignored). This rule does not apply to CJK (Chinese, Japanese, Korean) languages.

5-Star Rating Mechanism and Sorting

The **Search** feature is also enhanced with a rating mechanism that computes scores for every result that matches the search criteria. These scores are then translated into a 5-star rating scheme and the stars are displayed to the right of each result. The search results are sorted depending on the following:

- Search entries that satisfy the phrase search criterion are presented first.
- The number of keywords found in a single page (the higher the number, the better).
- The context (for example, a word found in a title scores better than a word found in unformatted text). The search ranking order, sorted by relevance is as follows:
  - The search term is included in a meta keyword
  - The search term is in the title of the page
  - The search term is in bold text in a paragraph
  - The search term is in normal text in a paragraph

Tag Element Scoring Values

HTML tag elements are also assigned a scoring value and these values are evaluated for the search results. For information about editing these values, see *Editing Scoring Values of Tag Elements in Search Results* on page 972.

Excluded Terms

To improve performance, the **Search** feature excludes certain *stop words*. For example, the English version of such *stop words* include: *a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with*.

WebHelp Search Results Page

When you enter search terms in the **Search** field, the results are displayed in a results page. When you click on a result, the topic is opened in the main pane and the search results are highlighted. The **Search** field also includes an *autocomplete* feature.

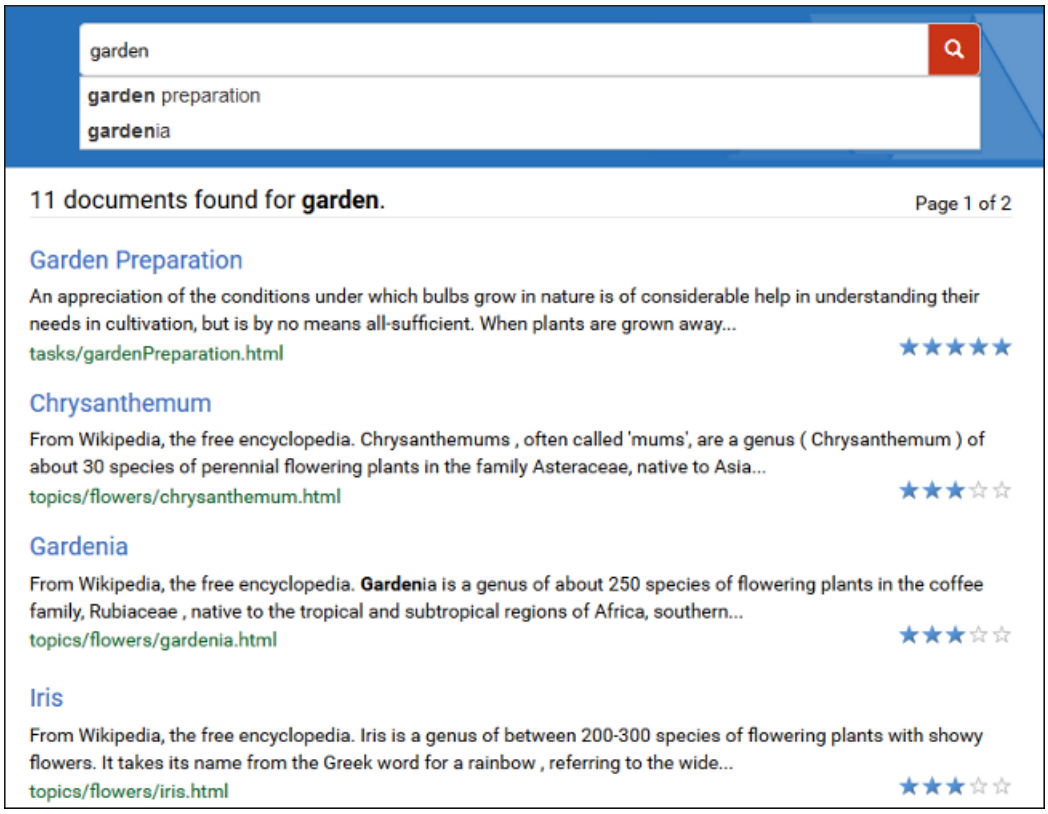


Figure 475: WebHelp Responsive Search Results Page

Autocomplete Suggestions in the Search Text Field

When you are typing in the search text field, proposals are presented to help you to compute the search query. The information proposed when you are typing is collected from:

- The search queries from the history of the previous searches.
- The titles collected from your documentation.
- Documentation index terms and keywords. For example, in a DITA topic, the keywords and index terms are specified in the topic prolog section like this:

```
<prolog>
  <metadata>
 <keywords><indexterm>databases</indexterm></keywords>
 <keyword>installing</keyword>
 <keyword>uninstalling</keyword>
    <keyword>prerequisites</keyword>
  </metadata>
</prolog>
```

Missing Terms

If you enter multiple search terms (other than *stop words*), for any result that the search engine found at least one term but not one or more of the other terms, the **Missing** terms will be listed below each result.

Browser Compatibility

This output format is compatible with the most recent versions of the following common browsers:

- Edge
- Internet Explorer (IE 9 or newer)
- Chrome
- Firefox
- Safari
- Opera

WebHelp Responsive with Feedback System

WebHelp is a form of online help that consists of a series of web pages (XHTML format). Its advantages include platform independence, ability to update content continuously, and it can be viewed using a regular web browser. The Oxygen XML Editor WebHelp system includes several variants to suit your specific needs. The **WebHelp Responsive with Feedback** variant features a very flexible layout, is designed to adapt to any screen size, and includes a feedback system that allows your users to make comments and allows you to manage and reply to them. This variant is available for DITA document types.

This type of WebHelp system can be generated by using the ***DITA Map WebHelp Responsive with Feedback*** *transformation scenario* and following the *instructions for deploying the system*.

Layout

The layout of the **WebHelp Responsive with Feedback** system is platform independent and is able to adapt to any screen size. It is highly customizable and relies on a *template mechanism* that allows you to control the position of various functional *template components* to suit your particular requirements.

You can select from several different styles of layouts (for example, by default, you can select either a *tiles* or *tree* style of layout). Furthermore, each of these layouts include a collection of skins that you can choose from, or you can customize your own.

The **WebHelp Responsive with Feedback** system also contains a **Comments** section at the bottom of the pane. This section is where you can interact with users through a comment system.

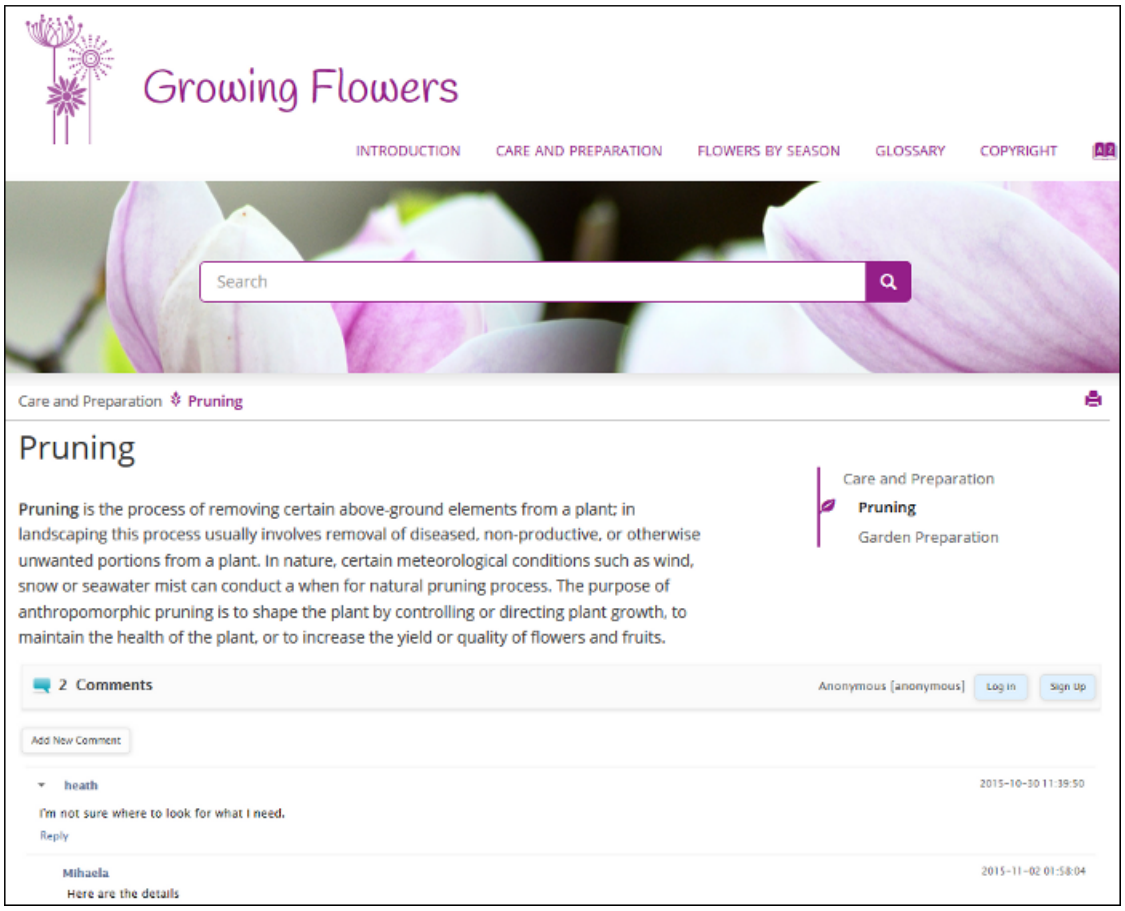


Figure 476: WebHelp Output

Managing Comments

To add a new comment, click the **Add New Comment** button, or click **Reply** to add a comment to an existing thread. You can click on the **Log in** button on the right side of this bar to be authenticated as a user and your user

name will be included in any comments that you add. If you do not have a user name, you can click on the **Sign Up** button to create a new user.

After you log in, your name and user name are displayed in the **Comments** bar, along with the **Log off** and **Edit** buttons. Click the **Edit** button to open the **User Profile** dialog box where you can customize the following options:

- **Your Name** - You can use this field to edit the initial name that you used to create your user profile.
- **Your email address** - You can use this field to edit the initial email address that you used to create your profile.
- You can choose to receive an email in the following situations:
  - When a comment is left on a page that you commented on.
  - When a comment is left on any topic in the WebHelp Classic system.
  - When a reply is left to one of my comments.
- **New Password** - Allows you to enter a new password for your user account.

  Note: The **Current Password** field from the top of the **User Profile** is mandatory if you want to save the changes you make.

If you are an administrator, you can manage user information and comments. For more information, see *Managing Users and Comments in a Feedback-Enabled WebHelp System* on page 949.

WebHelp Responsive Search Engine
Search Rules

Rules that are applied during a search include:

- You can use quotes to perform an exact search for multiple word phrases (for example, *"grow flowers"* will only return results if both words are found consecutively and exactly as they are typed in the search field). This type of search is known as a *phrase search*.
- *Boolean Search* is supported using the following operators: *and*, *or*, *not*. When there are two adjacent search terms without an operator, *or* is used as the default search operator (for example, *grow flowers* is the same as *grow or flowers*).
- The space character separates keywords (an expression such as *grow flowers* counts as two separate keywords: *grow* and *flowers*).
- `indexterm` and `keywords` DITA elements are an effective way to increase the ranking of a page (for example, content inside a *keywords* element weighs more than an *H1* HTML element).
- Words composed by merging two or more words with colon (":"), minus ("-"), underline ("_"), or dot (".") characters count as a single word.
- Always search for words containing three or more characters (shorter words, such as *to* or *of* are ignored). This rule does not apply to CJK (Chinese, Japanese, Korean) languages.

5-Star Rating Mechanism and Sorting

The **Search** feature is also enhanced with a rating mechanism that computes scores for every result that matches the search criteria. These scores are then translated into a 5-star rating scheme and the stars are displayed to the right of each result. The search results are sorted depending on the following:

- Search entries that satisfy the phrase search criterion are presented first.
- The number of keywords found in a single page (the higher the number, the better).
- The context (for example, a word found in a title scores better than a word found in unformatted text). The search ranking order, sorted by relevance is as follows:
  - The search term is included in a meta keyword
  - The search term is in the title of the page
  - The search term is in bold text in a paragraph
  - The search term is in normal text in a paragraph

Tag Element Scoring Values

HTML tag elements are also assigned a scoring value and these values are evaluated for the search results. For information about editing these values, see *Editing Scoring Values of Tag Elements in Search Results* on page 972.

Excluded Terms

To improve performance, the **Search** feature excludes certain *stop words*. For example, the English version of such *stop words* include: *a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with*.

WebHelp Search Results Page

When you enter search terms in the **Search** field, the results are displayed in a results page. When you click on a result, the topic is opened in the main pane and the search results are highlighted. The **Search** field also includes an *autocomplete* feature.

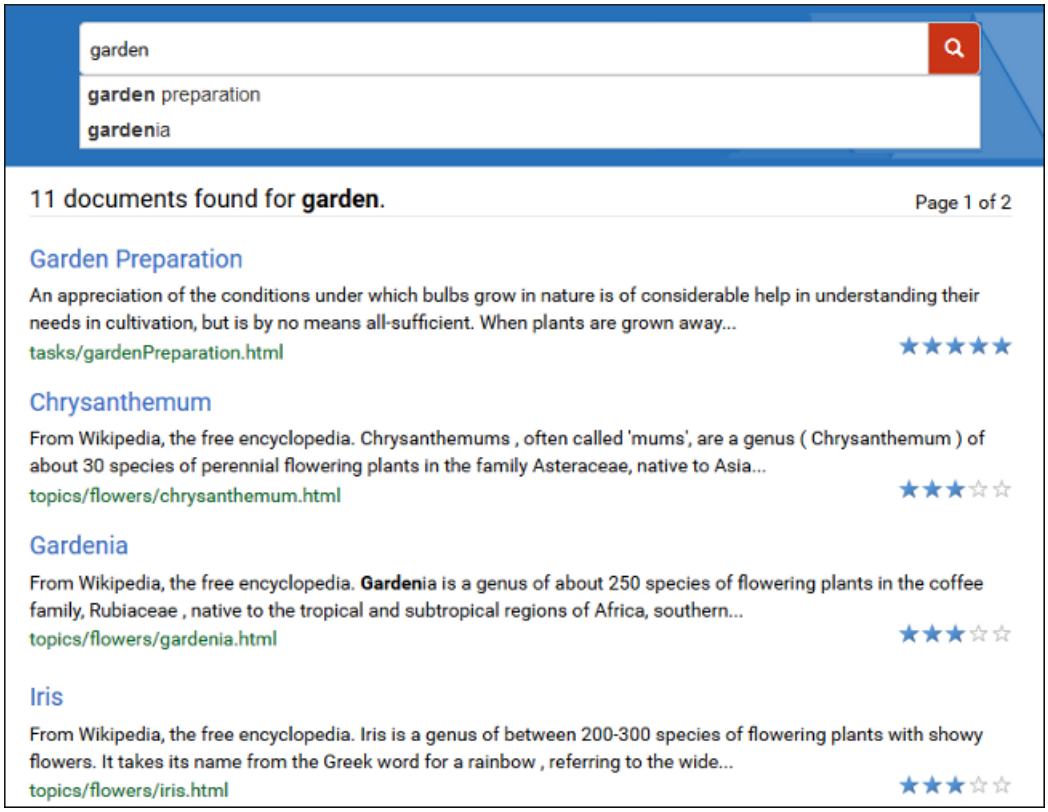


Figure 477: WebHelp Responsive Search Results Page

Autocomplete Suggestions in the Search Text Field

When you are typing in the search text field, proposals are presented to help you to compute the search query. The information proposed when you are typing is collected from:

- The search queries from the history of the previous searches.
- The titles collected from your documentation.
- Documentation index terms and keywords. For example, in a DITA topic, the keywords and index terms are specified in the topic prolog section like this:

```
<prolog>
  <metadata>
 <keywords><indexterm>databases</indexterm></keywords>
 <keyword>installing</keyword>
 <keyword>uninstalling</keyword>
    <keyword>prerequisites</keyword>
  </metadata>
</prolog>
```

Missing Terms

If you enter multiple search terms (other than *stop words*), for any result that the search engine found at least one term but not one or more of the other terms, the **Missing** terms will be listed below each result.

Browser Compatibility

This output format is compatible with the most recent versions of the following common browsers:

- Edge
- Internet Explorer (IE 9 or newer)
- Chrome
- Firefox
- Safari
- Opera

Deploying a Feedback-Enabled WebHelp System

System Requirements

The feedback-enabled WebHelp system of Oxygen XML Editor requires a standard server deployment. You can request this from your server admin and it needs the following system components:

- A Web server (such as *Apache Web Server*)
- A *MySQL* or *MariaDB* database server
- A database admin tool (such as *phpMyAdmin*)
- PHP Version 5.1.6 or later

Oxygen XML WebHelp system supports most of the recent versions of the following browsers: Chrome, Firefox, Edge, Internet Explorer, Safari, Opera.

Create WebHelp with Feedback Database

The **WebHelp with Feedback** system needs a database to store user details and the actual feedback, and a user added to it with all privileges. After this is created, you should have the following information:

- Database name
- Username
- Password

Exactly how you create the database and user depends on your web host and your particular needs.

Example:

The following procedure uses *phpMyAdmin* to create a MySQL database for the feedback system and a MySQL user with privileges for that database. The feedback system uses these credentials to connect to the database.

Using *phpMyAdmin* to create a database:

1. Access the *phpMyAdmin* instance running on your server.
2. Click *Databases* (in the right frame) and then create a *database*. You can give it any name you want (for example *comments*).
3. Create a user with connection privileges for this database.
4. Under *localhost*, in the right frame, click *Privileges* and then at the bottom of the page click the **reload the privileges** link.

Deploying the WebHelp with Feedback Output

If you have a web server configured with PHP and MySQL, you can deploy the **WebHelp with Feedback** output by following these steps:

1. Connect to your server using an FTP client.
2. Locate the home directory (from now on, referred to as *DOCUMENT_ROOT*) of your server.
3. Copy the transformation output folder into the `DOCUMENT_ROOT` folder.
4. Rename it to something relevant (for example, `myProductWebHelp`).
5. Open the output folder (for example, `http://[YOUR_SERVER]/myProductWebHelp/`). You are redirected to the installation wizard. Proceed with the installation as follows:
   a. Verify that the prerequisites are met.
   b. Press **Start Installation**.

c. Configure the **Deployment Settings** section. Default values are provided, but you should adjust them as needed.

   Tip: You can change some of the options later. The installation creates a `config.php` file in `[OXYGEN_WEBHELP_INSTALL_DIR]/feedback/resources/php/config/config.php` where all your configuration options are stored.

d. Configure the **MySql Database Connection Settings** section. Use the information (database name, username, password) from the *Create WebHelp with Feedback Database section* to fill-in the appropriate text boxes.

   Warning: Selecting the **Create new database structure** option will overwrite any existing data in the selected database, if it already exists. Therefore, it is useful the first time you install the **WebHelp with Feedback** system, but you do not want to select this option on subsequent deployments.

e. If you are using a domain (such as *OpenLDAP* or *Active Directory*) to manage users in your organization, select the **Enable LDAP Autehntication** option. This will allow you to configure the LDAP server, which will provide information and credentials for users who will access the WebHelp system. Also, this will allow you to choose which of the domain users will have administrator privileges.

f. If the **Create new database structure** option is selected, the **Create WebHelp Administrator Account** section becomes available. Here you can set the administrator account data. The administrator is able to moderate new posts and manage WebHelp users.

   The same database can be used to store comments for multiple **WebHelp with Feedback** deployments. If a topic is available in multiple deployments and there are comments associated with it, you can choose to display the comments in all deployments that share the database. To do this, select the **Display comments from other products** option. In the **Display comments from** section, a list with the deployments sharing the same database is displayed. Select the deployments allowed to share common feedback.

   Note: You can restrict the displayed comments of a product depending on its version. If you have two products that use the same database and you restrict one of them to display comments starting from a certain version, the comments of the other product are also displayed from the specified version onwards.

g. Press **Next Step**.

h. Remove the installation folder from your web server.

   Important: When you publish subsequent iterations of your **WebHelp with Feedback** system, you will not upload the `/install` folder in the output, as you only need it uploaded the first time you create the installation. On subsequent uploads, you will just upload the other output files.

i. In your Web browser, go to your **WebHelp with Feedback** system main page.

Testing Your WebHelp with Feedback System

To test your system, create a user and post a comment. Check to see if the notification emails are delivered to your email inbox.

Note: To read debug messages generated by the system:

1. Enable *JavaScript* logging by doing one of the following:
   - Open the `log.js` file, locate the `var log= new Log(Level.NONE);` line, and change the logging level to: `Level.INFO`, `Level.DEBUG`, `Level.WARN`, or `Level.ERROR`.
   - Append `?log=true` to the WebHelp URL.
2. Inspect the PHP and Apache server log files.

Documentation Product ID and Version

When you run a **WebHelp with Feedback** transformation scenario, by default you are prompted for a documentation product ID and version number. This is needed when multiple WebHelp systems are deployed on the same server. Think of your WebHelp output as a *product*. If you have three different WebHelp outputs, you have three different *products* (each with their own unique documentation product ID). This identifier is included in a configuration file so that comments are tied to a particular output (product ID and version number).

Note: The **WebHelp with Feedback** installation includes a configuration option (**Display comments from other products**) that allows you to choose to have comments visible in other specified *products*.

Related Information:

Managing Users and Comments in a Feedback-Enabled WebHelp System on page 949

Refreshing the Content of a Feedback-Enabled WebHelp System

It is common to update the content of an existing installation of a **WebHelp with Feedback** system on a regular basis. In this case, reinstalling the whole system is not a viable option since it might result in the loss of the comments associated with your topics. Also, reconfiguring the system every time you want to refresh it may be time consuming.

Fortunately, you can refresh just the content without losing the comments or the initial system configuration. To do so, follow these steps:

1. Execute the transformation scenario that produces the *WebHelp with Feedback* output directory.
2. Go to the output directory (specified in the **Output** tab of the transformation scenario), locate the `\feedback\resources\php\config\config.php` file, and delete it.
3. Locate the `\feedback\install` directory and delete it.
4. Copy the remaining structure of the output folder and paste it into your *WebHelp with Feedback* system installation directory, overwriting the existing content.

Managing Users and Comments in a Feedback-Enabled WebHelp System

When you installed the **WebHelp with Feedback** system the first time (assuming the ***Create new database structure*** *option* was selected), you should have been prompted to create an administrator account (or a user named `administrator` was created by default). As an administrator, you have access to manage comments posted in your feedback-enabled WebHelp system. You can also manage the user information (such as role, status, or notification options).

To manage comments and user information, follow these steps:

1. At the bottom of each specific topic there is a **Comments** navigation bar and on the right side there is a **Log in** button. Click this button and log in with your administrator credentials. This gives you access to an **Admin Panel** button.
2. Click the **Admin Panel** button to display an administration page.

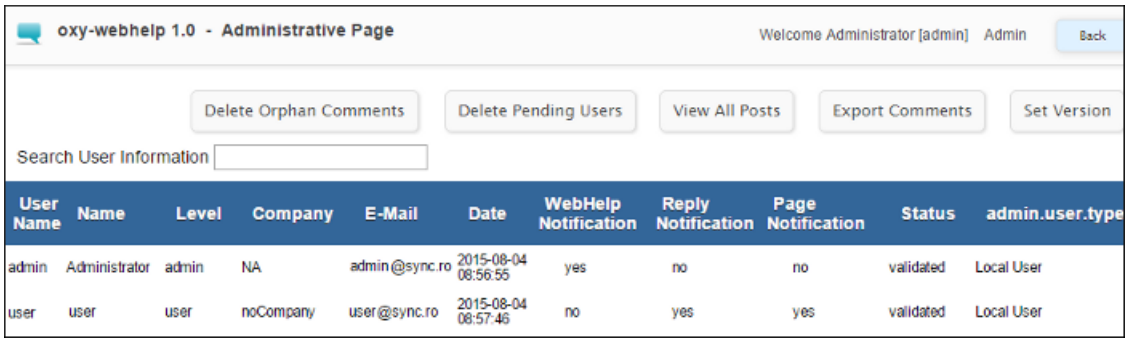


Figure 478: Administrative Page

3. Use this page to manage the following options:

   Delete Orphaned Comments

   Allows you to delete comments that are no longer associated with a topic in your WebHelp system.

   Delete Pending Users

   Allows you to delete user accounts that you do not wish to activate.

   View All Posts

   Allows you to view all the comments that are associated with topics in your WebHelp system.

   Export Comments

   Allows you to export all posts associated with topics in your WebHelp system into an XML file.

   Set Version

   Use this action to display comments starting with a particular version.

Manage User Information

To edit the details for a user, click on the corresponding row. This opens a window that allows you to customize the following information associated with the user:

Name

The full name of the user.

Level

Use this field to modify the privilege level (role) for the selected user. You can choose from the following:

- **User** - Regular user, able to post comments and receive e-mail notifications.
- **Moderator** - In addition to the regular **User** rights, this type of user has access to the **Admin Panel** where a moderator can view, delete, export comments, and set the version of the feedback-enabled WebHelp system.
- **Admin** - Full administrative privileges. Can manage WebHelp-specific settings, users, and their comments.

Company

The name of the organization associated with the user.

E-Mail

The contact email address for the user. This is also the address where the WebHelp system sends notifications.

WebHelp Notification

When selected, the user receives notifications when comments are posted anywhere in your feedback-enabled WebHelp system.

Reply Notification

When selected, the user receives notifications when comments are posted as a reply to one of their comments.

Page Notification

When selected, the user receives notifications when comments are posted on a topic where they previously posted a comment.

Date

The date the user registered is displayed.

Status

Use this drop-down list to change the status of the user. You can choose from the following:

- **Created** - The user is created but does not yet have any rights for the feedback-enabled WebHelp system.
- **Validated** - The user is able to use the feedback-enabled WebHelp system.
- **Suspended** - The user has no rights for the feedback-enabled WebHelp system.

Warning: The key used for identifying the page a comment is attached to is the relative file path to the output page. Since the output file and folder names mirror the source, any change to the file name (or its folder) in the source will affect the comments associated with that WebHelp page. If you change the file name or path, the comment history for that topic will become orphaned (a change to the topic ID does not affect the comment history).

WebHelp Responsive Template Mechanism

The WebHelp Responsive template mechanism is the base of the system and it is responsible for defining its output. It consists of a set of HTML template files and other additional resources (such as images, CSS, and JavaScript files). Each of the HTML template files contain one or more template components (such as title, table of contents, search input, etc.) whose placement inside the template will define the final layout of the output.

This mechanism allows you to create multiple layouts simply by creating templates that define the location of where the various components will be displayed.

Creating WebHelp Responsive Templates

To create a new WebHelp Responsive template, follow these steps:

1. Locate the following folder in your DITA-OT directory (*DITA-OT-DIR*):

   ```
   DITA-OT-DIR/plugins/com.oxygenxml.webhelp/templates/dita/
   ```

2. Duplicate the `bootstrap` folder and rename it to whatever you want your new template to be identified as (for example, `myTemplate`).
3. Customize the structure of the new template according to your needs. For example, if you only want to keep one of the template variants, open the `myTemplate/variants` folder and delete all of its subdirectories, except for that one (for instance, the `tiles` directory). Keep in mind that the structure of the template directory is important. The names of folders at certain levels correspond to the names of templates and skins, while components and resources are defined and referenced in certain files or folders at specific locations within the directory structure. For more information, see *WebHelp Responsive Template Directory Structure* on page 951.
4. You can also customize the structure of the skins within the template variants. For example, if you only want to keep one of the skins in the `tiles` variant, open the `myTemplate/variants/tiles` folder and delete all of its subdirectory skins, except for that one (for instance, the `light` directory). You can also edit the `skin.css` file that is located in the skin directory to customize the styling. If your customization of the CSS file requires additional resources (such as images, fonts, or other CSS files), they need to be placed in the `resources` folder at the same level with the `skin.css` file.
5. To customize the components that appear in the WebHelp Responsive output, you can modify the HTML template files that define the output. For more information, see *WebHelp Responsive Template Files* on page 953.

Related Information:

Customizing the WebHelp Responsive Output on page 963

WebHelp Responsive Template Directory Structure

A certain directory structure is required for the WebHelp Responsive templates. The names of folders at certain levels correspond to the names of template variants and skins, components are defined in specific files, and various resources need to be located in specific locations within the directory structure.

The templates are stored in *DITA-OT-DIR*`/plugins/com.oxygenxml.webhelp/templates/dita`.

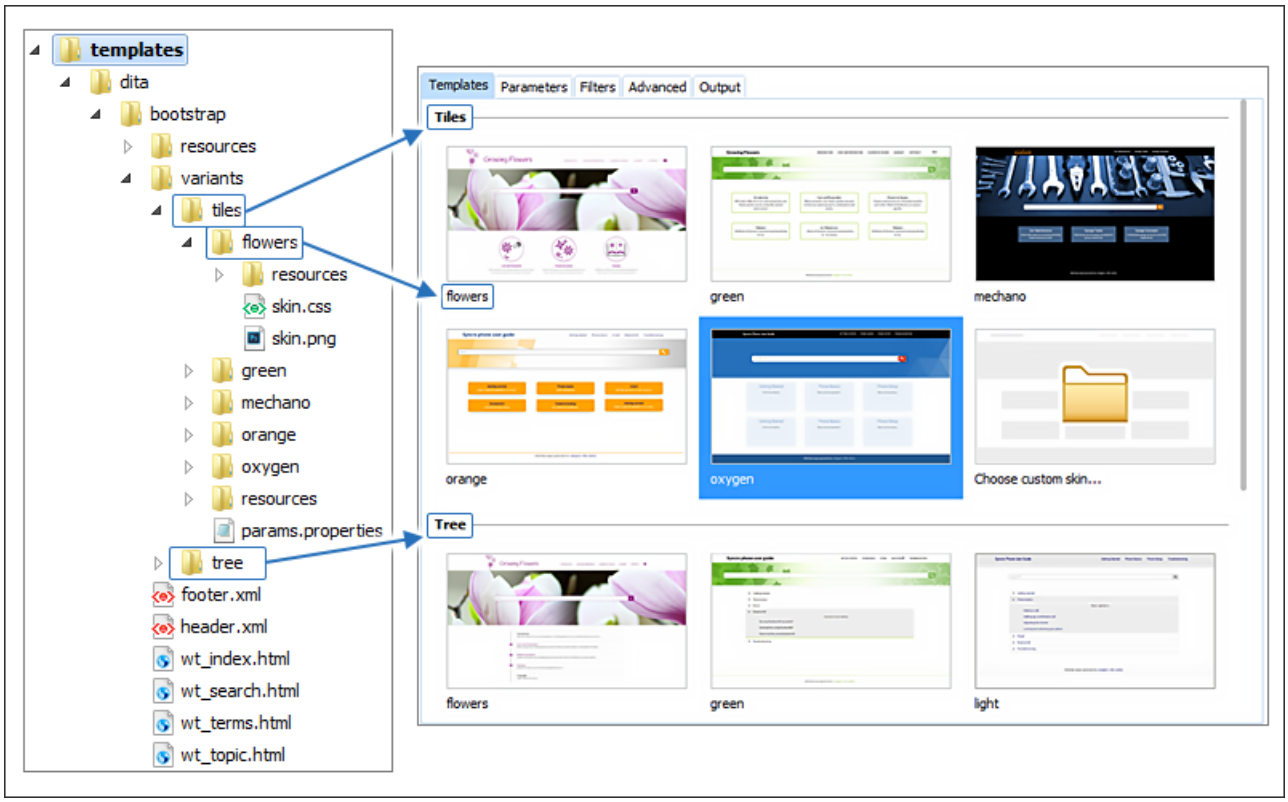


Figure 479: Templates Directory Structure

At the first level of the template directory we can find the following predefined files and folders:

- **`resources` Folder** - Contains all additional resources used by the template, such as images, CSS, and JavaScript files.
- **`variants` Folder** - Contains the template variants, including folders for each skin. See *Template Variants and Skins* on page 952.
- **Template Files** - The HTML template files that are used to generate the output:
  - `wt_index.html` - Used to produce the main home page of the WebHelp Responsive output. See *WebHelp Responsive Main Page Template* on page 953.
  - `wt_topic.html` - Used to generate the HTML pages associated with individual topics. See *WebHelp Responsive Topic Template* on page 955.
  - `wt_search.html` - Used to generate the HTML page that presents the search results. See *WebHelp Responsive Search Results Template* on page 956.
  - `wt_terms.html` - Used to generate the HTML page that presents the documentation index. See *WebHelp Responsive Index Terms Template* on page 956.

After the transformation scenario is executed, the resources and variants folders are copied in the `/oxygen-webhelp/template/` folder within the output directory (defined in the **Output** tab of the transformation scenario).

Template Variants and Skins

You could think of a *template* as being a set of WebHelp components that are placed in a predefined HTML layout. You can have multiple variants of the template. A WebHelp *template variant* is an instance of the template with a specific set of parameters. For example, you could have two variants of the WebHelp main page, one that displays the topics in a *tiles* style of layout, and another one that displays the topics in a *tree* style.

Each variant has its own directory that corresponds to the name of the variant. The name of the variant is displayed in the user interface when the variants are displayed (for example, in the **templates** tab of the transformation scenario).

Each variant directory may contain the following resources:

- **Skin Directories** - These folders represent skin templates for the current variant.
- **`params.properties` File** - This file specifies the values for the parameters imposed by the variant.
- **`resources` Folder** - This is an optional directory that contains resources that are specific to the current variant (such as images, CSS files, etc.) They will be copied to the output directory.

Skins

A variant *skin* represents a CSS file that allows you to alter the styling of the template. The CSS associated with a skin must be named **`skin.css`** and it must be stored as a first child of the skin directory.

Each skin might need additional resources (images, fonts) that must be stored in the `resources` directory in the root folder for that particular skin. The name of the skin directory is displayed in the user interface when you choose a skin (in the **templates** tab of the transformation scenario).

Each skin directory can also contain a **`skin.png`** preview image that will be displayed in the user interface and a properties file that contains a URL for the online preview of the skin. This image file must be stored as a first child of the skin directory.

For information about creating or customizing skins, see *Create or Customize a WebHelp Responsive Skin* on page 965.

WebHelp Responsive Template Files

The HTML pages that comprise the output of a WebHelp Responsive system are obtained after processing HTML template files. These files correspond to the type of page they generate in the output.

There are four types of template pages and corresponding HTML template files:

1. **Main Page Template** (`wt_index.html` file) - Used to produce the main home page of the WebHelp Responsive output.
2. **Topic Template** (`wt_topic.html` file) - Used to generate the HTML pages associated with individual topics.
3. **Search Results Template** (`wt_search.html` file) - Used to generate the HTML page that presents the search results.
4. **Index Terms Template** (`wt_terms.html` file) - Used to generate the HTML page that presents the documentation index.

The HTML template files are located in the following directory:

```
DITA-OT-DIR/plugins/com.oxygenxml.webhelp/templates/dita/bootstrap/
```

WebHelp Responsive Main Page Template

The *Main Page Template* is used to generate the home page of the WebHelp Responsive output. The name of the template file is `wt_index.html` and it is located in the following directory:

```
DITA-OT-DIR/plugins/com.oxygenxml.webhelp/templates/dita/bootstrap/
```

The main function of the home page is to display top level information and provide links that help you easily navigate to any of the top level topics of the publication. These links can be rendered in either a *Tiles* or *Tree* style of layout. The HTML page produced for the home page also consists of various other components, such as a logo, title, menu, search field, or index link.

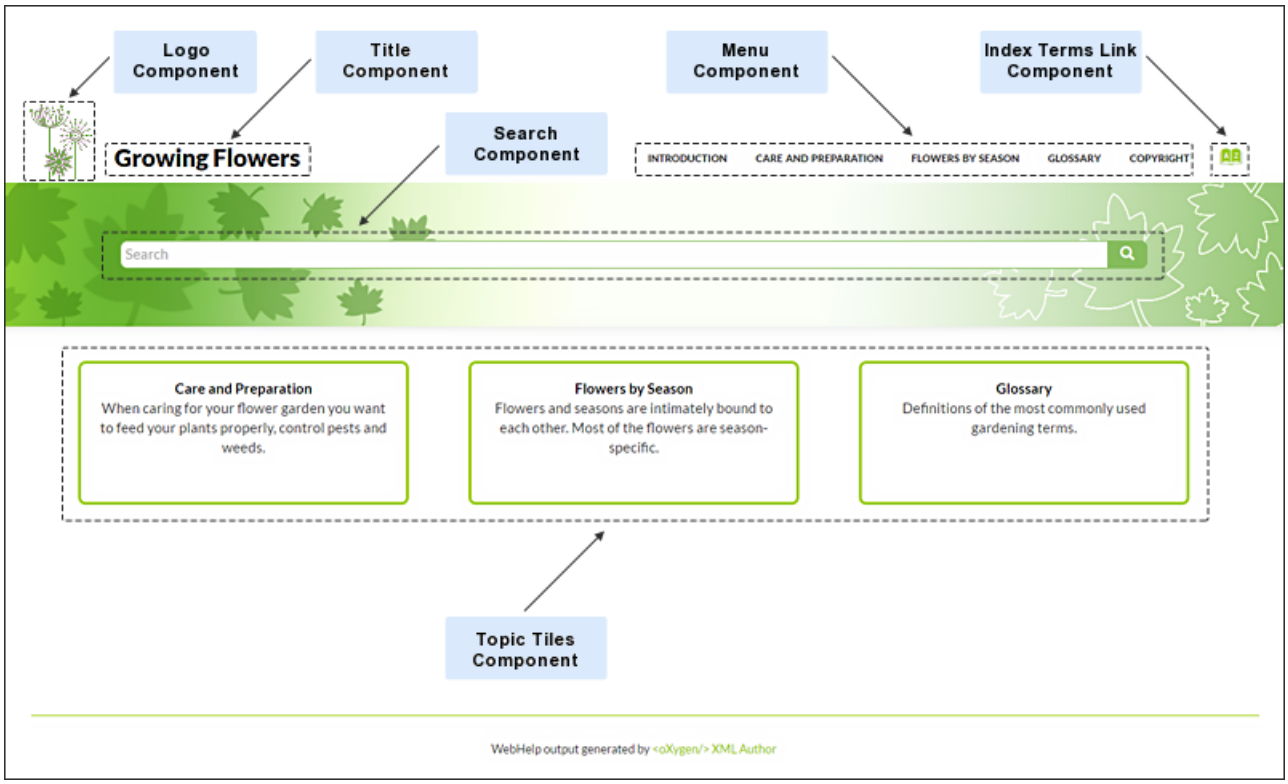


Figure 480: Examples of Main Page Components for a Tiles Style of Layout

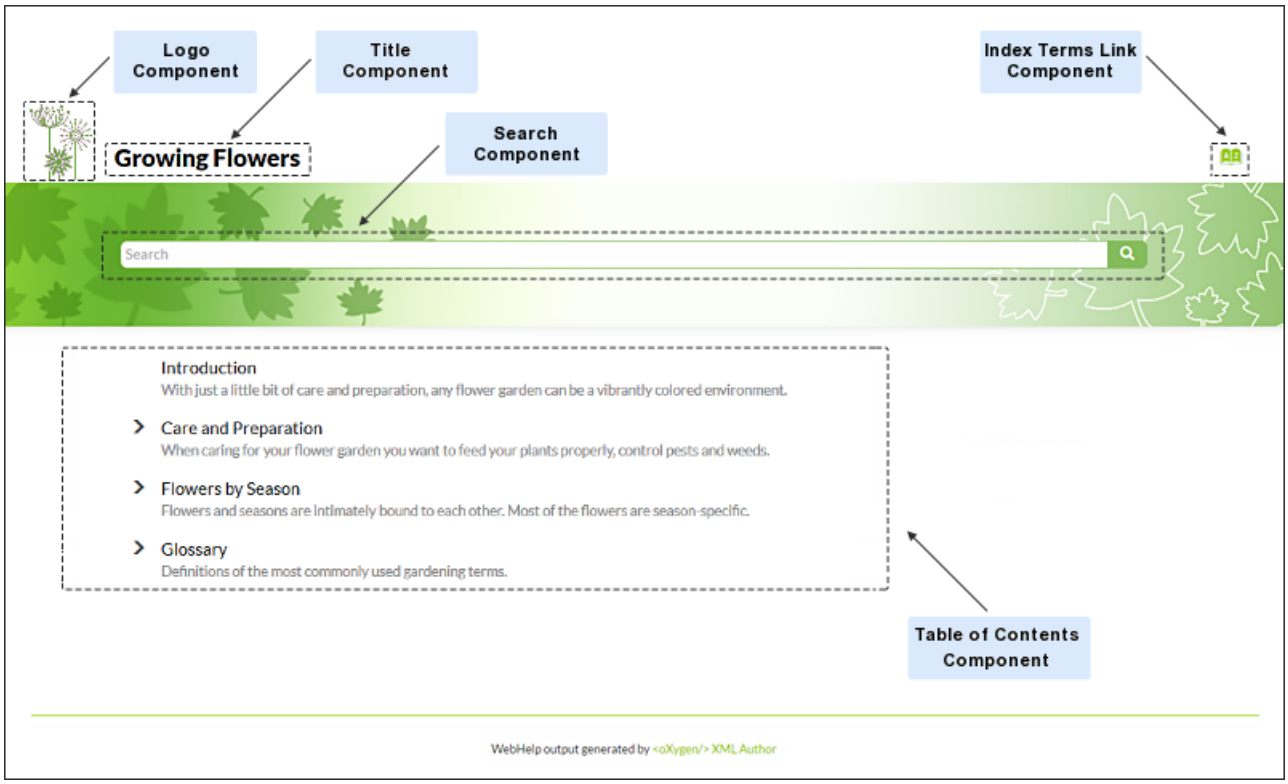


Figure 481: Examples of Main Page Components for a Tree Style of Layout

The components that can be referenced in the `wt_index.html` template file are as follows:

- *Publication Title*
- *Publication Logo*
- *Search Input*

- *Print Link*
- *Main Page Menu*
- *Main Page Topic Tiles*
- *Main Page Table of Contents*
- *Index Terms Link*
- *Link to Skin Resources*

WebHelp Responsive Topic Template

The *Topic Template* is used to generate HTML pages for each topic in the WebHelp Responsive output. The name of the template file is `wt_topic.html` and it is located in the following directory:

```
DITA-OT-DIR/plugins/com.oxygenxml.webhelp/templates/dita/bootstrap/
```

The HTML pages produced for each topic consist of the topic content along with various other additional components, such as a title, menu, navigation breadcrumb, print icon, or side table of contents.

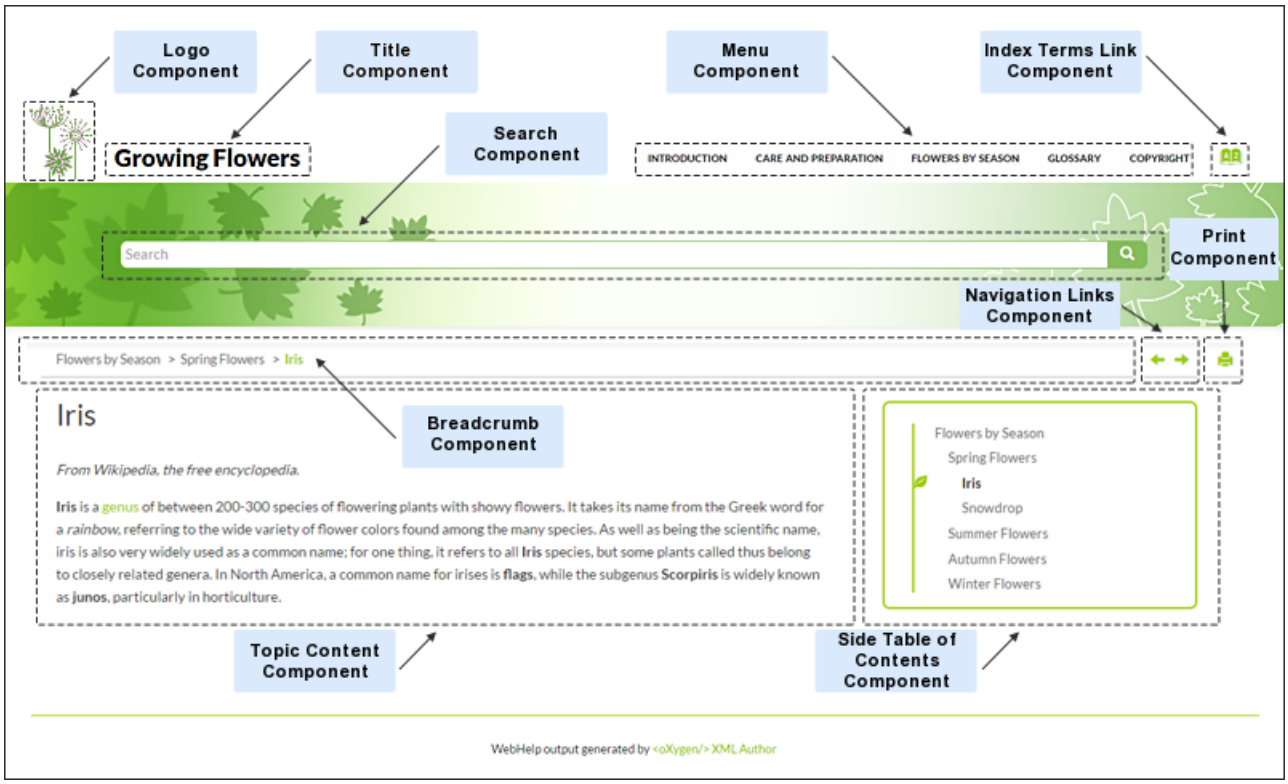


Figure 482: Examples of Topic Page Components

The components that can be referenced in the `wt_topic.html` template file are as follows:

- *Publication Title*
- *Publication Logo*
- *Search Input*
- *Topic Breadcrumb*
- *Navigational Links*
- *Print Link*
- *Topic Content*
- *Topic Side TOC*
- *Topic Feedback*
- *Main Page Menu*
- *Index Terms Link*
- *Child Links*
- *Related Links*

- *Link to Skin Resources*

WebHelp Responsive Search Results Template

The *Search Results Template* is used to generate HTML pages that present search results in the WebHelp Responsive output. The name of the template file is `wt_search.html` and it is located in the following directory:

```
DITA-OT-DIR/plugins/com.oxygenxml.webhelp/templates/dita/bootstrap/
```

The HTML page that is produced consists of a search results component along with various other additional components, such as a title, menu, or index link.

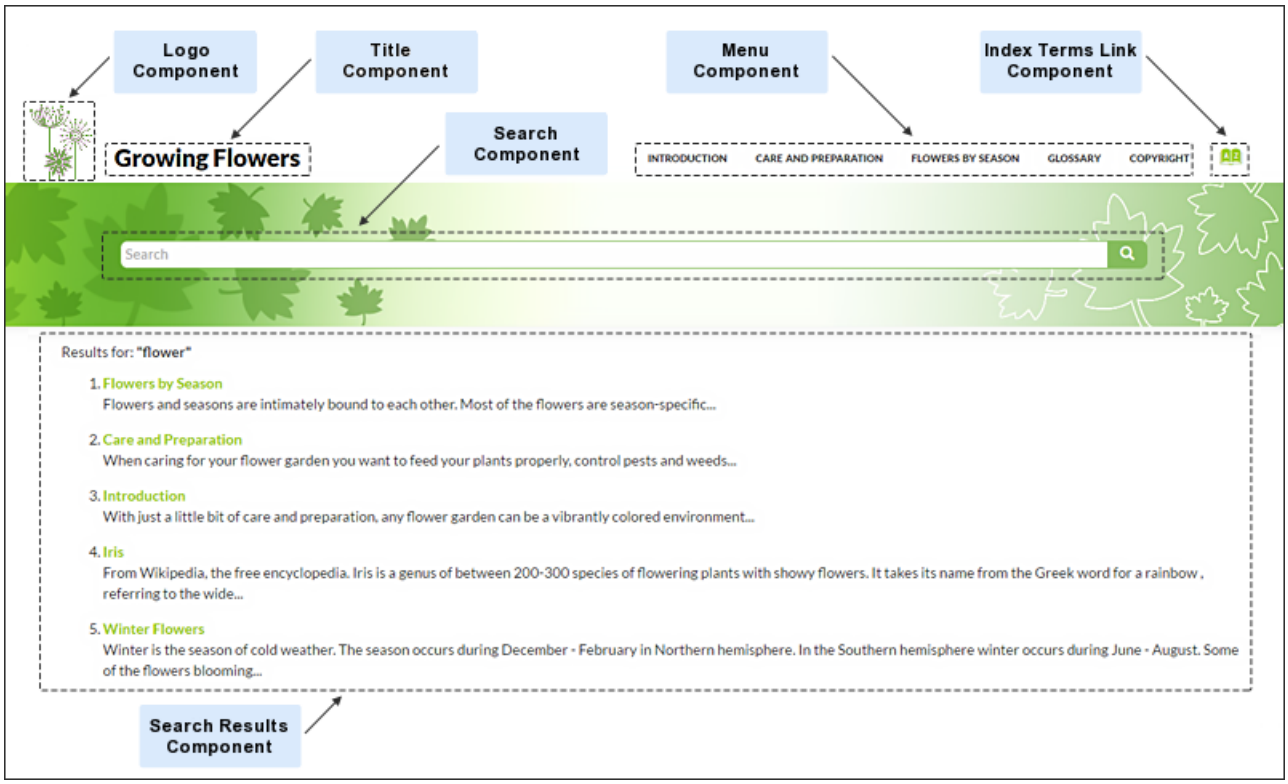


Figure 483: Examples of Search Results Page Components

The components that can be referenced in the `wt_search.html` template file are as follows:

- *Publication Title*
- *Publication Logo*
- *Search Input*
- *Search Results*
- *Print Link*
- *Main Page Menu*
- *Index Terms Link*
- *Link to Skin Resources*

WebHelp Responsive Index Terms Template

The *Index Terms Template* is used to generate HTML pages that present index terms in the WebHelp Responsive output. The name of the template file is `wt_terms.html` and it is located in the following directory:

```
DITA-OT-DIR/plugins/com.oxygenxml.webhelp/templates/dita/bootstrap/
```

The HTML page that is produced consists of an index terms section along with various other additional components, such as a title, menu, or search field.

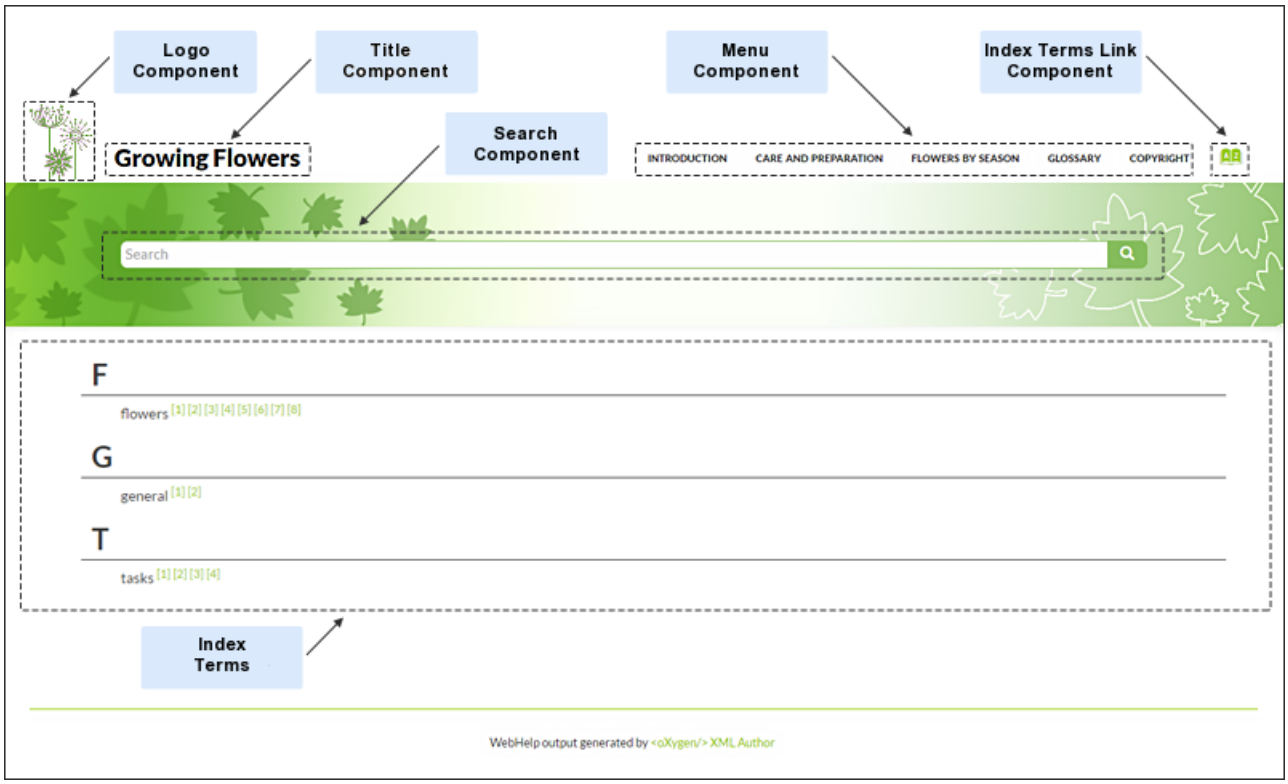


Figure 484: Examples of Index Terms Page Components

The components that can be referenced in the `wt_terms.html` template file are as follows:

- *Publication Title*
- *Publication Logo*
- *Search Input*
- *Print Link*
- *Main Page Menu*
- *Index Terms Link*
- *Link to Skin Resources*

WebHelp Responsive Template Components

A WebHelp Responsive *template component* adds dynamics to a WebHelp *template page*. The rendering of a component depends on the context of where it is placed and its content depends on the transformed *DITA map*.

Some of the template components can be used in all the *types of template pages*, while some are only available for certain template pages. For instance, the *Publication Title* component can be used in all pages, but the *Navigation Breadcrumb* component can only be used in *Topic Template* pages.

To include a *template component* in the output of a particular type of *template page*, you have to reference a specific element in the particular *template file*. All the elements associated with a template component should belong to the `http://www.oxygenxml.com/webhelp/components` namespace.

Every component could contain custom content or reference another component. To specify where the component content will be located in the output you can use the `component_content` element as a descendant of the component element.

Example:

```
<whc:webhelp_search_input class="navbar-form wh_main_page_search"
              role="form" >
        <whc:include_html href="{$webhelp.include.before.search}"/>
  <div class="search_input">
   <whc:component_content/>
  </div>
</whc:webhelp_search_input>
```

The various components and where they can be used are as follows:

Publication Title [webhelp_publication_title]

This component generates the publication title in the output. To generate this component, the `webhelp_publication_title` element must be specified in the template file. It can be used in all four types of template pages:

- ***Main Page Template*** *(`wt_index.html` file)*
- ***Topic Template*** *(`wt_topic.html` file)*
- ***Search Results Template*** *(`wt_search.html` file)*
- ***Index Terms Template*** *(`wt_terms.html` file)*

In the template file, the `webhelp_publication_title` element can be specified as in the following example:

```
<whc:webhelp_publication_title
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_publication_title`.

Publication Logo [webhelp_logo]

This component generates a logo image in the output. To generate this component, the `webhelp_logo` element must be specified in the template file. It can be used in all four types of template pages:

- ***Main Page Template*** *(`wt_index.html` file)*
- ***Topic Template*** *(`wt_topic.html` file)*
- ***Search Results Template*** *(`wt_search.html` file)*
- ***Index Terms Template*** *(`wt_terms.html` file)*

In the template file, the `webhelp_logo` element can be specified as in the following example:

```
<whc:webhelp_logo
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In addition, you must also specify the path of the logo image in the `webhelp.logo.image` transformation parameter (in the **Parameters** tab in the transformation scenario). You can set the `webhelp.logo.image.target.url` parameter to generate a link to a URL when you click the logo image. If this parameter is not set, a link to the home page will automatically be generated.

In the output, you will find an element with the class: `wh_logo`.

Search Input [webhelp_search_input]

This component is used to generate the input widget associated with search function in the output. To generate this component, the `webhelp_search_input` element must be specified in the template file. It can be used in all four types of template pages:

- ***Main Page Template*** *(`wt_index.html` file)*
- ***Topic Template*** *(`wt_topic.html` file)*
- ***Search Results Template*** *(`wt_search.html` file)*
- ***Index Terms Template*** *(`wt_terms.html` file)*

In the template file, the `wh_search_input` element can be specified as in the following example:

```
<whc:webhelp_search_input
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_search_input`.

Search Results [webhelp_search_results]

This component is used to generate a placeholder to signal where the search results will be presented in the output. To generate this component, the `webhelp_search_results` element must be specified in the template file. It can be used in the following type of template page:

- ***Search Results Template*** *(`wt_search.html` file)*

In the template file, the `webhelp_search_results` element can be specified as in the following example:

```
<whc:webhelp_search_results
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_search_results`.

Topic Breadcrumb [webhelp_breadcrumb]

This component generates a breadcrumb that displays the path of the current topic. To generate this component, the `webhelp_breadcrumb` element must be specified in the template file. It can be used in the following type of template page:

- ***Topic Template*** *(`wt_topic.html` file)*

In the template file, the `webhelp_breadcrub` element can be specified as in the following example:

```
<whc:webhelp_breadcrumb
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_breadcrumb`. This element will contain a list with items that correspond to the topics in the path. The first item in the list has a link to the main page with the home class. The last item in the list corresponds to the current topic and has the `active` class set.

Navigational Links [webhelp_navigation_links]

This component generates navigation links to the next and previous topics. To generate this component, the `webhelp_navigation_links` element must be specified in the template file. It can be used in the following type of template page:

- ***Topic Template*** *(`wt_topic.html` file)*

In the template file, the `webhelp_navigation_links` element can be specified as in the following example:

```
<whc:webhelp_navigation_links
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_navigation_links`. This element will contain the links to the next and previous topics.

Topic Content [webhelp_topic_content]

This component generates the content of a topic and it represent the content of the HTML files as they are produced by the DITA-OT processor. To generate this component, the `webhelp_topic_content` element must be specified in the template file. It can be used in the following type of template page:

- ***Topic Template*** *(`wt_topic.html` file)*

In the template file, the `webhelp_topic_content` element can be specified as in the following example:

```
<whc:webhelp_topic_content
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_topic_content`.

Topic Side TOC [webhelp_side_toc]

This component generates a mini table of contents for the current topic. It will contain links to the children of current topic, its siblings, and all of its ancestors. To generate this component, the `webhelp_side_toc` element must be specified in the template file. It can be used in the following type of template page:

- ***Topic Template*** *(`wt_topic.html` file)*

In the template file, the `webhelp_side_toc` element can be specified as in the following example:

```
<whc:webhelp_side_toc
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_side_toc`. This element will contain links to the topics that are close to the current topic.

Topic Feedback [webhelp_feedback]

This component generates a placeholder for where the comments section will be presented. To generate this component, the `webhelp_feedback` element must be specified in the template file. It can be used in the following type of template page:

- ***Topic Template*** *(`wt_topic.html` file)*

In the template file, the `webhelp_feedback` element can be specified as in the following example:

```
<whc:webhelp_feedback
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

Child Links [webhelp_child_links]

For all topics with subtopics (child topics), this component generates a list of links to each child topic. To generate this component, the `webhelp_child_links` element must be specified in the template file. It can be used in the following type of template page:

- ***Topic Template*** *(`wt_topic.html` file)*

In the template file, the `webhelp_child_links` element can be specified as in the following example:

```
<whc:webhelp_child_links
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

Related Links [webhelp_related_links]

For all topics that contain related links, this component generates a list of related links that will appear in the output. To generate this component, the `webhelp_related_links` element must be specified in the template file. It can be used in the following type of template page:

- ***Topic Template*** *(`wt_topic.html` file)*

In the template file, the `webhelp_related_links` element can be specified as in the following example:

```
<whc:webhelp_related_links
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

Main Page Topic Tiles [webhelp_tiles]

This component generates the tiles section in the main page. This section will contain a tile for each root topic of the published documentation. Each topic tile has three sections that corresponds to the topic title, short description, and image. To generate this component, the `webhelp_tiles` element must be specified in the template file. It can be used in the following type of template page:

- ***Main Page Template*** *(`wt_index.html` file)*

In the template file, the `webhelp_tiles` element can be specified as in the following example:

```
<whc:webhelp_tiles
```

```
      xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_tiles`.

If you want to control the HTML structure that is generated for a WebHelp tile you can also specify the template for a tile by using the `whc:webhelp_tile` component, as in the following example:

```
<whc:webhelp_tile class="col-md-4">
  <!-- Place holder for tile's image -->
  <whc:webhelp_tile_image/>

  <div class="wh_tile_text">
    <!-- Place holder for tile's title -->
     <whc:webhelp_tile_title/>

     <!-- Place holder for tile's shordesc -->
     <whc:webhelp_tile_shortdesc/>
  </div>
</whc:webhelp_tile>
```

For information about customizing the tiles, see *Configuring the Tiles on the Main Page* on page 970.

Main Page Table of Contents [webhelp_main_page_toc]

This component generates a simplified Table of Contents. It is simplified because it contains only two levels from the documentation hierarchy. To generate this component, the `webhelp_main_page_toc` element must be specified in the template file. It can be used in the following type of template page:

- ***Main Page Template*** *(`wt_index.html` file)*

In the template file, the `webhelp_main_page_toc` element can be specified as in the following example:

```
<whc:webhelp_main_page_toc
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_main_page_toc`.

Main Page Menu [webhelp_topics_menu]

This component generates a menu with all the documentation topics. To generate this component, the `webhelp_topics_menu` element must be specified in the template file. It can be used in the following type of template page:

- ***Main Page Template*** *(`wt_index.html` file)*

In the template file, the `webhelp_topics_menu` element can be specified as in the following example:

```
<whc:webhelp_topics_menu
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_topics_menu`.

You can control the maximum level of topics that will be included in the menu using the `webhelp.top.menu.depth` transformation parameter (in the **Parameters** tab of the transformation scenario.

For information about customizing the menu, see *Customizing the Menu* on page 972.

Print Link [webhelp_print_link]

This component is used to generate a print icon that opens the print dialog box for your particular browser. To generate this component, the `webhelp_print_link` element must be specified in the template file. It can be used in all four types of template pages:

- ***Main Page Template*** *(`wt_index.html` file)*
- ***Topic Template*** *(`wt_topic.html` file)*
- ***Search Results Template*** *(`wt_search.html` file)*
- ***Index Terms Template*** *(`wt_terms.html` file)*

In the template file, the `webhelp_print_link` element can be specified as in the following example:

```
<whc:webhelp_print_link
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_print_link`.

Include HTML files [webhelp_include_html]

This component can be used to include custom HTML files. To generate this component, the `include_html` element must be specified in the template file. It can be used in all four types of template pages:

- ***Main Page Template*** *(`wt_index.html` file)*
- ***Topic Template*** *(`wt_topic.html` file)*
- ***Search Results Template*** *(`wt_search.html` file)*
- ***Index Terms Template*** *(`wt_terms.html` file)*

In the template file, the `include_html` element can be specified as in the following example:

```
<whc:include_html href="${wh.param}"/>
```

Where the `href` can have the following values:

- **any URL** - In this case, the file to be included is specified as an URL.
- **{$oxygen-webhelp-template-dir}/file_to_include.html** - To include resources that are part of the template.
- **${webhelp.param}** - To include a resource whose path is specified through a WebHelp transformation scenario parameter. The value of this parameter can be a simple HTML fragment, in which case it will be copied to the output.

Index Terms Link [webhelp_indexterms_link]

This component can be used to generate a link to the index terms page (`indexterms.html`). If the published documentation does not contains any index terms, then the link will not be generated. To generate this component, the `webhelp_indexterms_link` element must be specified in the template file. It can be used in all four types of template pages:

- ***Main Page Template*** *(`wt_index.html` file)*
- ***Topic Template*** *(`wt_topic.html` file)*
- ***Search Results Template*** *(`wt_search.html` file)*
- ***Index Terms Template*** *(`wt_terms.html` file)*

In the template file, the `webhelp_indexterms_link` element can be specified as in the following example:

```
<whc:webhelp_indexterms_link
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_indexterms_link`. This element will contain a link to the `indexterms.html` page.

Link to Skin Resources [webhelp_skin_resources]

This component can be used to add a link to resources for the current WebHelp skin (such as the CSS file). To generate this component, the `webhelp_skin_resources` element must be specified in the template file. It can be used in all four types of template pages:

- ***Main Page Template*** *(`wt_index.html` file)*
- ***Topic Template*** *(`wt_topic.html` file)*
- ***Search Results Template*** *(`wt_search.html` file)*
- ***Index Terms Template*** *(`wt_terms.html` file)*

In the template file, the `webhelp_skin_resources` element can be specified as in the following example:

```
<whc:webhelp_skin_resources/>
```

In the output, you will find a link to the skin resources.

Customizing the WebHelp Responsive Output

To change the overall appearance of your WebHelp Responsive output, you can use several different customization methods or a combination of methods. If you are familiar with CSS and coding, you can style your WebHelp output through your own custom stylesheets. You can also customize your output by modifying existing templates, create your own, or by configuring certain options and parameters in the transformation scenario.

This section includes topics that explain various ways to customize your WebHelp Responsive system output, such as how to configure the tiles on the main page, add logos in the title area, integrate with social media, localizing the interface, and much more.

WebHelp Responsive Customization Methods

There are several methods that you can use to customize your WebHelp Responsive output. This topic provides information on each of the methods and highlights its advantages so that you can choose the best possible method based upon your needs.

Insert Custom XHTML Fragments in Predefined Placeholders

The WebHelp Responsive template contains a series of component placeholders. Some of these placeholders are left empty in the default output configurations, but you can use them to display custom content. This method is recommended if you want to use an existing skin and simply make some minor changes or additions in certain locations within the final output.

Advantages:

- This method is very easy, since the fragments for the placeholders can be specified in the transformation scenario.
- Advanced knowledge of CSS styling is not required for this method.

Each such placeholder has an associated parameter in the transformation scenario **Parameters** tab. These predefined empty placeholder parameters are illustrated and described below:

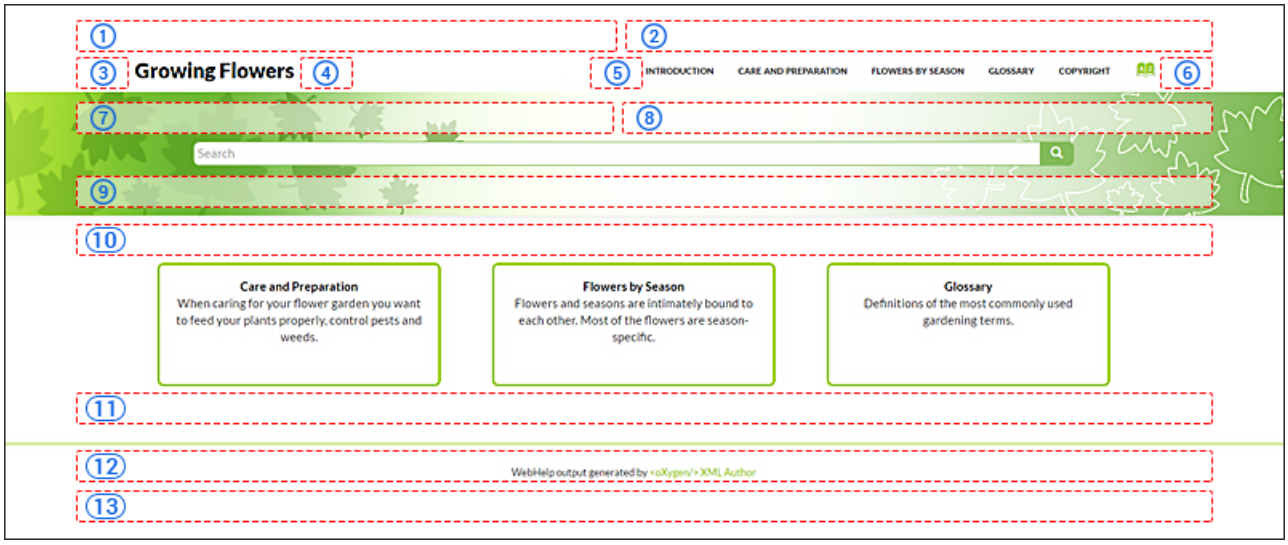


Figure 485: Predefined Placeholders Diagram

Each of these parameters can hold either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment:

1- webhelp.fragment.head

In the generated output it displays a given XHTML fragment as a page header. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

2- webhelp.fragment.before.body

In the generated output it displays a given XHTML fragment before the page body. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

3- webhelp.fragment.before.logo_and_title

In the generated output it displays a given XHTML fragment before the logo and title. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

4- webhelp.fragment.after.logo_and_title

In the generated output it displays a given XHTML fragment after the logo and title. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

5- webhelp.fragment.before.top_menu

In the generated output it displays a given XHTML fragment before the top menu. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

6- webhelp.fragment.after.top_menu

In the generated output it displays a given XHTML fragment after the top menu. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

7- webhelp.fragment.before.main.page.search

In the generated output it displays a given XHTML fragment before the search field. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

8- webhelp.fragment.welcome

In the generated output it displays a given XHTML fragment as a welcome message (or title). The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

9- webhelp.fragment.after.main.page.search

In the generated output it displays a given XHTML fragment after the search field. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

10- webhelp.fragment.before.toc_or_tiles

In the generated output it displays a given XHTML fragment before the table of contents or tiles in the main page. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

11- webhelp.fragment.after.toc_or_tiles

In the generated output it displays a given XHTML fragment after the table of contents or tiles in the main page. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

12- webhelp.fragment.footer

In the generated output it displays a given XHTML fragment as the page footer. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

13- webhelp.fragment.after.body

In the generated output it displays a given XHTML fragment after the page body. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

EXAMPLE:

To insert a message above the search field component in the main page of the output, follow this procedure:

1. Edit the **WebHelp Responsive** transformation scenario.
2. Go to **Parameters** tab and find the parameter associated with the place holder that you want to use. In this case, it is called `webhelp.fragment.welcome`.
3. Edit the parameter. Depending on the size of the content you want to add, you can insert one of the following:
   - A small well-formed XHTML fragment, such as: `<i>Welcome to our user guide</i>`.
   - A path to a file that contains well-formed XHTML content.

Customize WebHelp Output with a Custom CSS

This method is recommended if you just want to adapt an existing skin (that is very close to what you need) and only need to change the styling of the final output. For example, this might be the case if you simply want to change a color, or adjust some of the margins or paddings of certain components.

Advantages:

- This method could be used as a quick and easy way to make small styling changes.
- The custom CSS can be distributed with your project and shared with other members of your team.
- This method can be used for advanced and precise styling.

Additional Notes:

- The fonts, images, and other resources must be stored in a remote server location.
- This type of customization will not appear in the `Templates` tab of the transformation scenario. Instead, the custom CSS needs to be set as a parameter of an existing transformation scenario.

To set a custom CSS to a transformation scenario:

1. Edit the **WebHelp Responsive** transformation scenario.
2. Open the **Parameters** tab.
3. For a DITA transformation, set the `args.css` parameter to the path of your custom CSS file. Also, set the `args.copycss` parameter to `yes` to automatically copy your custom CSS in the output folder when the transformation scenario is processed. Also, if your customization CSS requires additional resources, you can copy them to the generated output by specifying the *`webhelp.custom.resources` parameter*.

If you are using DITA, you could also override the default XSLT templates that are used for WebHelp transformations by using some extension points. For information about this method, see *Overriding an XSLT Processing Step of the DITA WebHelp Transformation* on page 979.

Create or Customize a WebHelp Responsive Skin

This method is recommended if you want a design that is not similar to any of the predefined skins, or if you want to make a lot of changes to one of the existing skins. This method is also useful if you want to distribute additional resources (such as fonts and images) together with a custom CSS.

Advantages:

- The customized skin will be available in the **Templates** tab of the transformation scenario.
- The resources are encapsulated into the `skin` directory and can be shared with other team members, along with a custom CSS file.

Additional Notes:

- This method requires access to the installation folder, or the use of an external DITA-OT engine (with the *WebHelp plugin* installed).

To create a new WebHelp Responsive skin, follow this procedure:

1. Locate the following folder in your DITA-OT directory (*DITA-OT-DIR*):

   ```
   DITA-OT-DIR/plugins/com.oxygenxml.webhelp/templates/dita/bootstrap/variants/
   ```

2. Here you can see some subdirectories corresponding to different variants for the same template. For instance, the default directories are `tiles` and `tree`.
3. In each of these variants, you will find a directory for each of the skins (for example, the default skins and their corresponding directories are: *flowers*, *green*, *light*, *mechano*, *orange*, etc.)
4. Duplicate one of the skin folders and rename it to whatever you want your new skin to be identified as.
5. Edit the `skin.css` file and customize it the way you want. If your customization of the CSS file requires additional resources (such as images, fonts, or other CSS files), they need to be placed in the `resources` folder at the same level with the `skin.css` file.

   Result: Your new skin should now be included in the list of skins in the **Templates** tab of the transformation scenario.

Tip: During development, you may want to regularly test your customization. To shorten the publishing time of your tests, use a small set of test files (you could use one of the Oxygen XML Editor sample projects). Also, you can use your web browser CSS inspector tool to lookup the CSS classes you want to modify.

Create a New WebHelp Responsive Template

This method can be used when you need to make significant structural changes to the WebHelp output. For example, if you want to move some components to other positions, or if you want to use a different responsive front-end *framework* than the default *Bootstrap framework* (for instance, if you want to switch to *ZURB Foundation*).

Advantages:

- This method allows you to fully customize the output.
- This method allows you to *change the structure of the generated HTML files*.
- You can *create your own skins* for the new template.

Additional Notes:

- This method requires access to the installation folder, or the use of an external DITA-OT engine (with the *WebHelp plugin* installed).

To create a new WebHelp Responsive template, follow these steps:

1. Locate the following folder in your DITA-OT directory (*DITA-OT-DIR*):

   ```
   DITA-OT-DIR/plugins/com.oxygenxml.webhelp/templates/dita/
   ```

2. Duplicate the `bootstrap` folder and rename it to whatever you want your new template to be identified as (for example, `myTemplate`).
3. Customize the structure of the new template according to your needs. For example, if you only want to keep one of the template variants, open the `myTemplate/variants` folder and delete all of its subdirectories, except for that one (for instance, the `tiles` directory). Keep in mind that the structure of the template directory is important. The names of folders at certain levels correspond to the names of templates and skins, while components and resources are defined and referenced in certain files or folders at specific locations within the directory structure. For more information, see *WebHelp Responsive Template Directory Structure* on page 951.
4. You can also customize the structure of the skins within the template variants. For example, if you only want to keep one of the skins in the `tiles` variant, open the `myTemplate/variants/tiles` folder and delete all of its subdirectory skins, except for that one (for instance, the `light` directory).
5. Edit the `skin.css` file that is located in the skin directory (for example, `myTemplate/variants/tiles/light`) and customize it the way you want. If your customization of the CSS file requires additional resources (such as images, fonts, or other CSS files), they need to be placed in the `resources` folder at the same level with the `skin.css` file.

   Result: Your new templates and skins should now be included in the **Templates** tab of the transformation scenario.

Tip: During development you regularly need to test your customization. To shorten the publishing time of your test, use a small project (you could use one of the Oxygen XML Editor sample projects). Also, you can use your web browser CSS inspector tool to lookup the CSS classes you want to modify.

For more information about WebHelp Responsive templates, see the *WebHelp Responsive Template Mechanism* on page 950 section.

Copying Additional Resources to WebHelp Output Directory

To copy additional resources (such as JavaScript, CSS or other resources) to the output directory of a WebHelp system, follow these steps:

1. Place all your resources in the same directory.
2. Edit the WebHelp transformation scenario.
3. Go to the **Parameters** tab.

4. Edit the value for the webhelp.custom.resources parameter and set it to the absolute path of the directory in step 1.
5. Click **OK** to save the changes to the transformation scenario.
   All files from the new directory will be copied to the root of the WebHelp output directory.

Adding Custom HTML Content in WebHelp Responsive Output

You can add custom HTML content in the WebHelp Responsive output by inserting it in a well-formed XML file that will be referenced in the transformation scenario. This content may include references to additional JavaScript, CSS, and other types of resources, or such resources can be inserted inline within the HTML content that is inserted in the XML file.

To include custom HTML content in the WebHelp Responsive output files, follow these steps:

1. Insert the HTML content in a well-formed XML file. There are several things to consider in regards to this XML file:
   a. **Well-Formedness** - If the file is not a *Well-formed XML document* (or fragments are not well-formed), the transformation will fail.

      A common use case is if you want to include several `script` or `link` elements. In this case, the XML fragment has multiple root elements and to make it well-formed, you can wrap it in an `html` element. This element tag will be filtered out and only its children will be copied to the output documents. Similarly, you can wrap your content in `head`, `body`, `html/head`, or `html/body` elements.
   b. **Referencing Resources in the XML File** - You can include references to local resources (such as JavaScript or CSS files) by using the predefined `${oxygen-webhelp-output-dir}` macro to specify their paths relative to the output directory:

      ```
      <html>
        <script type="text/javascript" src="${oxygen-webhelp-output-dir}/js/test.js"/>
        <link rel="stylesheet" type="text/css"
              href="${oxygen-webhelp-output-dir}/css/test.css" />
      </html>
      ```

      If you want that the path of your resource to be relative to the *templates directory*, you can use the `${oxygen-webhelp-template-dir}` macro.

      To copy the referenced resources to the output directory, follow the procedure in: *Copying Additional Resources to WebHelp Output Directory* on page 966.
   c. **Inline JavaScript or CSS Content** - If you want to include inline JavaScript or CSS content in the XML file, it is important to place this content inside an XML comment, as in the following examples:

      JavaScript:

      ```
      <script type="text/javascript">
        <!--
          /* Include JavaScript code here. */

          function myFunction() {
            return true;
          }
        -->
      </script>
      ```

      CSS:

      ```
      <style>
        <!--
          /* Include CSS style rules here. */

          *{
            color:red
          }
        -->
      </style>
      ```
2. Edit the WebHelp Responsive transformation scenario.
3. Go to the **Parameters** tab.
4. Edit the value of the *webhelp.fragment.head parameter* and set it to the absolute path of the XML file created in step 1. Your additional content will be included at the end of the `head` element of your output document.

Note: If you want to insert the content in another location within the output document, you can reference the XML file in any of the other parameters listed in *Insert Custom XHTML Fragments in Predefined Placeholders* on page 963.

5. Click **OK** to save the changes to the transformation scenario.

Related Information:

Copying Additional Resources to WebHelp Output Directory on page 966

WebHelp Responsive Template Directory Structure on page 951

WebHelp Responsive Customization Methods on page 963

Adding a Favicon in WebHelp Systems

You can add a custom *favicon* to your WebHelp system by simply using a parameter in the transformation scenario to point to your *favicon* image. This is available for DITA and DocBook WebHelp systems using **WebHelp Responsive**, **WebHelp Responsive with Feedback**, **WebHelp Classic**, **WebHelp Classic with Feedback**, or **WebHelp Classic Mobile** transformation scenarios.

To add a *favicon*, follow these steps:

1. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.
2. Locate the `webhelp.favicon` parameter and enter the file path that points to the image that will be use as the *favicon*.
3. Run the transformation scenario.

Adding a Logo Image in the Title Area

To add a logo in the title area of your WebHelp output, follow this procedure:

1. Edit a WebHelp transformation scenario, then open the **Parameters** tab.
2. Specify the path to your logo in the `webhelp.logo.image` parameter.
3. If you also want to add a link to your website when you click the logo image, set the URL in the `webhelp.logo.image.target.url` parameter.
4. Run the transformation scenario.

Adding a Welcome Message in the Home Page

The main page of the WebHelp Responsive output contains a set of empty placeholders that can be used to display customized text fragments. These placeholders are available to you through *WebHelp Responsive transformation scenario parameters*. One of these placeholders (identified through the `webhelp.fragment.welcome` parameter) was designed to display text content above the search box in the main page.

To add a customized welcome message in the main page of the WebHelp Responsive output, follow this procedure:

1. Edit a WebHelp Responsive transformation scenario.
2. Open the ***Templates** tab* and choose a *skin*.
3. Open the ***Parameters** tab* and edit the `webhelp.fragment.welcome` parameter. The value of this parameter can be one of the following:
   - A small well-formed XHTML fragment (such as: `<i>Welcome to the User Guide</i>`).
   - Path to a file that contains well-formed XHTML content.
4. Click **OK**, then click the **Apply associated** button to execute the transformation scenario.

Adding Video and Audio Objects in DITA WebHelp Output

You can insert references to video and audio media resources (such as videos, audio clips, or embedded HTML frames) in your DITA topics and then publish them to WebHelp output. The media objects can be played directly in all HTML5-based outputs, including WebHelp systems.

To add media objects in the WebHelp output generated from DITA documents, follow the procedures below.

Adding Videos to DITA WebHelp Output

1. Edit the DITA topic and insert a reference to the video through one of the following methods:
   - Use the ***Insert Media Object*** *toolbar action*.
   - Drag (or copy) the video file from your system explorer or the ***Project*** *view* and drop (or paste) it into your document.
   - Manually add an `object` element, as in one of the following examples:

     ```
     <object outputclass="video" type="video/mp4" data="MyVideo.mp4"/>
     ```

     or, instead of the `data` attribute, you can specify the video using a parameter like this:

     ```
     <object outputclass="video">
       <param name="src" value="videos/MyVideo.mp4"/>
     </object>
     ```
2. Apply a **DITA to WebHelp** transformation scenario to obtain the output.

   Result: The transformation converts the `object` element to an HTML5 `video` element.

   ```
   <video controls="controls"><source type="video/mp4" src="MyVideo.mp4"></source>
   </video>
   ```

Adding Audio Clips to DITA WebHelp Output

1. Edit the DITA topic and insert a reference to the audio clip through one of the following methods:
   - Use the ***Insert Media Object*** *toolbar action*.
   - Drag (or copy) the audio file from your system explorer or the ***Project*** *view* and drop (or paste) it into your document.
   - Manually add an `object` element, as in one of the following examples:

     ```
     <object outputclass="audio" type="audio/mpeg" data="MyClip.mp3"/>
     ```

     or, instead of the `data` attribute, you can specify the video using a parameter like this:

     ```
     <object outputclass="audio">
       <param name="src" value="audio/MyClip.mp3"/>
     </object>
     ```
2. Apply a **DITA to WebHelp** transformation scenario to obtain the output.

   Result: The transformation converts the `object` element to an HTML5 `audio` element.

   ```
   <audio controls="controls"><source type="audio/mpeg" src="MyClip.mp3"></source>
   </audio>
   ```

Adding Embedded HTML Frames (such as YouTube videos) to DITA WebHelp Output

1. Edit the DITA topic and insert a reference to the embedded object by using the ***Insert Media Object*** *toolbar action* or by manually adding an `object` element, as in one of the following examples:

   ```
   <object outputclass="iframe" data="https://www.youtube.com/embed/m_vv2s5Trn4"/>
   ```

   or, instead of the `data` attribute, you can specify the object using a parameter like this:

   ```
   <object outputclass="iframe">
     <param name="src" value="http://www.youtube.com/embed/m_vv2s5Trn4"/>
   </object>
   ```
2. Apply a **DITA to WebHelp** transformation scenario to obtain the output.

   Result: The transformation converts the `object` element to an HTML5 `iframe` element.

   ```
   <iframe controls="controls" src="https://www.youtube.com/embed/m_vv2s5Trn4">
   </iframe>
   ```

For more information, see the following video demonstration: *https://www.oxygenxml.com/demo/Media_Objects.html*.

Related Information:

Adding Video, Audio, and Embedded HTML Resources in DITA Topics on page 1626

Configuring the Tiles on the Main Page

The *tiles* version of the main page of the WebHelp Responsive output displays a tile for each topic found on the first level of the *DITA map*. However, you might want to customize the way they look or even to hide some of them.

Depending on your particular setup, you can choose to customize the these tiles either by setting metadata information in the *DITA map* or by customizing the CSS that is associated with the *DITA map*.

How to Hide Some of the Tiles

If your documentation is very large or there is a large number of topics on the first level, you might want to hide some of the tiles. Also, this might be useful if you only want to display the topics in the first page that are most relevant to your intended audience.

There are two methods for doing this. One of them involves editing the *DITA map* and marking the topics that do not need to be displayed as tiles, and another one that uses a small CSS customization level to hide some tiles identified by the ID of the topic.

Editing the DITA Map

To edit the metadata in the *DITA map* to control which topics on the first level of the *DITA map* will not be displayed as a tile, follow these steps:

1. Open the *DITA map* in the **Text** editing mode of Oxygen XML Editor.
2. Add the following metadata information in the `topicref` element (or any of its specializations) for each first-level topic that you do not want to be displayed as a tile:

```
<topicmeta>
  <data name="wh-tile">
    <data name="hide" value="yes"/>
  </data>
</topicmeta>
```

Customizing the CSS

To customize the CSS to control which topics on the first level of the *DITA map* will not be displayed as a tile, follow these steps:

1. Make sure you set an ID on the topic you want to hide.
2. Create a new CSS file that contains a rule that hides the tile generated for the topic (identified in the following example by the topic ID `growing-flowers`). The CSS file should have content that is similar to this:

```
.wh_tile [data-id='growing-flowers'] {
  display:none;
}
```

3. Use the *Customizing WebHelp Output with a Custom CSS* method to pass the CSS file you just created to the transformation scenario.

How to Add an Image to the Tiles

There are two methods that you can use to add an image to a tile. One of them involves editing the *DITA map*, and the other uses a CSS customization.

Editing the DITA Map

To edit the metadata in the *DITA map* to set an image to be displayed in a tile, follow these steps:

1. Open the *DITA map* in the **Text** editing mode of Oxygen XML Editor.

2. Add the following metadata information in the `topicref` element (or any of its specializations) for each first-level topic that will have an image displayed in the corresponding tile:

```
<topicmeta>
  <data name="wh-tile">
    <data name="image" href="img/tile-image.png" format="png">
     <data name="attr-width" value="64"/>
     <data name="attr-height" value="64"/>
    </data>
  </data>
</topicmeta>
```

Note: The `attr-width` and `attr-height` attributes can be used to control the size of the image, but they are optional.

Customizing the CSS

To customize the CSS to set an image to be displayed in a tile, follow these steps:

1. Make sure you set an ID on the topic that you want the tile include an image.
2. Create a new CSS file that contains a rule that associates an image with a specific tile. The CSS file should have content that is similar to this:

```
.wh_tile[data-id='growing-flowers']> div {
    background-image:url('resources/flower.png');
}
```

3. Use the *Customizing WebHelp Output with a Custom CSS* or the *Create a WebHelp Responsive Skin* method to pass the CSS file you just created to the transformation scenario.

Change Numbering Styles for Ordered Lists

Ordered lists (`ol`) are usually numbered in XHTML output using numerals. If you want to change the numbering to alphabetical, follow these steps:

1. Define a custom `outputclass` value and set it as an attribute of the ordered list, as in the following example:

```
<ol outputclass="number-alpha">
    <li>A</li>
    <li>B</li>
    <li>C</li>
</ol>
```

2. Add the following code snippet in a custom CSS file:

```
ol.number-alpha{
    list-style-type:lower-alpha;
}
```

3. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.
   a. For a DITA transformation, set the `args.css` parameter to the path of your custom CSS file. Also, set the `args.copycss` parameter to `yes` to automatically copy your custom CSS in the output folder when the transformation scenario is processed.
   b. For a DocBook transformation, set the `html.stylesheet` parameter to the path of your custom CSS file.
4. Run the transformation scenario.

CSS Styling to Customize WebHelp Output

One way to customize WebHelp output is to use custom CSS styling. This method can be used to make small, simple styling changes or more advanced, precise changes. To implement the styling in your WebHelp output, you simply need to create the custom CSS file and reference it in your transformation scenario.

As a practical example, to hide the horizontal separator line between the content and footer, follow these steps:

1. Create a custom CSS file that contains the following snippet:

```
.footer_separator {
  display:none;
}
```

2. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.

a. For a DITA transformation, set the `args.css` parameter to the path of your custom CSS file. Also, set the `args.copycss` parameter to `yes` to automatically copy your custom CSS in the output folder when the transformation scenario is processed.
b. For a DocBook transformation, set the `html.stylesheet` parameter to the path of your custom CSS file.

3. Run the transformation scenario.

Customizing the Menu

By default, the menu component is displayed in all WebHelp Responsive pages. However, you might want to hide it completely, or only display some of its menu entries.

How to Hide Some of the Menu Entries

There are two methods for doing this. One of them involves editing the *DITA map* and marking the topics that do not need to be included in the menu, and another one that uses a small CSS customization.

Editing the DITA Map

To edit the metadata in the *DITA map* to control which topics will not be displayed in the menu, follow these steps:

1. Open the *DITA map* in the **Text** editing mode of Oxygen XML Editor.
2. Add the following metadata information in the `topicref` element (or any of its specializations) for each topic you do not want to be displayed in the menu:

```
<topicmeta>
  <data name="wh-menu">
    <data name="hide" value="yes"/>
  </data>
</topicmeta>
```

Customizing the CSS

To customize the CSS to control which topics will not be displayed in the menu, follow these steps:

1. Make sure you set an ID on the topic that you do not want to include in the menu.
2. Create a new CSS file that contains a rule that hides the menu entry generated for the topic (identified by the topic ID `growing-flowers` in the following example). The CSS file should have content that is similar to this:

```
.wh_top_menu *[data-id='growing-flowers'] {
  display:none;
}
```

3. Use the *Customizing WebHelp Output with a Custom CSS* method to pass the CSS file you just created to the transformation scenario.

How to Hide the Entire Menu

If you do not want to include a main menu in the pages of the WebHelp Responsive output, you can instruct the transformation scenario to skip the menu generation completely. To do so, follow this procedure:

1. Edit a WebHelp Responsive transformation scenario.
2. Open the ***Templates** tab* and choose a *skin*.
3. Open the ***Parameters** tab* and set the value of the `webhelp.show.top.menu` parameter to no.
4. Click **OK**, then click the **Apply associated** button to execute the transformation scenario.

Editing Scoring Values of Tag Elements in Search Results

The WebHelp **Search** feature is enhanced with a rating mechanism that computes scores for every page that matches the search criteria. HTML tag elements are assigned a scoring value and these values are evaluated for the search results. Oxygen XML Editor includes a properties file that defines the scoring values for tag elements and this file can be edited to customize the values according to your needs.

To edit the scoring values of HTML tag element for enhancing WebHelp search results, follow these steps:

1. Edit the scoring properties file for DITA or DocBook WebHelp systems. The properties file includes instructions and examples to help you with your customization.

a) For DITA WebHelp systems, edit the following file: *DITA-OT-DIR*`\plugins\com.oxygenxml.webhelp\indexer\scoring.properties`.
b) For DocBook WebHelp system, edit the following file: *[OXYGEN_INSTALL_DIR]*`\frameworks\docbook\xsl\com.oxygenxml.webhelp\indexer\scoring.properties`.

The values that can be edited in the `scoring.properties` file:

```
h1 = 10
h2 = 9
h3 = 8
h4 = 7
h5 = 6
h6 = 5
b = 5
strong = 5
em = 3
i=3
u=3
div.toc=-10
title=20
div.ignore=ignored
meta_keywords = 20
meta_indexterms = 20
meta_description = 25
shortdesc=25
```

2. Save your changes to the file.
3. Re-run your WebHelp system transformation scenario.

Exclude Certain DITA Topics from WebHelp Search Results

The WebHelp **Search** engine does not index DITA topics that have the `@search` attribute set to no. This is useful if you have topics in your *DITA map* structure that you do not want to be included in search results for your WebHelp system.

To exclude DITA topics from WebHelp search results, follow these steps:

1. Edit the *DITA map* and for any `topicref` that you want to exclude from search results, set the `search` attribute to no.

   For example:

```
<topicref href="../topics/internal-topic1.dita" search="no"/>
```

2. Save your changes to the *DITA map*.
3. Re-run your WebHelp system transformation scenario.

Flag DITA Content in WebHelp Output

Flagging content in WebHelp output involves defining a set of images that will be used for marking content across your information set.

To flag DITA content, you need to create a filter file that defines properties that will be applied on elements to be flagged. Generally, flagging is supported for *block elements* (such as paragraphs), but not for phrase-level elements within a paragraph. This ensures that the images that will flag the content are easily scanned by the reader, instead of being buried in text.

Follow this procedure:

1. Create a DITA filter file in the directory where you want to add the file. Give the file a descriptive name, such as `audience-flag-build.ditaval`.
2. Define the property of the element you want to be flagged. For example, if you want to flag elements that have the `audience` attribute set to `programmer`, the content of the DITAVAL file should look like the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<val>
  <prop att="audience" val="programmer" action="flag"
 img="D:\resource\delta.gif" alt="sample alt text"/>
</val>
```

   Note that for an element to be flagged, at least one attribute-value pair needs to have a property declared in the DITAVAL file.

3. Specify the DITAVAL file in the **Filters** tab of the transformation scenario.
4. Run the transformation scenario.

Integrating Social Media and Google Tools in WebHelp Output

Oxygen XML Editor includes support for integrating some of the most popular social media sites in WebHelp output.

How to Add a Facebook Like Button in WebHelp Responsive Output

To add a Facebook™ *Like* widget to your WebHelp output, follow these steps:

1. Go to the *Facebook Developers* website.
2. Fill-in the displayed form, then click the **Get Code** button.
   A dialog box that contains code snippets is displayed.
3. Copy the two code snippets and paste them into a `<div>` element inside an XML file called `facebook-widget.xml`.

   Make sure you follow these rules:

   - The file must be well-formed.
   - The code for each `script` element must be included in an XML comment.
   - The start and end tags for the XML comment must be on a separate line.

   The content of the XML file should look like this:

```
<div id="facebook">
    <div id="fb-root"/>
    <script>
        <!--
            (function(d, s, id) {
        var js, fjs = d.getElementsByTagName(s)[0];
        if (d.getElementById(id)) return;
        js = d.createElement(s); js.id = id;
        js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
        fjs.parentNode.insertBefore(js, fjs);
        }(document, 'script', 'facebook-jssdk'));
        -->
    </script>
    <div class="fb-like" data-layout="standard" data-action="like"
        data-show-faces="true" data-share="true"/>
</div>
```

4. In Oxygen XML Editor, click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
5. Select an existing WebHelp Responsive transformation scenario (depending on your needs, it may be with or without feedback) and click the **Duplicate** button to open the **Edit Scenario** dialog box.
6. Switch to the **Parameters** tab. Depending on where you want to display the button, edit *one of the parameters that begin with `webhelp.fragment`*. Set that parameter to reference the `facebook-widget.xml` file that you created earlier.
7. Click **Ok**.
8. Run the transformation scenario.

How to Add a Google+ Button in WebHelp Responsive Output

To add a *Google+* widget to your WebHelp Responsive output, follow these steps:

1. Go to the *Google Developers* website.
2. Fill-in the displayed form.
   The preview area on the right side displays the code and a preview of the widget.
3. Copy the code snippet displayed in the preview area and paste it into a `div` element inside an XML file called `google-plus-button.xml`.

   Make sure that the content of the file is well-formed.

   The content of the XML file should look like this:

```
<div id="google-plus">
  <!-- Place this tag in your head or just before your close body tag. -->
  <script src="https://apis.google.com/js/platform.js" async defer></script>
```

```
  <!-- Place this tag where you want the +1 button to render. -->
  <div class="g-plusone" data-annotation="inline" data-width="300"></div>
</div>
```

4. In Oxygen XML Editor, click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
5. Select an existing WebHelp Responsive transformation scenario (depending on your needs, it may be with or without feedback) and click the **Duplicate** button to open the **Edit Scenario** dialog box.
6. Switch to the **Parameters** tab. Depending on where you want to display the button, edit *one of the parameters that begin with `webhelp.fragment`*. Set that parameter to reference the `google-plus-button.xml` file that you created earlier.
7. Click **Ok**.
8. Run the transformation scenario.

Related Information:

DITA Map to WebHelp Output on page 808

How to Add Tweet Button in WebHelp Responsive Output

To add a Twitter™ *Tweet* widget to your WebHelp Responsive output, follow these steps:

1. Go to the *Tweet button generator* page.
2. Fill-in the displayed form.
   The **Preview and code** area displays the code.
3. Copy the code snippet displayed in the **Preview and code** area and paste it into a `div` element inside an XML file called `tweet-button.xml`.

   Make sure you follow these rules:

   - The file must be well-formed.
   - The code for each `script` element must be included in an XML comment.
   - The start and end tags for the XML comment must be on a separate line.

   The content of the XML file should look like this:

```
<div id="twitter">
  <a href="https://twitter.com/share" class="twitter-share-button">Tweet</a>
  <script>
    <!--
      !function (d, s, id) {
      var
      js, fjs = d.getElementsByTagName(s)[0], p = /^http:/.test(d.location)
 ? 'http': 'https';
      if (! d.getElementById(id)) {
      js = d.createElement(s);
      js.id = id;
      js.src = p + '://platform.twitter.com/widgets.js';
      fjs.parentNode.insertBefore(js, fjs);
      }
      }
      (document,
      'script', 'twitter-wjs');
     -->
  </script>
</div>
```

4. In Oxygen XML Editor, click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
5. Select an existing WebHelp Responsive transformation scenario (depending on your needs, it may be with or without feedback) and click the **Duplicate** button to open the **Edit Scenario** dialog box.
6. Switch to the **Parameters** tab. Depending on where you want to display the button, edit *one of the parameters that begin with `webhelp.fragment`*. Set that parameter to reference the `tweet-button.xml` file that you created earlier.
7. Click **Ok**.
8. Run the transformation scenario.

Related Information:

DITA Map to WebHelp Output on page 808

How to Integrate Google Analytics in WebHelp Responsive Output

To allow your WebHelp Responsive system to benefit from Google Analytics reports, follow these steps:

1. Create a new Google Analytics account (if you do not already have one) and log on.
2. Choose the Analytics solution that fits the needs of your website.
3. Follow the on-screen instructions to obtain a **Tracking Code** that contains your *Tracking ID*.

   A **Tracking Code** looks like this:

```
<script>
 (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
 (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
 m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
 })(window,document,'script','//www.google-analytics.com/analytics.js','ga');

 ga('create', 'UA-XXXXXXXX-X', 'auto');
 ga('send', 'pageview');
</script>
```

4. Save the Tracking Code (obtained in the previous step) in a new HTML file called `googleAnalytics.html`.
5. In Oxygen XML Editor, click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
6. Select an existing WebHelp Responsive transformation scenario (depending on your needs, it may be with or without feedback) and click the **Duplicate** button to open the **Edit Scenario** dialog box.
7. Switch to the **Parameters** tab. Depending on where you want to display the button, edit *one of the parameters that begin with `webhelp.fragment`*. Set that parameter to reference the `googleAnalytics.html` file that you created earlier.
8. Click **Ok**.
9. Run the transformation scenario.

Related Information:

DITA Map to WebHelp Output on page 808

How to Integrate Google Search in WebHelp Responsive Output

You can integrate Google Search into your WebHelp Responsive output.

To allow your WebHelp system to use Google Search, follow these steps:

1. Go to the Google Custom Search Engine page using your Google account.
2. Press the **Create a custom search engine** button.
3. Follow the on-screen instructions to create a search engine for your site. At the end of this process you should obtain a code snippet.

   A Google Search script looks like this:

```
<script>
 (function()  {
  var cx =
    '000888210889775888983:8mn4x_mf-yg';
  var gcse = document.createElement('script');
  gcse.type = 'text/javascript';
  gcse.async = true;
  gcse.src = (document.location.protocol == 'https:' ?
    'https:' : 'http:') +         '//www.google.com/cse/cse.js?cx=' + cx;
  var s = document.getElementsByTagName('script')[0];
  s.parentNode.insertBefore(gcse, s);
 }
 )();
</script>
```

4. Save the script into a well-formed HTML file called `googlecse.html`.
5. In Oxygen XML Editor, click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
6. Select an existing WebHelp Responsive transformation scenario (depending on your needs, it may be with or without feedback) and click the **Duplicate** button to open the **Edit Scenario** dialog box.
7. Switch to the **Parameters** tab and edit the `webhelp.google.search.script` parameter to reference the `googlecse.html` file that you created earlier.

8. You can also use the `webhelp.google.search.results` parameter to choose where to display the search results.
   a) Create an HTML file with the following content: `<div class="gcse-searchresults-only" data-autoSearchOnLoad="true" data-queryParameterName-"searchQuery"></div>` (you must use the *HTML5 version for the GCSE*). For more information about other supported attributes, see *Google Custom Search: Supported Attributes*.
   b) Set the value of the `webhelp.google.search.results` parameter to the file path of the file you just created. If this parameter is not specified, the following code is used: `<div class="gcse-searchresults-only" data-autoSearchOnLoad="true" data-queryParameterName="searchQuery"></div>`.
9. Click **Ok**.
10. Run the transformation scenario.

Related Information:

Integrating Social Media and Google Tools in WebHelp Output on page 974

Localizing the Email Notifications of WebHelp with Feedback Systems

The feedback-enabled WebHelp systems use emails to notify users when comments are posted. These emails are based on templates stored in the WebHelp directory. The default messages are in English, French, German, and Japanese. These messages are copied into the WebHelp system deployment directory during the execution of the corresponding transformation scenario.

Suppose that you want to localize the emails into Dutch (*nl*). Follow these steps:

DocBook WebHelp Classic with Feedback

1. Create the following directory:

   [OXYGEN_INSTALL_DIR]`\frameworks\docbook\xsl\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\nl`
2. Copy all English template files from *[OXYGEN_INSTALL_DIR]*`\frameworks\docbook\xsl\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\en` and paste them into the directory you just created.
3. Edit the HTML files from the *[OXYGEN_INSTALL_DIR]*`\frameworks\docbook\xsl\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\nl` directory and translate the content into Dutch.
4. Start Oxygen XML Editor and edit the **DocBook WebHelp Classic with Feedback** transformation scenario.
5. In the **Parameters** tab, look for the `l10n.gentext.default.language` parameter and set its value to the appropriate language code. In our example, use the value `nl` for Dutch.

   Note: If you set the parameter to a value such as `LanguageCode-CountryCode` (for example, `en-us`), the transformation scenario will only use the language code
6. Run the transformation scenario to obtain the **WebHelp with Feedback** output.

DITA WebHelp Classic with Feedback or WebHelp Responsive with Feedback

1. Create the following directory:

   DITA-OT-DIR`\plugins\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\nl`
2. Copy all English template files from *DITA-OT-DIR*`\plugins\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\en` and paste them into the directory you just created.
3. Edit the HTML files from the *DITA-OT-DIR*`\plugins\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\nl` directory and translate the content into Dutch.
4. Start Oxygen XML Editor and edit the **DITA Map WebHelp Classic with Feedback** or **DITA Map WebHelp Responsive with Feedback** transformation scenario.
5. In the **Parameters** tab, look for the `args.default.language` parameter and set its value to the appropriate language code. In our example, use the value `nl` for Dutch.

Note: If you set the parameter to a value such as `LanguageCode-CountryCode` (for example, `en-us`), the transformation scenario will only use the language code

6. Run the transformation scenario to obtain the **WebHelp with Feedback** output.

Localizing the Interface of WebHelp Output (for DITA Map Transformations)

Static labels that are used in the WebHelp output are kept in translation files in the *`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp/oxygen-webhelp/resources/localization` folder. Translation files have the *strings-lang1-lang2.xml* name format, where *lang1* and *lang2* are ISO language codes. For example, the US English text is kept in the *strings-en-us.xml* file.

To localize the interface of the WebHelp output for *DITA map* transformations, follow these steps:

1. Look for the *strings-[lang1]-[lang2].xml* file in *`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp/oxygen-webhelp/resources/localization` directory (for example, the Canadian French file would be: *strings-fr-ca.xml*). If it does not exist, create one starting from the *strings-en-us.xml* file.
2. Translate all the labels from the above language file. Labels are stored in XML elements that have the following format: `<str name="Label name">Caption</str>`.
3. Run the predefined transformation scenario called **Run DITA OT Integrator** by executing it from the **Apply Transformation Scenario(s)** dialog box. If the *integrator* is not visible, select the **Show all scenarios** action that is available in the **Settings** drop-down menu.
4. Make sure that the new XML file that you created in the previous two steps is listed in the file *`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp/oxygen-webhelp/resources/localization/strings.xml`. In our example for the Canadian French file, it should be listed as: `<lang xml:lang="fr-ca" filename="strings-fr-ca.xml"/>`.
5. Edit any of the **DITA Map to WebHelp** transformation scenarios (with or without feedback, or the mobile version) and set the args.default.language parameter to the code of the language you want to localize (for example, *fr-ca* for Canadian French).
6. Run the transformation scenario to produce the WebHelp output.

Related Information:

Searching Japanese Content in WebHelp Pages on page 978

Searching Japanese Content in WebHelp Pages

To optimize the indexing of Japanese content in WebHelp pages, the Lucene Kuromoji Japanese analyzer can be used. This analyzer is included in the Oxygen XML Editor installation kit.

Activating Japanese Indexing in DITA WebHelp Systems

To activate the Japanese indexing in your WebHelp system generated from DITA content, follow these steps:

1. Set the language for your content to Japanese with one of the following two methods:
   - Edit a **DITA to WebHelp** transformation scenario and in the ***Parameters*** *tab*, set the value of the `args.default.language` parameter to `ja-jp`.
   - Set the `xml:lang` attribute on the root of the *DITA map* and the referenced topics to `ja-jp`.
2. For the analyzer to work properly, search terms that are entered into the WebHelp search text field must be separated by spaces.
3. Run the **DITA to WebHelp** transformation scenario to generate the output.

Optionally a Japanese user dictionary can be set with the *`webhelp.search.japanese.dictionary` parameter*.

Activating Japanese Indexing in DocBook WebHelp Systems

To activate the Japanese indexing in your WebHelp system generated from DocBook content, follow these steps:

1. Edit a **DocBook to WebHelp** transformation scenario and in the ***Parameters*** *tab*, set the value of the `l10n.gentext.default.language` parameter to `ja`.
2. Run the transformation scenario to generate the output.

Related Information:

Localizing the Interface of WebHelp Output (for DITA Map Transformations) on page 978
Localizing the Interface of WebHelp Output (for DocBook Transformations) on page 1011

Overriding an XSLT Processing Step of the DITA WebHelp Transformation

Since WebHelp output is primarily obtained by running XSLT transformations over the DITA input files (through the **DITA Map WebHelp** transformation scenarios), one customization method would be to override the default XSLT templates that are used by the WebHelp transformations.

There are two methods available to override the XSLT stylesheets implied by the WebHelp transformation.

- The first method is to use one of the ***WebHelp XSLT-import extension points*** that allow you to import an XSLT stylesheet so that it becomes part of the normal build. This method uses the same mechanism as the *DITA-OT XSLT-import extension points*. Note that this method will affect all the outputs generated with the WebHelp system.
- The second method implies that you will *create a DITA-OT extension plugin* with a custom *transtype* and use an Ant build file to define the transformation process. The main difference from the first customization method is that you will not affect all WebHelp transformations. Only the transformations that use the declared plugin *transtype* will be affected.

WebHelp XSLT-Import and XSLT-Parameter Extension Points

The WebHelp XSLT-Import extension points allows you to extend the XSLT stylesheets associated with some of the WebHelp transformation steps. The DITA-OT plug-in installer adds an XSLT import statement in the default WebHelp XSLT so that the XSLT stylesheet referenced by the extension point becomes part of the normal build.

Example:

```
<plugin id="com.oxygenxml.webhelp.extension">
  <feature extension="com.oxygenxml.webhelp.xsl.dita2webhelp" file="xsl/fixup.xsl"/>
</plugin>
```

Attention: The customizations you make by using this extension point will affect all WebHelp transformations. If you want to have a customization that is only available for a certain transformation, please use the *Overriding a WebHelp XSLT Stylesheet from an Ant Build File* method.

XSLT-Import Extension Points

The following extension points are available:

com.oxygenxml.webhelp.xsl.dita2webhelp

Extension point to override the XSLT stylesheet (`dita2webhelpImpl.xsl`) that produces an HTML file for each DITA topic. Depending on the type of WebHelp transformation you are currently using, the path to this stylesheet is as follows:

- ***WebHelp Classic* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\desktop\dita2webhelpImpl.xsl`
- ***WebHelp Classic Mobile* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\mobile\dita2webhelpImpl.xsl`
- ***WebHelp Responsive* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\responsive\dita2webhelpImpl.xsl`

com.oxygenxml.webhelp.xsl.createMainFiles

Extension point to override the XSLT stylesheet (`createMainFilesImpl.xsl`) that produces the WebHelp main HTML page (`index.html`). Depending on the type of WebHelp transformation you are currently using, the path to this stylesheet is as follows:

- ***WebHelp Classic* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\desktop\createMainFilesImpl.xsl`
- ***WebHelp Classic Mobile* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\mobile\createMainFilesImpl.xsl`

- ***WebHelp Responsive* Transformations** - *DITA-OT-DIR*`\plugins\com.oxygenxml.webhelp\xsl\dita\responsive\createMainFilesImpl.xsl`

com.oxygenxml.webhelp.xsl.createTocXML

Extension point to override the XSLT stylesheet (`tocDita.xsl`) that produces the `toc.xml` file. This file contains information extracted from the *DITA map* and it is mainly used to construct the WebHelp Table of Contents and navigational links. The path to this stylesheet is: *DITA-OT-DIR*`\plugins\com.oxygenxml.webhelp\xsl\dita\tocDita.xsl`.

XSLT-Parameter Extension Points

If your customization stylesheet declares one or more XSLT parameters and you want to control their values from the transformation scenario, you can use one of the following XSLT parameter extension points:

com.oxygenxml.webhelp.xsl.dita2webhelp.param

Use this extension point to pass parameters to the stylesheet specified using the ***com.oxygenxml.webhelp.xsl.dita2webhelp*** *extension point*.

com.oxygenxml.webhelp.xsl.createMainFiles.param

Use this extension point to pass parameters to the stylesheet specified using the ***com.oxygenxml.webhelp.xsl.createMainFiles*** *extension point*.

com.oxygenxml.webhelp.xsl.createTocXml.param

Use this extension point to pass parameters to the stylesheet specified using the ***com.oxygenxml.webhelp.xsl.createTocXml*** *extension point*.

Related Information:

[DITA-OT] XSLT-Import Extension Points

[DITA-OT] XSLT-Parameter Extension Points

Example: Customizing Side TOC Using XSLT-Import/Parameter Extension Points

This topic provides some common use-cases to demonstrate how to use the WebHelp XSLT-Import extension points to customize the Table of Contents displayed on the right side of the WebHelp output (the default transformation generates a mini table of contents for the current topic and it contains links to the children of current topic, its siblings, and all of its ancestors). The first use-case uses an XSLT-Import extension point while the second uses an XSLT-Parameter extention point.

Use Case 1: WebHelp XSLT-Import extension point to change which topics are displayed in the Side TOC

Suppose you want to customize the side TOC to only include the current topic and its child topics (while excluding its siblings and ancestors).

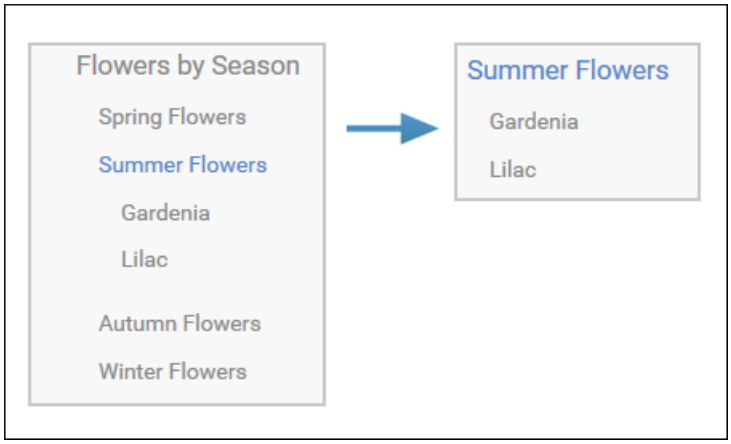


Figure 486: Example: Filtered Side Table of Contents

The default XSLT template responsible for this functionality is defined in: *DITA-OT-DIR*`\plugins\com.oxygenxml.webhelp\xsl\dita\responsive\navigationLinks.xsl`.

```
<xsl:template
          match="
          toc:topic
          [not(@toc = 'no')]
          [not(@processing-role = 'resource-only')]"
          mode="toc-pull" priority="10">
```

This template computes the link for the current topic along with the links for the sibling topics, and then propagates them recursively to the parent topic. For this specific use-case, you need to override this template so that it will produce the link for the current topic along with just the child links that the template already receives.

You can implement this functionality with a WebHelp extension plugin that uses the ***com.oxygenxml.webhelp.xsl.dita2webhelp*** *extension point*. This extension point allows you to specify a customization stylesheet that will override the template described above.

To add this functionality as a DITA-OT plugin, follow these steps:

1. In the *DITA-OT-DIR*`\plugins\` folder, create a folder for this plugin (for example, `com.oxygenxml.webhelp.custom.sidetoc`).
2. Create a **plugin.xml** file (in the folder you created in step 1) that specifies the extension point and your customization stylesheet. For example:

```
<plugin id="com.oxygenxml.webhelp.custom.sidetoc">
  <feature extension="com.oxygenxml.webhelp.xsl.dita2webhelp"
              file="custom_side_TOC.xsl"/>
</plugin>
```

3. Create your customization stylesheet (for example, **custom_side_TOC.xsl**), and edit it to override the template that produces the side TOC:

```
<xsl:template
              match="
              toc:topic
              [not(@toc = 'no')]
              [not(@processing-role = 'resource-only')]"
              mode="toc-pull"
              priority="10">

              <xsl:param name="pathFromMaplist" as="xs:string"/>
              <xsl:param name="children" select="()" as="element()*"/>
              <xsl:param name="parent" select="parent::*" as="element()?"/>

              <xsl:apply-templates select="." mode="sideTocEntry">
              <xsl:with-param name="pathFromMaplist" select="$pathFromMaplist"/>
              <xsl:with-param name="children" select="$children"/>
              </xsl:apply-templates>
</xsl:template>
```

4. Use the **Run DITA OT Integrator** transformation scenario found in the **DITA Map** section in the ***Configure Transformation Scenario(s)*** *dialog box*.
5. Run a **DITA Map WebHelp Responsive** transformation scenario to obtain the customized side TOC.

Use-Case 2: WebHelp XSLT-Parameter extension point to control which topics are displayed in the Side TOC from the transformation scenario

Another possibility to customize what is displayed in the side Table of Contents is to add a transformation parameter that will control the XSLT customization when the transformation scenario is applied. For this use-case, you can use the ***com.oxygenxml.webhelp.xsl.dita2webhelp.param*** *extension point*.

To add this functionality, follow these steps:

1. Create a DITA-OT plugin structure by following the first 3 steps in the *procedure above*.
2. In the customization stylesheet that you created in step 3, declare the `side_toc_only_children` parameter and modify the template to match the topic only when the `side_toc_only_children` parameter is set to *yes*:

```
<xsl:param name="side_toc_only_children" select="'yes'"/>
              ...
              <xsl:template
              match="
              toc:topic
              [not(@toc = 'no')]
```

```
              [not(@processing-role = 'resource-only')]
              [$side_toc_only_children = 'yes']"
              ...
```

3. Edit the **plugin.xml** file to specify the **com.oxygenxml.webhelp.xsl.dita2webhelp.param** extension point and a custom parameter file by adding the following line:

```
<feature extension="com.oxygenxml.webhelp.xsl.dita2webhelp.param"
              file="custom_params.xml"/>
```

4. Create a custom parameter file (for example, **custom_params.xml**). It should look like this:

```
<dummy>
              <param name="side_toc_only_children" expression="${side_toc_only_children}"
              if="side_toc_only_children"/>
</dummy>
```

5. Use the **Run DITA OT Integrator** transformation scenario found in the **DITA Map** section in the ***Configure Transformation Scenario(s)*** *dialog box*.
6. Edit a **DITA Map WebHelp Responsive** transformation scenario and in the ***Parameters*** *tab*, specify the desired value (*yes* or *no*) for your custom parameter (`side_toc_only_children`).
7. Run the transformation scenario.

Related Information:

[DITA-OT] XSLT-Import Extension Points

[DITA-OT] XSLT-Parameter Extension Points

Overriding a WebHelp XSLT Stylesheet from an Ant Build File

To create a WebHelp XSLT customization that is only available for a certain DITA OT transformation, the extension plugin should *declare a custom transtype*. The WebHelp XSLT stylesheets can be overridden from an ANT file provided by the DITA-OT extension plugin. From the Ant target associated with the plugin, you will specify a custom XSLT stylesheet that imports the original WebHelp stylesheet and add some customization templates.

The following procedure explains how to create a DITA-OT extension plugin that uses this extension method:

1. In the *`DITA-OT-DIR`*`\plugins\` folder, create a folder for this plugin (for example, `com.oxygenxml.webhelp.responsive.custom`).
2. Create a **plugin.xml** file (in the folder you created in step 1) that specifies a new DITA-OT `transtype` and the build file associated with the plugin. For example:

```
<plugin id="com.oxygenxml.webhelp.responsive.customization">
    <feature extension="dita.conductor.target.relative" file="integrator.xml"/>
    <transtype name="webhelp-responsive-custom" extends="webhelp-responsive"
      desc="WebHelp Responsive Customization"/>
</plugin>
```

3. Create the **integrator.xml** file that will import the actual plugin Ant build file.

```
<project basedir="." name="Webhelp Responsive Customization">
    <import file="build.xml"/>
</project>
```

4. Create the **build.xml** file that overrides the value of properties associated with the *XSLT stylesheets used to produce HTML files*. The following Ant properties can be overridden to specify your customization stylesheets:

   `args.wh.xsl`

   Specify this property if you want to customize the *XSLT stylesheet used to produce an HTML file for each topic*.

   `args.create.main.files.xsl`

   Specify this property if you want to customize the *XSLT stylesheet used to produce the main HTML file*.

   `args.createTocXML.xsl`

   Specify this property if you want to customize the *XSLT stylesheet used to produce the `toc.xml` file*. The toc.xml file contains information extracted from DITA map and it is mainly used to create the WebHelp TOC.

   For example, to customize a WebHelp Responsive transformation type, the build file should look like:

```
<project basedir="." name="Webhelp Responsive Customization">
```

```
  <target name="dita2webhelp-responsive-custom">
    <!--
      Override this property if you want to customize the XSLT stylesheet
      used to produce an HTML file for each topic
    -->
    <property
      name="args.wh.xsl"
      value="${dita.plugin.com.oxygenxml.webhelp.responsive.custom.dir}
                                                /xsl/dita2webhelpCustom.xsl"/>
    <!--
      Override this property if you want to customize the XSLT stylesheet
      used to produce the main HTML file.
    -->
    <property
      name="args.create.main.files.xsl"
      value="${dita.plugin.com.oxygenxml.webhelp.responsive.custom.dir}
                                            /xsl/createMainFilesCustom.xsl"/>
    <!--
      Override this property if you want to customize the XSLT stylesheet
      used to produce the toc.xml file.
    -->
    <property
      name="args.createTocXML.xsl"
      value="${dita.plugin.com.oxygenxml.webhelp.responsive.custom.dir}
                                            /xsl/createTocXMLCustom.xsl"/>
    <!--
      Depending on which version of WebHelp you want to customize,
      you need to delegate to different build targets:
      * dita2webhelp-responsive - when you are customizing the Webhelp Responsive
      * dita2webhelp-mobile - when you are customizing the Webhelp Mobile
      * dita2webhelp - when you are customizing the Webhelp Classic
    -->
    <antcall target="dita2webhelp-responsive"/>
  </target>
 </project>
```

Note: Depending on which version of WebHelp you want to customize, you need to call one of the following build targets:

- **dita2webhelp-responsive** - For WebHelp Responsive (with or without feedback) output.
- **dita2webhelp** - For WebHelp Classic (with or without feedback) output.
- **dita2webhelp-mobile** - For WebHelp Classic Mobile output (deprecated).

5. Create an **xsl** directory in the plugin customization directory (that you created in step 1) to store the customized XSLT stylesheets.

Referencing the WebHelp XSLT Stylesheets from Your Customizations

Note that your customization stylesheets should import the original stylesheets that they override. To reference the WebHelp stylesheets, you can use the **plugin:com.oxygenxml.dita-ot.plugin.webhelp** prefix. This prefix is rewritten by an XML catalog to the WebHelp root directory.

Customizing the `dita2webhelp.xsl` Stylesheet

The XSLT stylesheet that customizes the WebHelp **dita2webhelp.xsl** stylesheet should look like:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:math="http://www.w3.org/2005/xpath-functions/math"
   exclude-result-prefixes="xs math"
   version="2.0">
   <!--
     Import the original stylesheet used to produce an HTML file for each topic.
   -->
   <xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/
                                          dita/responsive/dita2webhelp.xsl"/>
   <!--
     Please add your customization templates here.
   -->
</xsl:stylesheet>
```

Depending on which version of WebHelp you want to customize, you need to import one of the following XSLT stylesheets:

- **dita2webhelp-responsive** (WebHelp Responsive) -

  `<xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/responsive/dita2webhelp.xsl"/>`
- **dita2webhelp** (WebHelp Classic) -

```
<xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/desktop/
dita2webhelp.xsl"/>
```

- **dita2webhelp-mobile** (WebHelp Classic Mobile) -

```
<xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/mobile/
dita2webhelp.xsl"/>
```

Customizing the `createMainFiles.xsl` Stylesheet

The XSLT stylesheet that customizes the WebHelp **createMainFiles.xsl** stylesheet should look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:math="http://www.w3.org/2005/xpath-functions/math"
    exclude-result-prefixes="xs math"
    version="2.0">
    <!--
        Import the original stylesheet used to produce the main HTML file.
    -->
    <xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/
                                            responsive/createMainFiles.xsl"/>
    <!--
        Please add your customization templates here.
    -->

</xsl:stylesheet>
```

Depending on which version of WebHelp you want to customize, you need to import one of the following XSLT stylesheets:

- **dita2webhelp-responsive** (WebHelp Responsive) -

```
<xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/
responsive/createMainFiles.xsl"/>
```

- **dita2webhelp** (WebHelp Classic) -

```
<xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/desktop/
createMainFiles.xsl"/>
```

- **dita2webhelp-mobile** (WebHelp Classic Mobile) -

```
<xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/mobile/
createMainFiles.xsl"/>
```

Customizing the `tocDita.xsl` File

The XSLT stylesheet that customizes the WebHelp **tocDita.xsl** stylesheet should look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:math="http://www.w3.org/2005/xpath-functions/math"
    exclude-result-prefixes="xs math"
    version="2.0">
    <!--
        Import the original stylesheet used to produce the toc.xml file.
    -->
    <xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/tocDita.xsl"/>
    <!--
        Please add your customization templates here.
    -->

</xsl:stylesheet>
```

Search Engine Optimization for DITA WebHelp

A **DITA Map WebHelp** transformation scenario can be configured to produce a `sitemap.xml` file that is used by search engines to aid crawling and indexing mechanisms. A *sitemap* lists all pages of a WebHelp system and allows webmasters to provide additional information about each page, such as the date it was last updated, change frequency, and importance of each page in relation to other pages in your WebHelp deployment.

The structure of the `sitemap.xml` file looks like this:

```
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.example.com/topics/introduction.html</loc>
    <lastmod>2014-10-24</lastmod>
```

```
    <changefreq>weekly</changefreq>
    <priority>0.5</priority>
  </url>
  <url>
    <loc>http://www.example.com/topics/care.html#care</loc>
    <lastmod>2014-10-24</lastmod>
    <changefreq>weekly</changefreq>
    <priority>0.5</priority>
  </url>
   .  .  .
</urlset>
```

Each page has a `<url>` element structure containing additional information, such as:

- `loc` - the URL of the page. This URL must begin with the protocol (such as `http`), if required by your web server. It is constructed from the value of the `webhelp.sitemap.base.url` parameter from the transformation scenario and the relative path to the page (collected from the `href` attribute of a `topicref` element in the *DITA map*).

  Note: The value must have fewer than 2,048 characters.
- `lastmod` (optional) - the date when the page was last modified. The date format is `YYYY-MM-DD`.
- `changefreq` (optional) - indicates how frequently the page is likely to change. This value provides general information to assist search engines, but may not correlate exactly to how often they crawl the page. Valid values are: `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, and `never`. The first time the `sitemap.xml` file is generated, the value is set based upon the value of the `webhelp.sitemap.change.frequency` parameter in the DITA WebHelp transformation scenario. You can change the value in each `url` element by editing the `sitemap.xml` file.

  Note: The value `always` should be used to describe documents that change each time they are accessed. The value `never` should be used to describe archived URLs.
- `priority` (optional) - the priority of this page relative to other pages on your site. Valid values range from 0.0 to 1.0. This value does not affect how your pages are compared to pages on other sites. It only lets the search engines know which pages you deem most important for the crawlers. The first time the `sitemap.xml` file is generated, the value is set based upon the value of the `webhelp.sitemap.priority` parameter in the DITA WebHelp transformation scenario. You can change the value in each `url` element by editing the `sitemap.xml` file.

Creating and Editing the `sitemap.xml` File

Follow these steps to produce a `sitemap.xml` file for your WebHelp system, which can then be edited to fine-tune search engine optimization:

1. **Edit** the transformation scenario you currently use for obtaining your WebHelp output. This opens the **Edit DITA Scenario** dialog box.
2. Open the **Parameters** tab and set a value for the following parameters:
   - `webhelp.sitemap.base.url` - the URL of the location where your WebHelp system is deployed

     Note: This parameter is required for Oxygen XML Editor to generate the `sitemap.xml` file.
   - `webhelp.sitemap.change.frequency` - how frequently the WebHelp pages are likely to change (accepted values are: `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, and `never`)
   - `webhelp.sitemap.priority` - the priority of each page (value ranging from 0.0 to 1.0)
3. Run the transformation scenario.
4. Look for the `sitemap.xml` file in the transformation's output folder. Edit the file to fine-tune the parameters of each page, according to your needs.

Support for Right-to-Left (RTL) Oriented Languages for DITA WebHelp

To activate support for RTL (right-to-left) languages in WebHelp output, edit the *DITA map* and set the `xml:lang` attribute on its root element (`map`). The corresponding attribute value can be set for following RTL languages:

- `ar-eg` - Arabic
- `he-il` - Hebrew
- `ur-pk` - Urdu

WebHelp Responsive Runtime Additional Parameters

A deployed WebHelp Responsive system can accept the following GET parameters:

- `contextId` - The WebHelp JavaScript engine will look for this value in the `context-help-map.xml` mapping file and load the corresponding help page. For more information, see the *Context-Sensitive WebHelp System* topic.

  Note: You can use an *anchor* in the `contextId` parameter to jump to a specific section in a document. For example, `contextId=topicID#anchor`.
- `appname` - You can use this parameter in conjunction with `contextId` to search for this value in the corresponding *appname attribute value* in the mapping file.

  ```
  http://localhost/webhelp/index.html?contextId=topicID&appname=myApplication
  ```
- `searchQuery` - You can use this parameter to perform a search operation when WebHelp is loaded. For example, if you want to open WebHelp showing all search results for *growing flowers*, the URL should look like this: `http://localhost/webhelp/index.html?searchQuery=growing%20flowers`.

Related Information:

Context-Sensitive WebHelp Responsive System on page 986

Context-Sensitive WebHelp Responsive System

Context-sensitive help systems assist users by providing specific informational topics for certain components of a user interface, such as a button or window. This mechanism works based on mappings between a unique ID defined in the topic and a corresponding HTML page.

Generating Context-Sensitive Help

When WebHelp Responsive output is generated by Oxygen XML Editor, the transformation process produces an XML mapping file called `context-help-map.xml` and copies it in the output folder of the transformation. This XML file maps an ID to a corresponding HTML page through an `appContext` element, as in the following example:

```
<map productID="oxy-webhelp" productVersion="1.1">
  <appContext helpID="myapp-functionid1" path="tasks/app-help1.html"/>
  <appContext helpID="myapp-functionid2" path="tasks/app-help1.html"/>
    .  .  .
</map>
```

The possible attributes are as follows:

`helpID`

A Unique ID provided by a topic from two possible sources (`resourceid` element or `id` attribute):

`resourceid`

The `resourceid` element is mapped into the `appContext` element and can be specified in either the `topicref` within a *DITA map* or in a `prolog` within a DITA topic. The `resourceid` element accepts the following attributes:

- **`appname`** - A name for the external application that references the topic. If this attribute is not specified, its value is considered to be empty (" ").
- **`appid`** - An ID used by an application to identify the topic.
- **`id`** - Specifies a value that is used by a specific application to identify the topic, but this attribute is ignored if an **`appid`** attribute is used.

Note: Multiple `appid` values can be associated with a single `appname` value (and multiple `appname` values can be associated with a single `appid` value), but the values for both attributes work in combination to specify a specific ID for a specific application, and therefore each combination of values for the `appid` and `appname` attributes should be unique within the context of a single *root map*. For example, suppose that you need two different functions of an application to both open the same WebHelp page.

Example: `resourceid` Specified in a DITA Map

The `resourceid` element can be specified in a `topicmeta` element within a `topicref`.

```
<map title="App Help">
  <topicref href="app-help1.dita" type="task">
     <topicmeta>
       <resourceid appname="myapp" appid="functionid1"/>
       <resourceid appname="myapp" appid="functionid2"/>
     </topicmeta>
  </topicref>
</map>
```

Example: `resourceid` Specified in a DITA Topic

The `resourceid` element can be specified in a `prolog` element within a DITA topic.

```
<task id="app-help1">
  <title>My App Help</title>
  <prolog>
    <resourceid appname="myapp" appid="functionid1"/>
    <resourceid appname="myapp" appid="functionid2"/>
  </prolog>
...
</task>
```

For more information about the `resourceid` element, see *DITA Specifications: <resourceid>*.

`id`

If a `resourceid` element is not declared in the *DITA map* or DITA topic (as described above), the `id` attribute that is set on the topic root element is mapped into the `appContext` element.

Important: You should ensure that these defined IDs are unique in the context of the entire DITA project. If the IDs are not unique, the transformation scenario will display warning messages in the transformation console output and the help system will not work properly.

`path`

The path to a corresponding WebHelp page. This path is relative to the location of the `context-help-map.xml` mapping file.

`productID` (Applicable only if you are using the WebHelp Responsive with Feedback transformation scenario)

The ID of the product for your documentation project.

`productVersion` (Applicable only if you are using the WebHelp Responsive with Feedback transformation scenario)

The version of the product for your documentation project.

There are two ways of implementing context-sensitive help in your system:

- The XML mapping file can be loaded by a PHP script on the server side. The script receives the `contextId` value and will look it up in the XML file.
- Invoke the `index.html` WebHelp system file and pass the `contextId` parameter with a specific value. The WebHelp system will automatically open the help page associated with the value of the `contextId` parameter.

```
index.html?contextId=myDITATopic
```

Context-Sensitive Queries

You can use the URL field in your browser to search for topics in a context-sensitive WebHelp system with the assistance of the following parameters:

- `contextId` - The WebHelp JavaScript engine will look for this value in the `context-help-map.xml` mapping file and load the corresponding help page. For more information, see the *Context-Sensitive WebHelp System* topic.

  Note: You can use an *anchor* in the `contextId` parameter to jump to a specific section in a document. For example, `contextId=topicID#anchor`.

- `appname` - You can use this parameter in conjunction with `contextId` to search for this value in the corresponding *appname attribute value* in the mapping file.

```
http://localhost/webhelp/index.html?contextId=topicID&appname=myApplication
```

Related Information:

WebHelp Responsive Runtime Additional Parameters on page 986

Publishing WebHelp Responsive Output on a SharePoint Site

Since WebHelp output must be published locally, on the same machine where the WebHelp process is running, to publish your files directly to a SharePoint library you need to map a network drive to connect to SharePoint and change your file extensions to `.aspx`, as described in the steps below.

To publish WebHelp Responsive output on a SharePoint site, follow this procedure:

1. Map a network drive to connect to your SharePoint library. For more information, see: *https://support.microsoft.com/en-us/kb/2616712*.
2. To allow browsers to open your published files (rather than downloading them), you need to change the file extensions from `.html` to `.aspx`. Fortunately, this can be done in the transformation scenario.
   a. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.
   b. Set the `args.outext` parameter to `.aspx`.
   c. Run the transformation scenario.

WebHelp Classic System

WebHelp is a form of online help that consists of a series of web pages (XHTML format). Its advantages include platform independence, ability to update content continuously, and it can be viewed using a regular web browser. The Oxygen XML Editor WebHelp system includes several variants to suit your specific needs. The **WebHelp Classic** variant is designed for desktop systems when feedback from users is not necessary and it is available for DocBook and DITA document types.

Layout of the WebHelp Classic System Interface

The layout of the **WebHelp Classic** system is comprised of the following components:

Left Pane or Frame

This section on the left side of the help system includes the following tabs:

Content

A typical table of contents style presentation of your content. You can use the **Expand all/** **Collapse all** buttons to expand or collapse all the topics presented in the Table of Contents.

Note: You can enhance the appearance of items in the *Table of Contents*. See the *Customizing WebHelp Classic Systems chapter* for more details.

Index

Presents the index terms for your content. If your content does not contain any `indexterm` elements, this tab is not generated.

Search Results

This tab is generated when the **Search** field is used. It presents the search results in the form of links to topics where the search terms are found, along with a rating scheme for each result. For more details, see the *Search Feature section*.

Upper Pane or Frame

The upper section of the help system includes the following features:

Search Field

Use this feature to perform searches in your content. When you enter search terms in this field, the results are displayed in the **Search Results** tab in the left section of the help system, along with a rating scheme for each result. For more details, see the *Search Feature section*.

Frames Option

Click on this option to display the output rendered in HTML frames.

Print Option

Opens a dialog box with various printing options and a print preview.

Navigation Links

You can navigate through the content of your output using the navigation links or arrows in the upper-right part of the page. These arrows allow you to move to the **Parent topic**, **Previous topic**, or **Next topic**. Links to the parent topics of the currently opened topic are also presented at the top of the page.

Note: You can edit the `args.hide.parent.link` parameter to hide the **Parent**, **Next**, and **Previous** links.

Main Pane or Frame

The content of the help pages are rendered and displayed in this main section.

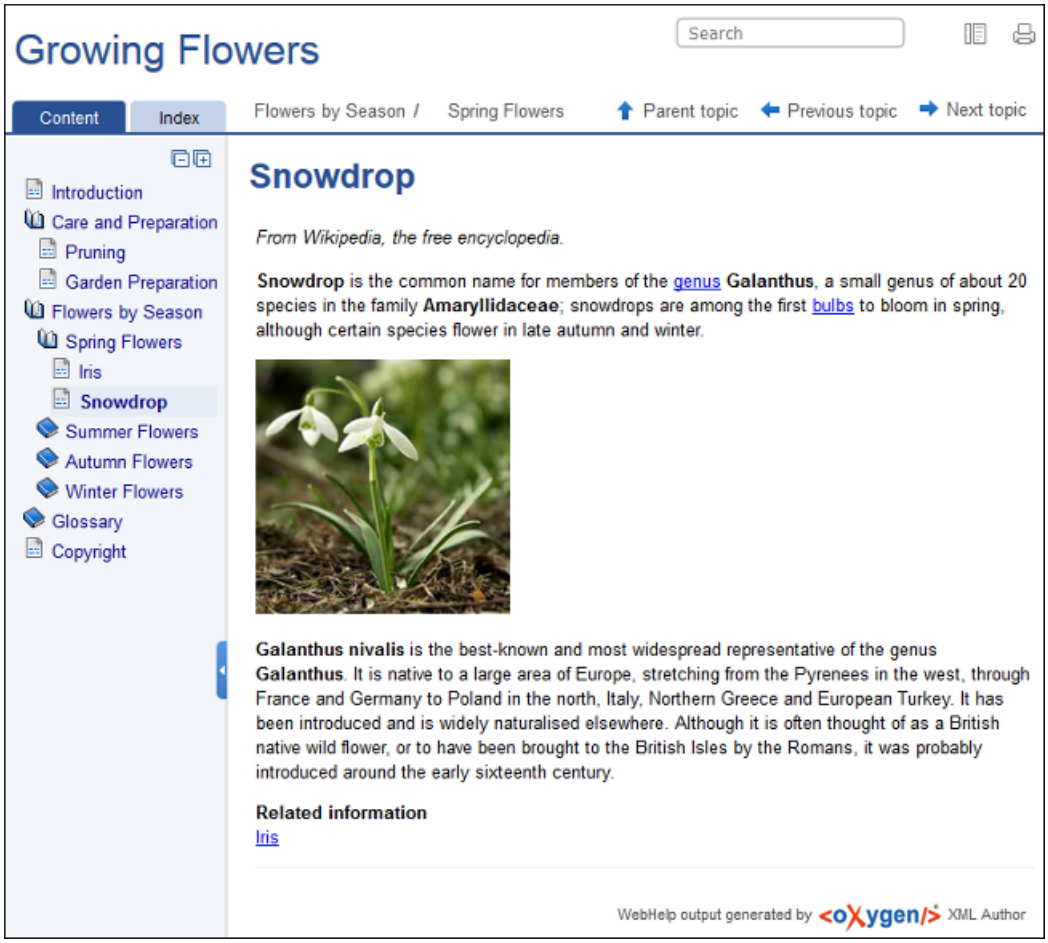


Figure 487: WebHelp Classic Output

WebHelp Classic Search Engine
Search Rules

Rules that are applied during a search include:

- You can use quotes to perform an exact search for multiple word phrases (for example, *"grow flowers"* will only return results if both words are found consecutively and exactly as they are typed in the search field). This type of search is known as a *phrase search*.
- *Boolean Search* is supported using the following operators: *and*, *or*, *not*. When there are two adjacent search terms without an operator, *or* is used as the default search operator (for example, *grow flowers* is the same as *grow or flowers*).

- The space character separates keywords (an expression such as *grow flowers* counts as two separate keywords: *grow* and *flowers*).
- `indexterm` and `keywords` DITA elements are an effective way to increase the ranking of a page (for example, content inside a *keywords* element weighs more than an *H1* HTML element).
- Words composed by merging two or more words with colon (":"), minus ("-"), underline ("_"), or dot (".") characters count as a single word.
- Always search for words containing three or more characters (shorter words, such as *to* or *of* are ignored). This rule does not apply to CJK (Chinese, Japanese, Korean) languages.

5-Star Rating Mechanism and Sorting

The **Search** feature is also enhanced with a rating mechanism that computes scores for every result that matches the search criteria. These scores are then translated into a 5-star rating scheme and the stars are displayed to the right of each result. The search results are sorted depending on the following:

- Search entries that satisfy the phrase search criterion are presented first.
- The number of keywords found in a single page (the higher the number, the better).
- The context (for example, a word found in a title scores better than a word found in unformatted text). The search ranking order, sorted by relevance is as follows:
  - The search term is included in a meta keyword
  - The search term is in the title of the page
  - The search term is in bold text in a paragraph
  - The search term is in normal text in a paragraph

Excluded Terms

To improve performance, the **Search** feature excludes certain *stop words*. For example, the English version of such *stop words* include: *a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with*.

WebHelp Classic Search Results Tab

When you enter search terms in the **Search** field at the top of the help system, the results are displayed in the **Search Results** tab in the left section. When you click on a result in the **Search Results** tab, that result is displayed in the main pane with the search terms highlighted. If you press **Enter** with the **Search** field empty, the highlights are removed.

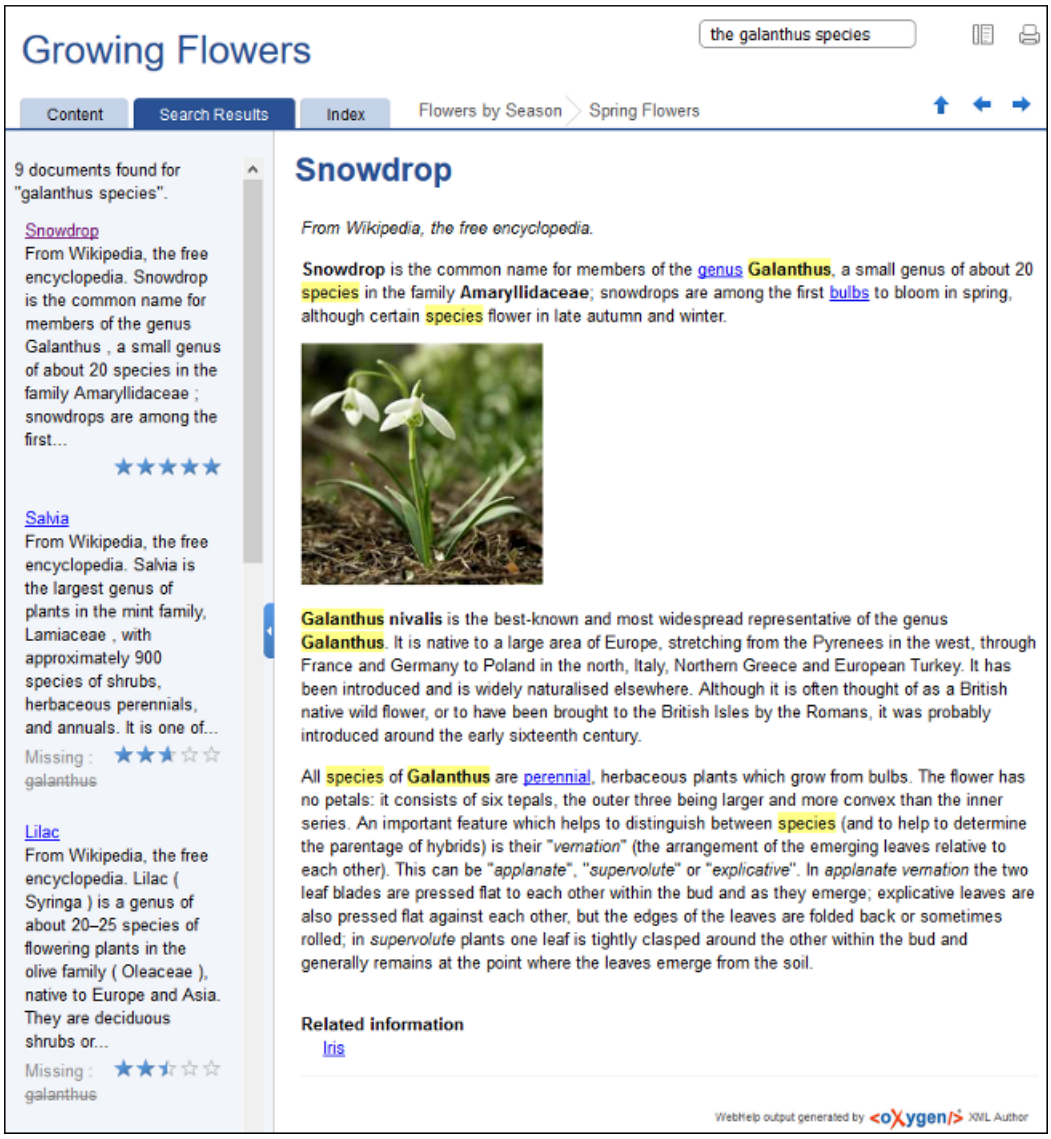


Figure 488: WebHelp Classic Search Results Tab

Missing Terms

If you enter multiple search terms (other than *stop words*), for any result that the search engine found at least one term but not one or more of the other terms, the **Missing** terms will be listed below each result.

Tag Element Scoring Values

HTML tag elements are also assigned a scoring value and these values are evaluated for the search results. For information about editing these values, see *Editing Scoring Values of Tag Elements in Search Results* on page 972.

Browser Compatibility

This output format is compatible with the most recent versions of the following common browsers:

- Edge
- Internet Explorer (IE 9 or newer)
- Chrome
- Firefox
- Safari
- Opera

Important: Due to some security restrictions in certain browsers (Google Chrome and Internet Explorer), WebHelp Classic pages loaded from the local system (through URLs of the `file:///...` format) may not work properly. We recommend that you load WebHelp Classic pages in Google Chrome or Internet Explorer only from a web server (with a URL such as `http://your.server.com/webhelp/index.html` or `http://localhost/web_pages/index.html`).

Warning: Due to some restrictions in web browsers in regards to JavaScript code, the *frameless* version (`index.html` start page) of the WebHelp Classic system should only be loaded from a web server (with a URL such as `http://your.server.com/webhelp/index.html` or `http://localhost/web_pages/index.html`). When loading WebHelp Classic pages from the local file system, the *frameset* version (`index_frames.html` start page) of the WebHelp Classic system should be used instead (`file:///...`).

WebHelp Classic with Feedback System

WebHelp is a form of online help that consists of a series of web pages (XHTML format). Its advantages include platform independence, ability to update content continuously, and a feedback mechanism that allows your authors and audience to interact with one another through comments. The Oxygen XML Editor WebHelp system includes several variants to suit your specific needs. The **WebHelp Classic with Feedback** variant is designed for desktop systems, includes a feedback system that allows your users to make comments and allows you to manage and reply to them, and it is available for DocBook and DITA document types.

Layout

The layout of the **WebHelp Classic with Feedback** system is comprised of the following components:

Left Pane or Frame

This section on the left side of the help system includes the following tabs:

Content

A typical table of contents style presentation of your content. You can use the **Expand all/ Collapse all** buttons to expand or collapse all the topics presented in the Table of Contents.

Note: You can enhance the appearance of items in the *Table of Contents*. See the *Customizing WebHelp Classic Systems chapter* for more details.

Index

Presents the index terms for your content. If your content does not contain any `indexterm` elements, this tab is not generated.

Search Results

This tab is generated when the **Search** field is used. It presents the search results in the form of links to topics where the search terms are found, along with a rating scheme for each result. For more details, see the *Search Feature section*.

Upper Pane or Frame

The upper section of the help system includes the following features:

Search Field

Use this feature to perform searches in your content. When you enter search terms in this field, the results are displayed in the **Search Results** tab in the left section of the help system, along with a rating scheme for each result. For more details, see the *Search Feature section*.

Frames Option

Click on this option to display the output rendered in HTML frames.

Print Option

Opens a dialog box with various printing options and a print preview.

Navigation Links

You can navigate through the content of your output using the navigation links or arrows in the upper-right part of the page. These arrows allow you to move to the **Parent topic**, **Previous topic**, or **Next topic**. Links to the parent topics of the currently opened topic are also presented at the top of the page.

Note: You can edit the `args.hide.parent.link` parameter to hide the **Parent**, **Next**, and **Previous** links.

Main Pane or Frame

The content of the help pages are rendered and displayed in this main section.

Feedback Section

The **WebHelp Classic with Feedback** system contains a **Comments** bar at the bottom of the main pane. This section is where you can interact with users through a comment system.

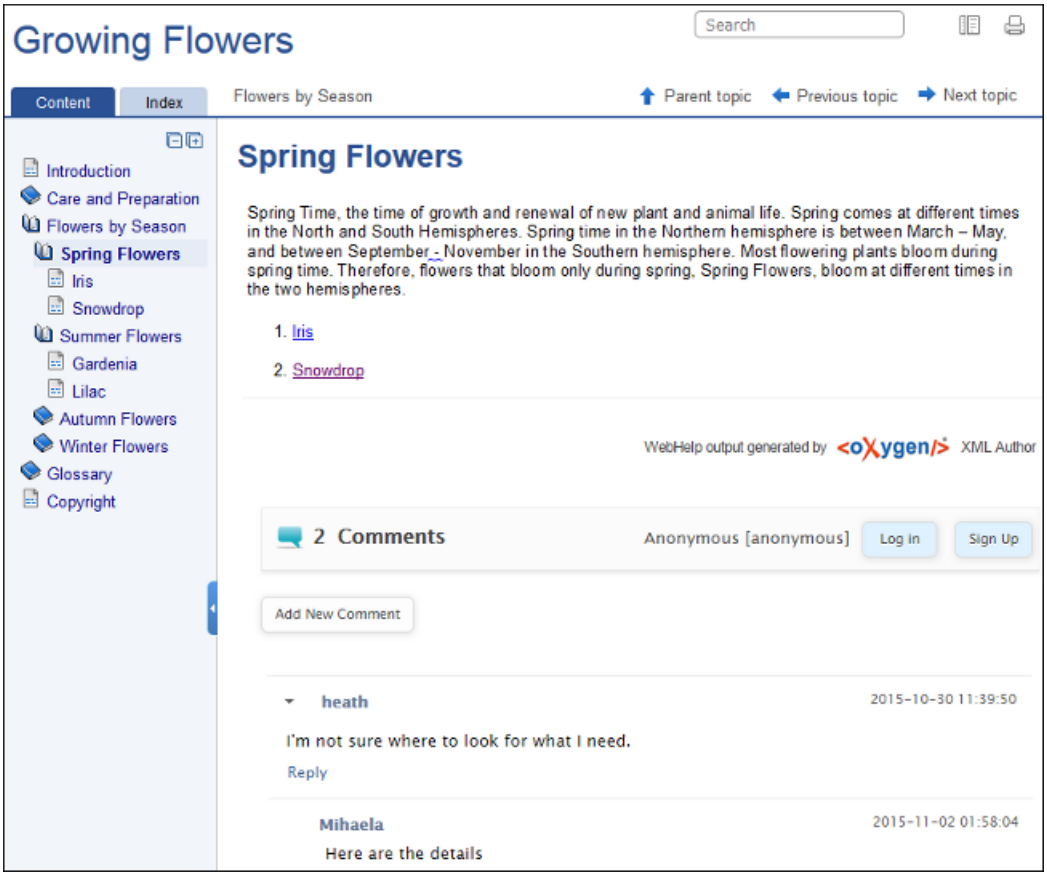


Figure 489: WebHelp Classic with Feedback System

Managing Comments

To add a new comment, click the **Add New Comment** button, or click **Reply** to add a comment to an existing thread. You can click on the **Log in** button on the right side of this bar to be authenticated as a user and your user name will be included in any comments that you add. If you do not have a user name, you can click on the **Sign Up** button to create a new user.

After you log in, your name and user name are displayed in the **Comments** bar, along with the **Log off** and **Edit** buttons. Click the **Edit** button to open the **User Profile** dialog box where you can customize the following options:

- **Your Name** - You can use this field to edit the initial name that you used to create your user profile.
- **Your email address** - You can use this field to edit the initial email address that you used to create your profile.
- You can choose to receive an email in the following situations:
  - When a comment is left on a page that you commented on.
  - When a comment is left on any topic in the WebHelp Classic system.
  - When a reply is left to one of my comments.
- **New Password** - Allows you to enter a new password for your user account.

  Note: The **Current Password** field from the top of the **User Profile** is mandatory if you want to save the changes you make.

If you are an administrator, you can manage user information and comments. For more information, see *Managing Users and Comments in a Feedback-Enabled WebHelp System* on page 949.

WebHelp Classic Search Engine

Search Rules

Rules that are applied during a search include:

- You can use quotes to perform an exact search for multiple word phrases (for example, *"grow flowers"* will only return results if both words are found consecutively and exactly as they are typed in the search field). This type of search is known as a *phrase search*.
- *Boolean Search* is supported using the following operators: *and*, *or*, *not*. When there are two adjacent search terms without an operator, *or* is used as the default search operator (for example, *grow flowers* is the same as *grow or flowers*).
- The space character separates keywords (an expression such as *grow flowers* counts as two separate keywords: *grow* and *flowers*).
- `indexterm` and `keywords` DITA elements are an effective way to increase the ranking of a page (for example, content inside a *keywords* element weighs more than an *H1* HTML element).
- Words composed by merging two or more words with colon (":"), minus ("-"), underline ("_"), or dot (".") characters count as a single word.
- Always search for words containing three or more characters (shorter words, such as *to* or *of* are ignored). This rule does not apply to CJK (Chinese, Japanese, Korean) languages.

5-Star Rating Mechanism and Sorting

The **Search** feature is also enhanced with a rating mechanism that computes scores for every result that matches the search criteria. These scores are then translated into a 5-star rating scheme and the stars are displayed to the right of each result. The search results are sorted depending on the following:

- Search entries that satisfy the phrase search criterion are presented first.
- The number of keywords found in a single page (the higher the number, the better).
- The context (for example, a word found in a title scores better than a word found in unformatted text). The search ranking order, sorted by relevance is as follows:
  - The search term is included in a meta keyword
  - The search term is in the title of the page
  - The search term is in bold text in a paragraph
  - The search term is in normal text in a paragraph

Excluded Terms

To improve performance, the **Search** feature excludes certain *stop words*. For example, the English version of such *stop words* include: *a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with*.

WebHelp Classic Search Results Tab

When you enter search terms in the **Search** field at the top of the help system, the results are displayed in the **Search Results** tab in the left section. When you click on a result in the **Search Results** tab, that result is displayed in the main pane with the search terms highlighted. If you press **Enter** with the **Search** field empty, the highlights are removed.

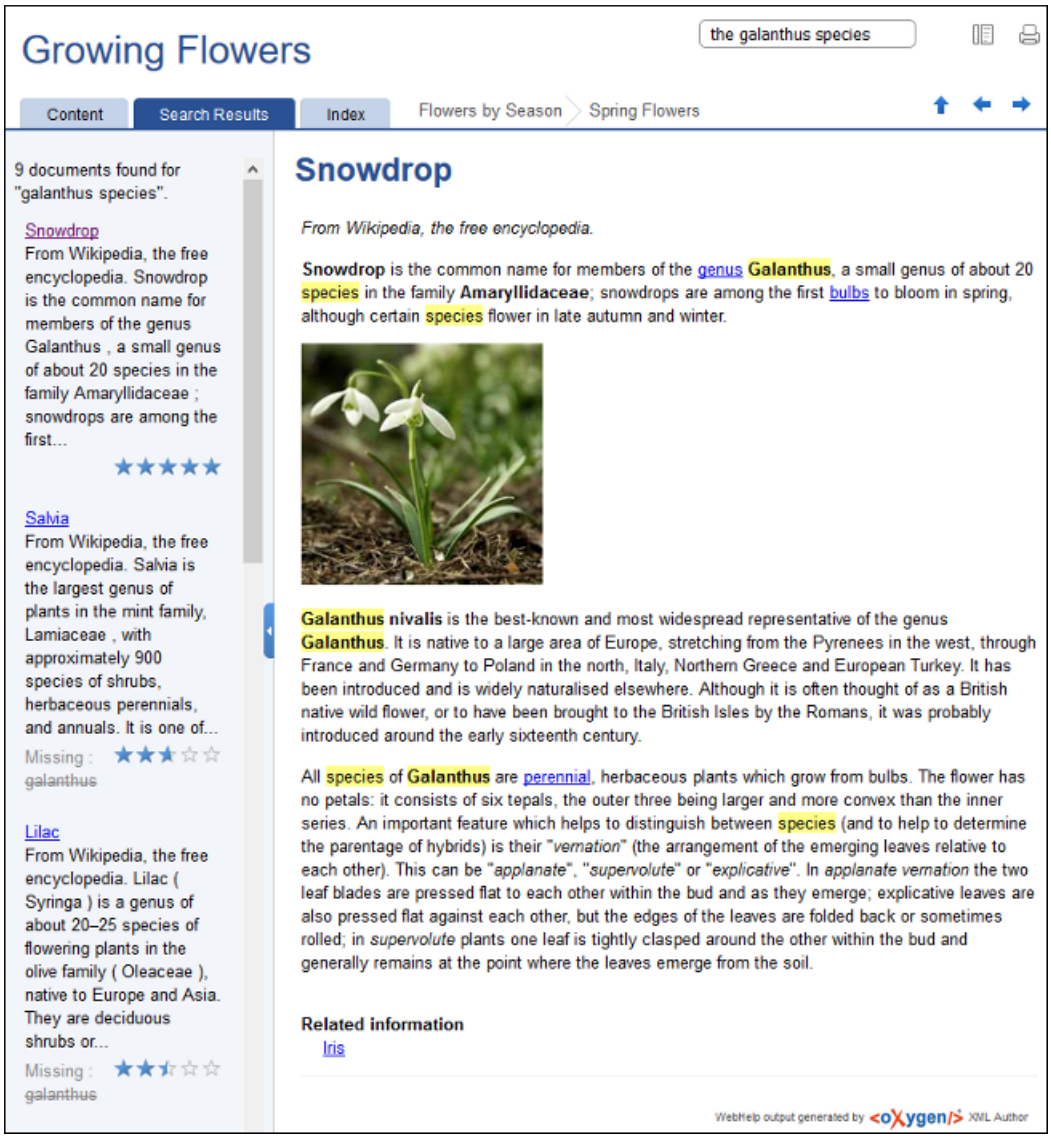


Figure 490: WebHelp Classic Search Results Tab

Missing Terms

If you enter multiple search terms (other than *stop words*), for any result that the search engine found at least one term but not one or more of the other terms, the **Missing** terms will be listed below each result.

Tag Element Scoring Values

HTML tag elements are also assigned a scoring value and these values are evaluated for the search results. For information about editing these values, see *Editing Scoring Values of Tag Elements in Search Results* on page 972.

Browser Compatibility

This output format is compatible with the most recent versions of the following common browsers:

- Edge
- Internet Explorer (IE 9 or newer)
- Chrome
- Firefox
- Safari
- Opera

Important: Due to some security restrictions in certain browsers (Google Chrome and Internet Explorer), WebHelp Classic pages loaded from the local system (through URLs of the `file:///...` format) may not work properly. We recommend that you load WebHelp Classic pages in Google Chrome or Internet Explorer only from a web server (with a URL such as `http://your.server.com/webhelp/index.html` or `http://localhost/web_pages/index.html`).

Deploying a Feedback-Enabled WebHelp System

System Requirements

The feedback-enabled WebHelp system of Oxygen XML Editor requires a standard server deployment. You can request this from your server admin and it needs the following system components:

- A Web server (such as *Apache Web Server*)
- A *MySQL* or *MariaDB* database server
- A database admin tool (such as *phpMyAdmin*)
- PHP Version 5.1.6 or later

Oxygen XML WebHelp system supports most of the recent versions of the following browsers: Chrome, Firefox, Edge, Internet Explorer, Safari, Opera.

Create WebHelp with Feedback Database

The **WebHelp with Feedback** system needs a database to store user details and the actual feedback, and a user added to it with all privileges. After this is created, you should have the following information:

- Database name
- Username
- Password

Exactly how you create the database and user depends on your web host and your particular needs.

Example:

The following procedure uses *phpMyAdmin* to create a MySQL database for the feedback system and a MySQL user with privileges for that database. The feedback system uses these credentials to connect to the database.

Using *phpMyAdmin* to create a database:

1. Access the *phpMyAdmin* instance running on your server.
2. Click *Databases* (in the right frame) and then create a *database*. You can give it any name you want (for example *comments*).
3. Create a user with connection privileges for this database.
4. Under *localhost*, in the right frame, click *Privileges* and then at the bottom of the page click the **reload the privileges** link.

Deploying the WebHelp with Feedback Output

If you have a web server configured with PHP and MySQL, you can deploy the **WebHelp with Feedback** output by following these steps:

1. Connect to your server using an FTP client.
2. Locate the home directory (from now on, referred to as *DOCUMENT_ROOT*) of your server.
3. Copy the transformation output folder into the `DOCUMENT_ROOT` folder.
4. Rename it to something relevant (for example, `myProductWebHelp`).
5. Open the output folder (for example, `http://[YOUR_SERVER]/myProductWebHelp/`). You are redirected to the installation wizard. Proceed with the installation as follows:
   a. Verify that the prerequisites are met.
   b. Press **Start Installation**.
   c. Configure the **Deployment Settings** section. Default values are provided, but you should adjust them as needed.

Tip: You can change some of the options later. The installation creates a `config.php` file in `[OXYGEN_WEBHELP_INSTALL_DIR]/feedback/resources/php/config/config.php` where all your configuration options are stored.

d. Configure the **MySql Database Connection Settings** section. Use the information (database name, username, password) from the *Create WebHelp with Feedback Database section* to fill-in the appropriate text boxes.

Warning: Selecting the **Create new database structure** option will overwrite any existing data in the selected database, if it already exists. Therefore, it is useful the first time you install the **WebHelp with Feedback** system, but you do not want to select this option on subsequent deployments.

e. If you are using a domain (such as *OpenLDAP* or *Active Directory*) to manage users in your organization, select the **Enable LDAP Autehntication** option. This will allow you to configure the LDAP server, which will provide information and credentials for users who will access the WebHelp system. Also, this will allow you to choose which of the domain users will have administrator privileges.

f. If the **Create new database structure** option is selected, the **Create WebHelp Administrator Account** section becomes available. Here you can set the administrator account data. The administrator is able to moderate new posts and manage WebHelp users.

The same database can be used to store comments for multiple **WebHelp with Feedback** deployments. If a topic is available in multiple deployments and there are comments associated with it, you can choose to display the comments in all deployments that share the database. To do this, select the **Display comments from other products** option. In the **Display comments from** section, a list with the deployments sharing the same database is displayed. Select the deployments allowed to share common feedback.

Note: You can restrict the displayed comments of a product depending on its version. If you have two products that use the same database and you restrict one of them to display comments starting from a certain version, the comments of the other product are also displayed from the specified version onwards.

g. Press **Next Step**.

h. Remove the installation folder from your web server.

Important: When you publish subsequent iterations of your **WebHelp with Feedback** system, you will not upload the `/install` folder in the output, as you only need it uploaded the first time you create the installation. On subsequent uploads, you will just upload the other output files.

i. In your Web browser, go to your **WebHelp with Feedback** system main page.

Testing Your WebHelp with Feedback System

To test your system, create a user and post a comment. Check to see if the notification emails are delivered to your email inbox.

Note: To read debug messages generated by the system:

1. Enable *JavaScript* logging by doing one of the following:
   - Open the `log.js` file, locate the `var log= new Log(Level.NONE);` line, and change the logging level to: `Level.INFO`, `Level.DEBUG`, `Level.WARN`, or `Level.ERROR`.
   - Append `?log=true` to the WebHelp URL.
2. Inspect the PHP and Apache server log files.

Documentation Product ID and Version

When you run a **WebHelp with Feedback** transformation scenario, by default you are prompted for a documentation product ID and version number. This is needed when multiple WebHelp systems are deployed on the same server. Think of your WebHelp output as a *product*. If you have three different WebHelp outputs, you have three different *products* (each with their own unique documentation product ID). This identifier is included in a configuration file so that comments are tied to a particular output (product ID and version number).

Note: The **WebHelp with Feedback** installation includes a configuration option (**Display comments from other products**) that allows you to choose to have comments visible in other specified *products*.

Related Information:

Managing Users and Comments in a Feedback-Enabled WebHelp System on page 949

Refreshing the Content of a Feedback-Enabled WebHelp System

It is common to update the content of an existing installation of a **WebHelp with Feedback** system on a regular basis. In this case, reinstalling the whole system is not a viable option since it might result in the loss of the comments associated with your topics. Also, reconfiguring the system every time you want to refresh it may be time consuming.

Fortunately, you can refresh just the content without losing the comments or the initial system configuration. To do so, follow these steps:

1. Execute the transformation scenario that produces the *WebHelp with Feedback* output directory.
2. Go to the output directory (specified in the **Output** tab of the transformation scenario), locate the `\feedback\resources\php\config\config.php` file, and delete it.
3. Locate the `\feedback\install` directory and delete it.
4. Copy the remaining structure of the output folder and paste it into your *WebHelp with Feedback* system installation directory, overwriting the existing content.

Managing Users and Comments in a Feedback-Enabled WebHelp System

When you installed the **WebHelp with Feedback** system the first time (assuming the ***Create new database structure*** *option* was selected), you should have been prompted to create an administrator account (or a user named `administrator` was created by default). As an administrator, you have access to manage comments posted in your feedback-enabled WebHelp system. You can also manage the user information (such as role, status, or notification options).

To manage comments and user information, follow these steps:

1. At the bottom of each specific topic there is a **Comments** navigation bar and on the right side there is a **Log in** button. Click this button and log in with your administrator credentials. This gives you access to an **Admin Panel** button.
2. Click the **Admin Panel** button to display an administration page.

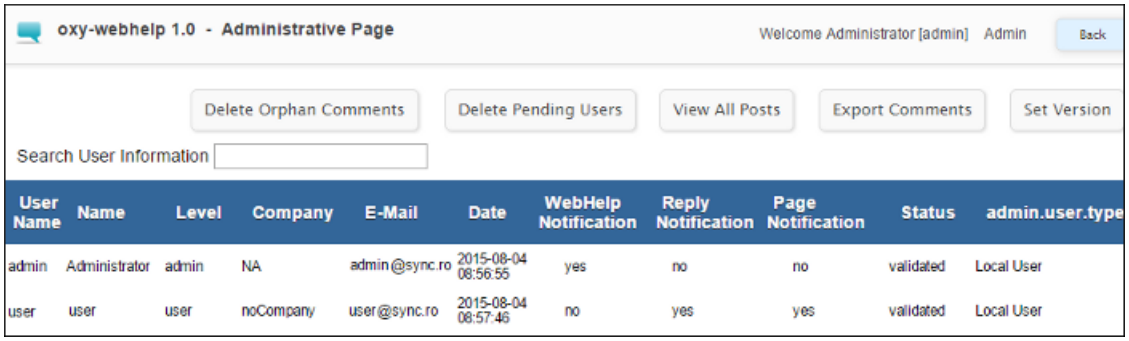


Figure 491: Administrative Page

3. Use this page to manage the following options:

   Delete Orphaned Comments

   Allows you to delete comments that are no longer associated with a topic in your WebHelp system.

   Delete Pending Users

   Allows you to delete user accounts that you do not wish to activate.

   View All Posts

   Allows you to view all the comments that are associated with topics in your WebHelp system.

   Export Comments

   Allows you to export all posts associated with topics in your WebHelp system into an XML file.

   Set Version

   Use this action to display comments starting with a particular version.

   Manage User Information

   To edit the details for a user, click on the corresponding row. This opens a window that allows you to customize the following information associated with the user:

Name

The full name of the user.

Level

Use this field to modify the privilege level (role) for the selected user. You can choose from the following:

- **User** - Regular user, able to post comments and receive e-mail notifications.
- **Moderator** - In addition to the regular **User** rights, this type of user has access to the **Admin Panel** where a moderator can view, delete, export comments, and set the version of the feedback-enabled WebHelp system.
- **Admin** - Full administrative privileges. Can manage WebHelp-specific settings, users, and their comments.

Company

The name of the organization associated with the user.

E-Mail

The contact email address for the user. This is also the address where the WebHelp system sends notifications.

WebHelp Notification

When selected, the user receives notifications when comments are posted anywhere in your feedback-enabled WebHelp system.

Reply Notification

When selected, the user receives notifications when comments are posted as a reply to one of their comments.

Page Notification

When selected, the user receives notifications when comments are posted on a topic where they previously posted a comment.

Date

The date the user registered is displayed.

Status

Use this drop-down list to change the status of the user. You can choose from the following:

- **Created** - The user is created but does not yet have any rights for the feedback-enabled WebHelp system.
- **Validated** - The user is able to use the feedback-enabled WebHelp system.
- **Suspended** - The user has no rights for the feedback-enabled WebHelp system.

Warning: The key used for identifying the page a comment is attached to is the relative file path to the output page. Since the output file and folder names mirror the source, any change to the file name (or its folder) in the source will affect the comments associated with that WebHelp page. If you change the file name or path, the comment history for that topic will become orphaned (a change to the topic ID does not affect the comment history).

Customizing WebHelp Classic Systems

To change the overall appearance of the WebHelp output, you can use the visual *WebHelp Skin Builder tool*, which does not require knowledge of CSS language. If you are familiar with CSS and coding, you can style your WebHelp output through your own custom stylesheets. You can also customize your output by modifying option and parameters in the transformation scenario.

This section includes topics that explain various ways to customize your WebHelp system output, such as how to improve the appearance of the Table of Contents, add logo images in the title area, integrate with social media, add custom headers and footers, and much more.

Copying Additional Resources to WebHelp Output Directory

To copy additional resources (such as JavaScript, CSS or other resources) to the output directory of a WebHelp system, follow these steps:

1. Place all your resources in the same directory.
2. Edit the WebHelp transformation scenario.
3. Go to the **Parameters** tab.
4. Edit the value for the webhelp.custom.resources parameter and set it to the absolute path of the directory in step 1.
5. Click **OK** to save the changes to the transformation scenario.
   All files from the new directory will be copied to the root of the WebHelp output directory.

Adding Custom HTML Content in WebHelp Classic Output

You can add custom HTML content in the WebHelp Classic output by inserting it in a well-formed XML file that will be referenced in the transformation scenario. This content may include references to additional JavaScript, CSS, and other types of resources, or such resources can be inserted inline within the HTML content that is inserted in the XML file.

To include custom HTML content in the WebHelp Classic output files, follow these steps:

1. Insert the HTML content in a well-formed XML file. There are several things to consider in regards to this XML file:

   a. **Well-Formedness** - If the file is not a *Well-formed XML document* (or fragments are not well-formed), the transformation will fail.

      A common use case is if you want to include several `script` or `link` elements. In this case, the XML fragment has multiple root elements and to make it well-formed, you can wrap it in an `html` element. This element tag will be filtered out and only its children will be copied to the output documents. Similarly, you can wrap your content in `head`, `body`, `html/head`, or `html/body` elements.

   b. **Referencing Resources in the XML File** - You can include references to local resources (such as JavaScript or CSS files) by using the predefined `${oxygen-webhelp-output-dir}` macro to specify their paths relative to the output directory:

      ```
      <html>
        <script type="text/javascript" src="${oxygen-webhelp-output-dir}/js/test.js"/>
        <link rel="stylesheet" type="text/css"
              href="${oxygen-webhelp-output-dir}/css/test.css" />
      </html>
      ```

      To copy the referenced resources to the output directory, follow the procedure in: *Copying Additional Resources to WebHelp Output Directory* on page 966.

   c. **Inline JavaScript or CSS Content** - If you want to include inline JavaScript or CSS content in the XML file, it is important to place this content inside an XML comment, as in the following examples:

      JavaScript:

      ```
      <script type="text/javascript">
        <!--
          /* Include JavaScript code here. */

          function myFunction() {
            return true;
          }
        -->
      </script>
      ```

      CSS:

      ```
      <style>
        <!--
          /* Include CSS style rules here. */

          *{
            color:red
          }
        -->
      </style>
      ```

2. Edit the WebHelp Classic transformation scenario.
3. Go to the **Parameters** tab.
4. Edit the value of the webhelp.head.script parameter and set it to the URL of the XML file created in step 1. Your additional content will be included at the end of the `head` element of your output document.

   Note: If you want to include the content in the `body` element, use the webhelp.body.script parameter instead.
5. Click **OK** to save the changes to the transformation scenario.

Related Information:

Copying Additional Resources to WebHelp Output Directory on page 966

Adding a Button in Code Snippet Areas in DITA WebHelp Classic Output

This task will get you started with how to add an action (such as a button or link) in the code snippet areas that are displayed in WebHelp Classic output created from a *DITA map* transformation. You can then attach your code that does the actual processing for the action.

Follow these steps:

1. Open the *`DITA-OT-DIR`*`\plugins\org.dita.xhtml\xsl\xslhtml\dita2htmlImpl.xsl` file.
2. Locate the `<xsl:template match="*[contains(@class, ' topic/pre ')]" mode="pre-fmt">` template to check the default behavior of this template.
3. Open the *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\desktop\fixup.xsl` file.
4. Create a `<xsl:template match="*[contains(@class, ' topic/pre ')]" mode="pre-fmt">` template to override the default processing.
5. This new template will include your code for creating the button. It will have the action code that does the actual processing attached to it (this can be written in JavaScript, for example).

   Example of a *Select all* button:

```
<xsl:template match="*[contains(@class, ' topic/pre ')]" mode="pre-fmt">
    <button onclick="SelectText(element)">Select all</button>
    <script>
      function SelectText(element) {
        var text = document.getElementById(element);
        var range = document.body.createTextRange();
        range.moveToElementText(text);
        range.select();
      }
    </script>
  </xsl:template>
```

Adding a Favicon in WebHelp Systems

You can add a custom *favicon* to your WebHelp system by simply using a parameter in the transformation scenario to point to your *favicon* image. This is available for DITA and DocBook WebHelp systems using **WebHelp Responsive**, **WebHelp Responsive with Feedback**, **WebHelp Classic**, **WebHelp Classic with Feedback**, or **WebHelp Classic Mobile** transformation scenarios.

To add a *favicon*, follow these steps:

1. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.
2. Locate the `webhelp.favicon` parameter and enter the file path that points to the image that will be use as the *favicon*.
3. Run the transformation scenario.

Adding a Logo Image in the Title Area

To add a logo in the title area of your WebHelp output, follow this procedure:

1. Edit a WebHelp transformation scenario, then open the **Parameters** tab.
2. Specify the path to your logo in the `webhelp.logo.image` parameter.
3. If you also want to add a link to your website when you click the logo image, set the URL in the `webhelp.logo.image.target.url` parameter.
4. Run the transformation scenario.

Adding Video and Audio Objects in DITA WebHelp Output

You can insert references to video and audio media resources (such as videos, audio clips, or embedded HTML frames) in your DITA topics and then publish them to WebHelp output. The media objects can be played directly in all HTML5-based outputs, including WebHelp systems.

To add media objects in the WebHelp output generated from DITA documents, follow the procedures below.

Adding Videos to DITA WebHelp Output

1. Edit the DITA topic and insert a reference to the video through one of the following methods:
   - Use the ***Insert Media Object*** *toolbar action*.
   - Drag (or copy) the video file from your system explorer or the ***Project*** *view* and drop (or paste) it into your document.
   - Manually add an `object` element, as in one of the following examples:

     ```
     <object outputclass="video" type="video/mp4" data="MyVideo.mp4"/>
     ```

     or, instead of the `data` attribute, you can specify the video using a parameter like this:

     ```
     <object outputclass="video">
       <param name="src" value="videos/MyVideo.mp4"/>
     </object>
     ```
2. Apply a **DITA to WebHelp** transformation scenario to obtain the output.

   Result: The transformation converts the `object` element to an HTML5 `video` element.

   ```
   <video controls="controls"><source type="video/mp4" src="MyVideo.mp4"></source>
   </video>
   ```

Adding Audio Clips to DITA WebHelp Output

1. Edit the DITA topic and insert a reference to the audio clip through one of the following methods:
   - Use the ***Insert Media Object*** *toolbar action*.
   - Drag (or copy) the audio file from your system explorer or the ***Project*** *view* and drop (or paste) it into your document.
   - Manually add an `object` element, as in one of the following examples:

     ```
     <object outputclass="audio" type="audio/mpeg" data="MyClip.mp3"/>
     ```

     or, instead of the `data` attribute, you can specify the video using a parameter like this:

     ```
     <object outputclass="audio">
       <param name="src" value="audio/MyClip.mp3"/>
     </object>
     ```
2. Apply a **DITA to WebHelp** transformation scenario to obtain the output.

   Result: The transformation converts the `object` element to an HTML5 `audio` element.

   ```
   <audio controls="controls"><source type="audio/mpeg" src="MyClip.mp3"></source>
   </audio>
   ```

Adding Embedded HTML Frames (such as YouTube videos) to DITA WebHelp Output

1. Edit the DITA topic and insert a reference to the embedded object by using the ***Insert Media Object*** *toolbar action* or by manually adding an `object` element, as in one of the following examples:

   ```
   <object outputclass="iframe" data="https://www.youtube.com/embed/m_vv2s5Trn4"/>
   ```

   or, instead of the `data` attribute, you can specify the object using a parameter like this:

   ```
   <object outputclass="iframe">
     <param name="src" value="http://www.youtube.com/embed/m_vv2s5Trn4"/>
   ```

```
</object>
```

2. Apply a **DITA to WebHelp** transformation scenario to obtain the output.

   Result: The transformation converts the `object` element to an HTML5 `iframe` element.

```
<iframe controls="controls" src="https://www.youtube.com/embed/m_vv2s5Trn4">
</iframe>
```

For more information, see the following video demonstration: *https://www.oxygenxml.com/demo/Media_Objects.html*.

Related Information:

Adding Video, Audio, and Embedded HTML Resources in DITA Topics on page 1626

Adding Videos in DocBook WebHelp Classic Output

You can insert references to videos in your DocBook topics and then publish them to **WebHelp Classic** output. The videos can be played directly in all HTML5-based outputs, including WebHelp systems.

To add videos in the **WebHelp Classic** output generated from DocBook documents, follow these steps:

1. Edit the DocBook document and reference the video using an `mediaobject` element, as in the following example:

```
<mediaobject>
  <videoobject>
    <videodata fileref="http://www.youtube.com/watch/v/VideoName"/>
  </videoobject>
</mediaobject>
```

2. Apply a *WebHelp* or *WebHelp with Feedback* transformation scenario to obtain the output.

Change Numbering Styles for Ordered Lists

Ordered lists (`ol`) are usually numbered in XHTML output using numerals. If you want to change the numbering to alphabetical, follow these steps:

1. Define a custom `outputclass` value and set it as an attribute of the ordered list, as in the following example:

```
<ol outputclass="number-alpha">
    <li>A</li>
    <li>B</li>
    <li>C</li>
</ol>
```

2. Add the following code snippet in a custom CSS file:

```
ol.number-alpha{
    list-style-type:lower-alpha;
}
```

3. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.
   a. For a DITA transformation, set the `args.css` parameter to the path of your custom CSS file. Also, set the `args.copycss` parameter to `yes` to automatically copy your custom CSS in the output folder when the transformation scenario is processed.
   b. For a DocBook transformation, set the `html.stylesheet` parameter to the path of your custom CSS file.
4. Run the transformation scenario.

Changing the Icons in a WebHelp Classic Table of Contents

You can change the icons that appear in a WebHelp Classic table of contents by assigning new image files in a custom CSS file. By default, the icons for the WebHelp Classic table of contents are defined with the following CSS codes (the first example is the icon that appears for a collapsed menu and the second for an expanded menu):

```
.hasSubMenuClosed{
    background: url('../img/book_closed16.png') no-repeat;
    padding-left: 16px;
    cursor: pointer;
}
```

```
.hasSubMenuOpened{
```

```
    background: url('../img/book_opened16.png') no-repeat;
    padding-left: 16px;
    cursor: pointer;
}
```

To assign other icons, use the following procedure:

1. Create a custom CSS file that assigns your desired icons to the `.hasSubMenuClosed` and `.hasSubMenuOpened` properties.

```
.hasSubMenuClosed{
    background: url('TOC-my-closed-button.png') no-repeat;
}
```

```
.hasSubMenuOpened{
    background: url('TOC-my-opened-button.png') no-repeat;
}
```

2. It is recommended that you store the image files in the same directory as the default icons.
   a) For DITA transformations: *DITA-OT-DIR*`\plugins\com.oxygenxml.webhelp\oxygen-webhelp\resources\img\`.
   b) For DocBook transformations: *[OXYGEN_INSTALL_DIR]*`\frameworks\docbook\xsl\com.oxygenxml.webhelp\oxygen-webhelp\resources\img\`.
3. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.
   a) For a DITA transformation, set the `args.css` parameter to the path of your custom CSS file. Also, set the `args.copycss` parameter to `yes` to automatically copy your custom CSS in the output folder when the transformation scenario is processed.
   b) For a DocBook transformation, set the `html.stylesheet` parameter to the path of your custom CSS file.
4. Run the transformation scenario.

CSS Styling to Customize WebHelp Output

One way to customize WebHelp output is to use custom CSS styling. This method can be used to make small, simple styling changes or more advanced, precise changes. To implement the styling in your WebHelp output, you simply need to create the custom CSS file and reference it in your transformation scenario.

As a practical example, to hide the horizontal separator line between the content and footer, follow these steps:

1. Create a custom CSS file that contains the following snippet:

```
.footer_separator {
  display:none;
}
```

2. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.
   a. For a DITA transformation, set the `args.css` parameter to the path of your custom CSS file. Also, set the `args.copycss` parameter to `yes` to automatically copy your custom CSS in the output folder when the transformation scenario is processed.
   b. For a DocBook transformation, set the `html.stylesheet` parameter to the path of your custom CSS file.
3. Run the transformation scenario.

Customize the Appearance of Selected Items in the Table of Contents

The appearance of selected items in the table of contents of WebHelp Classic output can be enhanced.

For example, to highlight the background of the selected item, follow these steps:

1. Locate the `toc.css` file in the following directory:
   a. For DITA transformations: *DITA-OT-DIR*`\plugins\com.oxygenxml.webhelp\oxygen-webhelp\resources\css`.
   b. For DocBook transformations: *[OXYGEN_INSTALL_DIR]*`\frameworks\docbook\xsl\com.oxygenxml.webhelp\oxygen-webhelp\resources\css`.
2. Edit that CSS file, find the `menuItemSelected` class, and change the value of the `background` property.

Note: You can also overwrite the same value from your own custom CSS and then specify the path to your CSS in the transformation scenario (see step 3 in the *Changing the Icons in a WebHelp Classic Table of Contents* topic.

Customizing WebHelp Output with a Custom CSS

By creating your own custom CSS stylesheet, you can customize the look and style of WebHelp output to fit your specific needs.

To use a custom CSS in WebHelp output, follow these steps:

1. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.
   a. For a DITA transformation, set the `args.css` parameter to the path of your custom CSS file. Also, set the `args.copycss` parameter to `yes` to automatically copy your custom CSS in the output folder when the transformation scenario is processed.
   b. For a DocBook transformation, set the `html.stylesheet` parameter to the path of your custom CSS file.
2. Run the transformation scenario.

Disable Caching in WebHelp Classic Output

In cases where a set of WebHelp Classic pages need to be updated on a regular basis to deliver the latest version of the documentation, the WebHelp pages should always be requested from the server upon re-loading it in a Web browser on the client side, rather than re-using an outdated *cached* version in the browser.

To disable caching in WebHelp Classic output, follow this procedure:

1. Edit the following XSL file for DITA or DocBook WebHelp systems:
   - For DITA WebHelp systems, edit the following file: *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\createMainFiles.xsl`.
   - For DocBook WebHelp system, edit the following file: *`[OXYGEN_INSTALL_DIR]`*`\frameworks\docbook\xsl\com.oxygenxml.webhelp\xsl\createMainFiles.xsl`.
2. Locate the following template in the XSL file: `<xsl:template name-"create-toc-common-file">` and add the following code snippet:

```
<meta http-equiv="Pragma" content="no-cache" />
<meta http-equiv="Expires" content="-1" />
```

   Note: The code should look like this:

```
  <html>
    <head>
      <xsl:if test="$withFrames">
      <base target="contentwin"/>
      </xsl:if>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
           <!-- Disable caching of WebHelp pages in web browser. -->
      <meta http-equiv="Pragma" content="no-cache" />
      <meta http-equiv="Expires" content="-1" />
....
```

3. Save your changes to the file.
4. Re-run your WebHelp system transformation scenario.

Editing Scoring Values of Tag Elements in Search Results

The WebHelp **Search** feature is enhanced with a rating mechanism that computes scores for every page that matches the search criteria. HTML tag elements are assigned a scoring value and these values are evaluated for the search results. Oxygen XML Editor includes a properties file that defines the scoring values for tag elements and this file can be edited to customize the values according to your needs.

To edit the scoring values of HTML tag element for enhancing WebHelp search results, follow these steps:

1. Edit the scoring properties file for DITA or DocBook WebHelp systems. The properties file includes instructions and examples to help you with your customization.
   a) For DITA WebHelp systems, edit the following file: *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\indexer\scoring.properties`.
   b) For DocBook WebHelp system, edit the following file: *`[OXYGEN_INSTALL_DIR]`*`\frameworks\docbook\xsl\com.oxygenxml.webhelp\indexer\scoring.properties`.

   The values that can be edited in the `scoring.properties` file:

```
h1 = 10
h2 = 9
```

```
h3 = 8
h4 = 7
h5 = 6
h6 = 5
b = 5
strong = 5
em = 3
i=3
u=3
div.toc=-10
title=20
div.ignore=ignored
meta_keywords = 20
meta_indexterms = 20
meta_description = 25
shortdesc=25
```

2. Save your changes to the file.
3. Re-run your WebHelp system transformation scenario.

Exclude Certain DITA Topics from WebHelp Search Results

The WebHelp **Search** engine does not index DITA topics that have the `@search` attribute set to no. This is useful if you have topics in your *DITA map* structure that you do not want to be included in search results for your WebHelp system.

To exclude DITA topics from WebHelp search results, follow these steps:

1. Edit the *DITA map* and for any `topicref` that you want to exclude from search results, set the `search` attribute to no.

   For example:

   ```
   <topicref href="../topics/internal-topic1.dita" search="no"/>
   ```

2. Save your changes to the *DITA map*.
3. Re-run your WebHelp system transformation scenario.

Flag DITA Content in WebHelp Output

Flagging content in WebHelp output involves defining a set of images that will be used for marking content across your information set.

To flag DITA content, you need to create a filter file that defines properties that will be applied on elements to be flagged. Generally, flagging is supported for *block elements* (such as paragraphs), but not for phrase-level elements within a paragraph. This ensures that the images that will flag the content are easily scanned by the reader, instead of being buried in text.

Follow this procedure:

1. Create a DITA filter file in the directory where you want to add the file. Give the file a descriptive name, such as `audience-flag-build.ditaval`.
2. Define the property of the element you want to be flagged. For example, if you want to flag elements that have the `audience` attribute set to `programmer`, the content of the DITAVAL file should look like the following example:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <val>
     <prop att="audience" val="programmer" action="flag"
    img="D:\resource\delta.gif" alt="sample alt text"/>
   </val>
   ```

   Note that for an element to be flagged, at least one attribute-value pair needs to have a property declared in the DITAVAL file.
3. Specify the DITAVAL file in the **Filters** tab of the transformation scenario.
4. Run the transformation scenario.

Integrating Social Media and Google Tools in WebHelp Output

Oxygen XML Editor includes support for integrating some of the most popular social media sites in WebHelp output.

How to Add a Facebook Like Button in WebHelp Classic Output

To add a Facebook™ *Like* widget to your WebHelp output, follow these steps:

1. Go to the *Facebook Developers* website.
2. Fill-in the displayed form, then click the **Get Code** button.
   A dialog box that contains code snippets is displayed.
3. Copy the two code snippets and paste them into a `<div>` element inside an XML file called `facebook-widget.xml`.

   Make sure you follow these rules:

   - The file must be well-formed.
   - The code for each `script` element must be included in an XML comment.
   - The start and end tags for the XML comment must be on a separate line.

   The content of the XML file should look like this:

```
<div id="facebook">
    <div id="fb-root"/>
    <script>
        <!--
            (function(d, s, id) {
        var js, fjs = d.getElementsByTagName(s)[0];
        if (d.getElementById(id)) return;
        js = d.createElement(s); js.id = id;
        js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
        fjs.parentNode.insertBefore(js, fjs);
        }(document, 'script', 'facebook-jssdk'));
        -->
    </script>
    <div class="fb-like" data-layout="standard" data-action="like"
        data-show-faces="true" data-share="true"/>
</div>
```

4. In Oxygen XML Editor, click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
5. Select an existing WebHelp transformation scenario (depending on your needs, it may be with or without feedback, or the mobile variant) and click the **Duplicate** button to open the **Edit Scenario** dialog box.
6. Switch to the **Parameters** tab and edit the `webhelp.footer.file` parameter to reference the `facebook-widget.xml` file that you created earlier.
7. Click **Ok**.
8. Run the transformation scenario.

Related Information:

DITA Map to WebHelp Output on page 808

How to Add a Google+ Button in WebHelp Classic Output

To add a *Google+* widget to your WebHelp output, follow these steps:

1. Go to the *Google Developers* website.
2. Fill-in the displayed form.
   The preview area on the right side displays the code and a preview of the widget.
3. Copy the code snippet displayed in the preview area and paste it into a `div` element inside an XML file called `google-plus-button.xml`.

   Make sure that the content of the file is well-formed.

   The content of the XML file should look like this:

```
<div id="google-plus">
  <!-- Place this tag in your head or just before your close body tag. -->
  <script src="https://apis.google.com/js/platform.js" async defer></script>

  <!-- Place this tag where you want the +1 button to render. -->
  <div class="g-plusone" data-annotation="inline" data-width="300"></div>
</div>
```

4. In Oxygen XML Editor, click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
5. Select an existing WebHelp transformation scenario (depending on your needs, it may be with or without feedback, or the mobile variant) and click the **Duplicate** button to open the **Edit Scenario** dialog box.
6. Switch to the **Parameters** tab and edit the `webhelp.footer.file` parameter to reference the `google-plus-button.xml` file that you created earlier.
7. Click **Ok**.
8. Run the transformation scenario.

Related Information:

DITA Map to WebHelp Output on page 808

How to Add Tweet Button in WebHelp Classic Output

To add a Twitter™ *Tweet* widget to your WebHelp output, follow these steps:

1. Go to the *Tweet button generator* page.
2. Fill-in the displayed form.
   The **Preview and code** area displays the code.
3. Copy the code snippet displayed in the **Preview and code** area and paste it into a `div` element inside an XML file called `tweet-button.xml`.

   Make sure you follow these rules:

   - The file must be well-formed.
   - The code for each `script` element must be included in an XML comment.
   - The start and end tags for the XML comment must be on a separate line.

   The content of the XML file should look like this:

```
<div id="twitter">
  <a href="https://twitter.com/share" class="twitter-share-button">Tweet</a>
  <script>
    <!--
      !function (d, s, id) {
      var
      js, fjs = d.getElementsByTagName(s)[0], p = /^http:/.test(d.location)
 ? 'http': 'https';
      if (! d.getElementById(id)) {
      js = d.createElement(s);
      js.id = id;
      js.src = p + '://platform.twitter.com/widgets.js';
      fjs.parentNode.insertBefore(js, fjs);
      }
      }
      (document,
      'script', 'twitter-wjs');
     -->
  </script>
</div>
```

4. In Oxygen XML Editor, click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
5. Select an existing WebHelp transformation scenario (depending on your needs, it may be with or without feedback, or the mobile variant) and click the **Duplicate** button to open the **Edit Scenario** dialog box.
6. Switch to the **Parameters** tab and edit the `webhelp.footer.file` parameter to reference the `tweet-button.xml` file that you created earlier.
7. Click **Ok**.
8. Run the transformation scenario.

Related Information:

DITA Map to WebHelp Output on page 808

How to Integrate Google Analytics in WebHelp Classic Output

To allow your WebHelp system to benefit from Google Analytics reports, follow these steps:

1. Create a new Google Analytics account (if you do not already have one) and log on.

2. Choose the Analytics solution that fits the needs of your website.
3. Follow the on-screen instructions to obtain a **Tracking Code** that contains your *Tracking ID*.

   A **Tracking Code** looks like this:

```
<script>
 (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
 (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
 m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
 })(window,document,'script','//www.google-analytics.com/analytics.js','ga');

 ga('create', 'UA-XXXXXXXX-X', 'auto');
 ga('send', 'pageview');
</script>
```

4. Save the Tracking Code (obtained in the previous step) in a new HTML file called `googleAnalytics.html`.
5. In Oxygen XML Editor, click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
6. Select an existing WebHelp transformation scenario (depending on your needs, it may be with or without feedback, or the mobile variant) and click the **Duplicate** button to open the **Edit Scenario** dialog box.
7. Switch to the **Parameters** tab and edit the `webhelp.footer.file` parameter to reference the `googleAnalytics.html` file that you created earlier.
8. Click **Ok**.
9. Run the transformation scenario.

Related Information:

DITA Map to WebHelp Output on page 808

How to Integrate Google Search in WebHelp Classic Output

You can integrate Google Search into your WebHelp output.

To allow your WebHelp system to use Google Search, follow these steps:

1. Go to the Google Custom Search Engine page using your Google account.
2. Press the **Create a custom search engine** button.
3. Follow the on-screen instructions to create a search engine for your site. At the end of this process you should obtain a code snippet.

   A Google Search script looks like this:

```
<script>
 (function()  {
  var cx =
    '000888210889775888983:8mn4x_mf-yg';
  var gcse = document.createElement('script');
  gcse.type = 'text/javascript';
  gcse.async = true;
  gcse.src = (document.location.protocol == 'https:' ?
    'https:' : 'http:') +         '//www.google.com/cse/cse.js?cx=' + cx;
  var s = document.getElementsByTagName('script')[0];
  s.parentNode.insertBefore(gcse, s);
 }
 )();
</script>
```

4. Save the script into a well-formed HTML file called `googlecse.html`.
5. In Oxygen XML Editor, click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
6. Select an existing WebHelp transformation scenario (depending on your needs, it may be with or without feedback, or the mobile variant) and click the **Duplicate** button to open the **Edit Scenario** dialog box.
7. Switch to the **Parameters** tab and edit the `webhelp.google.search.script` parameter to reference the `googlecse.html` file that you created earlier.
8. You can also use the `webhelp.google.search.results` parameter to choose where to display the search results.
   a) Create an HTML file with the following content: `<div class="gcse-searchresults-only" data-queryParameterName="searchQuery" >` (you must use the *HTML5 version for the GCSE*). It is

recommended that you set the `data-linkTarget` attribute value to `frm` for *frameless* versions of the WebHelp system or to `data-contentWin` for *frameset* versions of WebHelp. The default value is `_blank` and if you do not specify a value the search results will be loaded in a new window. For more information about other supported attributes, see *Google Custom Search: Supported Attributes*.

b) Set the value of the `webhelp.google.search.results` parameter to the file path of the file you just created. If this parameter is not specified, the following code is used: `<gcse:searchresults-only linkTarget="frm"></gcse:searchresults-only>`.

9. Click **Ok**.
10. Run the transformation scenario.

Related Information:

Integrating Social Media and Google Tools in WebHelp Output on page 974

Localizing the Email Notifications of WebHelp with Feedback Systems

The feedback-enabled WebHelp systems use emails to notify users when comments are posted. These emails are based on templates stored in the WebHelp directory. The default messages are in English, French, German, and Japanese. These messages are copied into the WebHelp system deployment directory during the execution of the corresponding transformation scenario.

Suppose that you want to localize the emails into Dutch (*nl*). Follow these steps:

DocBook WebHelp Classic with Feedback

1. Create the following directory:

   `[OXYGEN_INSTALL_DIR]\frameworks\docbook\xsl\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\nl`
2. Copy all English template files from `[OXYGEN_INSTALL_DIR]\frameworks\docbook\xsl\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\en` and paste them into the directory you just created.
3. Edit the HTML files from the `[OXYGEN_INSTALL_DIR]\frameworks\docbook\xsl\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\nl` directory and translate the content into Dutch.
4. Start Oxygen XML Editor and edit the **DocBook WebHelp Classic with Feedback** transformation scenario.
5. In the **Parameters** tab, look for the `l10n.gentext.default.language` parameter and set its value to the appropriate language code. In our example, use the value `nl` for Dutch.

   Note: If you set the parameter to a value such as `LanguageCode-CountryCode` (for example, `en-us`), the transformation scenario will only use the language code
6. Run the transformation scenario to obtain the **WebHelp with Feedback** output.

DITA WebHelp Classic with Feedback or WebHelp Responsive with Feedback

1. Create the following directory:

   `DITA-OT-DIR\plugins\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\nl`
2. Copy all English template files from `DITA-OT-DIR\plugins\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\en` and paste them into the directory you just created.
3. Edit the HTML files from the `DITA-OT-DIR\plugins\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\nl` directory and translate the content into Dutch.
4. Start Oxygen XML Editor and edit the **DITA Map WebHelp Classic with Feedback** or **DITA Map WebHelp Responsive with Feedback** transformation scenario.
5. In the **Parameters** tab, look for the `args.default.language` parameter and set its value to the appropriate language code. In our example, use the value `nl` for Dutch.

   Note: If you set the parameter to a value such as `LanguageCode-CountryCode` (for example, `en-us`), the transformation scenario will only use the language code
6. Run the transformation scenario to obtain the **WebHelp with Feedback** output.

Localizing the Interface of WebHelp Output

You can localize the interface of WebHelp output for DITA or DocBook transformations.

Localizing the Interface of WebHelp Output (for DITA Map Transformations)

Static labels that are used in the WebHelp output are kept in translation files in the `DITA-OT-DIR/plugins/com.oxygenxml.webhelp/oxygen-webhelp/resources/localization` folder. Translation files have the *strings-lang1-lang2.xml* name format, where *lang1* and *lang2* are ISO language codes. For example, the US English text is kept in the *strings-en-us.xml* file.

To localize the interface of the WebHelp output for *DITA map* transformations, follow these steps:

1. Look for the *strings-[lang1]-[lang2].xml* file in `DITA-OT-DIR/plugins/com.oxygenxml.webhelp/oxygen-webhelp/resources/localization` directory (for example, the Canadian French file would be: *strings-fr-ca.xml*). If it does not exist, create one starting from the *strings-en-us.xml* file.
2. Translate all the labels from the above language file. Labels are stored in XML elements that have the following format: `<str name="Label name">Caption</str>`.
3. Run the predefined transformation scenario called **Run DITA OT Integrator** by executing it from the **Apply Transformation Scenario(s)** dialog box. If the *integrator* is not visible, select the **Show all scenarios** action that is available in the **Settings** drop-down menu.
4. Make sure that the new XML file that you created in the previous two steps is listed in the file `DITA-OT-DIR/plugins/com.oxygenxml.webhelp/oxygen-webhelp/resources/localization/strings.xml`. In our example for the Canadian French file, it should be listed as: `<lang xml:lang="fr-ca" filename="strings-fr-ca.xml"/>`.
5. Edit any of the **DITA Map to WebHelp** transformation scenarios (with or without feedback, or the mobile version) and set the args.default.language parameter to the code of the language you want to localize (for example, *fr-ca* for Canadian French).
6. Run the transformation scenario to produce the WebHelp output.

Related Information:

Searching Japanese Content in WebHelp Pages on page 978

Localizing the Interface of WebHelp Output (for DocBook Transformations)

Static labels that are used in the WebHelp output are kept in translation files in the `[OXYGEN_INSTALL_DIR]/frameworks/docbook/xsl/com.oxygenxml.webhelp/oxygen-webhelp/resources/localization` folder. Translation files have the *strings-lang1-lang2.xml* name format, where *lang1* and *lang2* are ISO language codes. For example, the US English text is kept in the *strings-en-us.xml* file.

To localize the interface of the WebHelp output for DocBook transformations, follow these steps:

1. Look for the *strings-[lang1]-[lang2].xml* file in `[OXYGEN_INSTALL_DIR]/frameworks/docbook/xsl/com.oxygenxml.webhelp/oxygen-webhelp/resources/localization` directory (for example, the Canadian French file would be: *strings-fr-ca.xml*). If it does not exist, create one starting from the *strings-en-us.xml* file.
2. Translate all the labels from the above language file. Labels are stored in XML elements that have the following format: `<str name="Label name">Caption</str>`.
3. Make sure that the new XML file that you created in the previous two steps is listed in the file `[OXYGEN_INSTALL_DIR]/frameworks/docbook/xsl/com.oxygenxml.webhelp/oxygen-webhelp/resources/localization/strings.xml`. In our example for the Canadian French file, it should be listed as: `<lang xml:lang="fr-ca" filename="strings-fr-ca.xml"/>`.
4. Edit any of the DocBook to WebHelp transformation scenarios (with or without feedback, or the mobile version) and set the l10n.gentext.default.language parameter to the code of the language you want to localize (for example, *fr-ca* for Canadian French).
5. Run the transformation scenario to produce the WebHelp output.

Related Information:

Searching Japanese Content in WebHelp Pages on page 978

Searching Japanese Content in WebHelp Pages

To optimize the indexing of Japanese content in WebHelp pages, the Lucene Kuromoji Japanese analyzer can be used. This analyzer is included in the Oxygen XML Editor installation kit.

Activating Japanese Indexing in DITA WebHelp Systems

To activate the Japanese indexing in your WebHelp system generated from DITA content, follow these steps:

1. Set the language for your content to Japanese with one of the following two methods:
   - Edit a **DITA to WebHelp** transformation scenario and in the ***Parameters*** *tab*, set the value of the `args.default.language` parameter to `ja-jp`.
   - Set the `xml:lang` attribute on the root of the *DITA map* and the referenced topics to `ja-jp`.
2. For the analyzer to work properly, search terms that are entered into the WebHelp search text field must be separated by spaces.
3. Run the **DITA to WebHelp** transformation scenario to generate the output.

Optionally a Japanese user dictionary can be set with the *`webhelp.search.japanese.dictionary` parameter*.

Activating Japanese Indexing in DocBook WebHelp Systems

To activate the Japanese indexing in your WebHelp system generated from DocBook content, follow these steps:

1. Edit a **DocBook to WebHelp** transformation scenario and in the ***Parameters*** *tab*, set the value of the `l10n.gentext.default.language` parameter to `ja`.
2. Run the transformation scenario to generate the output.

Related Information:

Localizing the Interface of WebHelp Output (for DITA Map Transformations) on page 978
Localizing the Interface of WebHelp Output (for DocBook Transformations) on page 1011

Overriding an XSLT Processing Step of the DITA WebHelp Transformation

Since WebHelp output is primarily obtained by running XSLT transformations over the DITA input files (through the **DITA Map WebHelp** transformation scenarios), one customization method would be to override the default XSLT templates that are used by the WebHelp transformations.

There are two methods available to override the XSLT stylesheets implied by the WebHelp transformation.

- The first method is to use one of the ***WebHelp XSLT-import extension points*** that allow you to import an XSLT stylesheet so that it becomes part of the normal build. This method uses the same mechanism as the *DITA-OT XSLT-import extension points*. Note that this method will affect all the outputs generated with the WebHelp system.
- The second method implies that you will *create a DITA-OT extension plugin* with a custom *transtype* and use an Ant build file to define the transformation process. The main difference from the first customization method is that you will not affect all WebHelp transformations. Only the transformations that use the declared plugin *transtype* will be affected.

WebHelp XSLT-Import and XSLT-Parameter Extension Points

The WebHelp XSLT-Import extension points allows you to extend the XSLT stylesheets associated with some of the WebHelp transformation steps. The DITA-OT plug-in installer adds an XSLT import statement in the default WebHelp XSLT so that the XSLT stylesheet referenced by the extension point becomes part of the normal build.

Example:

```
<plugin id="com.oxygenxml.webhelp.extension">
  <feature extension="com.oxygenxml.webhelp.xsl.dita2webhelp" file="xsl/fixup.xsl"/>
</plugin>
```

Attention: The customizations you make by using this extension point will affect all WebHelp transformations. If you want to have a customization that is only available for a certain transformation, please use the *Overriding a WebHelp XSLT Stylesheet from an Ant Build File* method.

XSLT-Import Extension Points

The following extension points are available:

com.oxygenxml.webhelp.xsl.dita2webhelp

Extension point to override the XSLT stylesheet (`dita2webhelpImpl.xsl`) that produces an HTML file for each DITA topic. Depending on the type of WebHelp transformation you are currently using, the path to this stylesheet is as follows:

- ***WebHelp Classic* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\desktop\dita2webhelpImpl.xsl`
- ***WebHelp Classic Mobile* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\mobile\dita2webhelpImpl.xsl`
- ***WebHelp Responsive* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\responsive\dita2webhelpImpl.xsl`

com.oxygenxml.webhelp.xsl.createMainFiles

Extension point to override the XSLT stylesheet (`createMainFilesImpl.xsl`) that produces the WebHelp main HTML page (`index.html`). Depending on the type of WebHelp transformation you are currently using, the path to this stylesheet is as follows:

- ***WebHelp Classic* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\desktop\createMainFilesImpl.xsl`
- ***WebHelp Classic Mobile* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\mobile\createMainFilesImpl.xsl`
- ***WebHelp Responsive* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\responsive\createMainFilesImpl.xsl`

com.oxygenxml.webhelp.xsl.createTocXML

Extension point to override the XSLT stylesheet (`tocDita.xsl`) that produces the `toc.xml` file. This file contains information extracted from the *DITA map* and it is mainly used to construct the WebHelp Table of Contents and navigational links. The path to this stylesheet is: *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\tocDita.xsl`.

XSLT-Parameter Extension Points

If your customization stylesheet declares one or more XSLT parameters and you want to control their values from the transformation scenario, you can use one of the following XSLT parameter extension points:

com.oxygenxml.webhelp.xsl.dita2webhelp.param

Use this extension point to pass parameters to the stylesheet specified using the ***com.oxygenxml.webhelp.xsl.dita2webhelp*** *extension point*.

com.oxygenxml.webhelp.xsl.createMainFiles.param

Use this extension point to pass parameters to the stylesheet specified using the ***com.oxygenxml.webhelp.xsl.createMainFiles*** *extension point*.

com.oxygenxml.webhelp.xsl.createTocXml.param

Use this extension point to pass parameters to the stylesheet specified using the ***com.oxygenxml.webhelp.xsl.createTocXml*** *extension point*.

Related Information:

[DITA-OT] XSLT-Import Extension Points

[DITA-OT] XSLT-Parameter Extension Points

Example: Customizing Side TOC Using XSLT-Import/Parameter Extension Points

This topic provides some common use-cases to demonstrate how to use the WebHelp XSLT-Import extension points to customize the Table of Contents displayed on the right side of the WebHelp output (the default transformation generates a mini table of contents for the current topic and it contains links to the children of current topic, its siblings, and all of its ancestors). The first use-case uses an XSLT-Import extension point while the second uses an XSLT-Parameter extention point.

Use Case 1: WebHelp XSLT-Import extension point to change which topics are displayed in the Side TOC

Suppose you want to customize the side TOC to only include the current topic and its child topics (while excluding its siblings and ancestors).

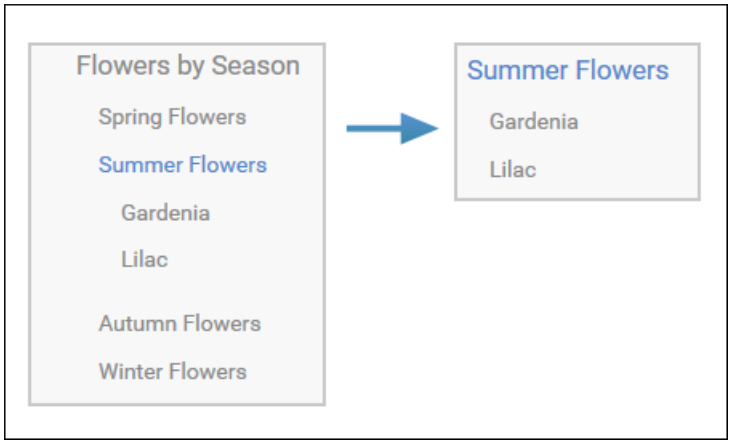


Figure 492: Example: Filtered Side Table of Contents

The default XSLT template responsible for this functionality is defined in: *DITA-OT-DIR*`\plugins\com.oxygenxml.webhelp\xsl\dita\responsive\navigationLinks.xsl`.

```
<xsl:template
          match="
          toc:topic
          [not(@toc = 'no')]
          [not(@processing-role = 'resource-only')]"
          mode="toc-pull" priority="10">
```

This template computes the link for the current topic along with the links for the sibling topics, and then propagates them recursively to the parent topic. For this specific use-case, you need to override this template so that it will produce the link for the current topic along with just the child links that the template already receives.

You can implement this functionality with a WebHelp extension plugin that uses the ***com.oxygenxml.webhelp.xsl.dita2webhelp*** *extension point*. This extension point allows you to specify a customization stylesheet that will override the template described above.

To add this functionality as a DITA-OT plugin, follow these steps:

1. In the *DITA-OT-DIR*`\plugins\` folder, create a folder for this plugin (for example, `com.oxygenxml.webhelp.custom.sidetoc`).
2. Create a **plugin.xml** file (in the folder you created in step 1) that specifies the extension point and your customization stylesheet. For example:

```
<plugin id="com.oxygenxml.webhelp.custom.sidetoc">
  <feature extension="com.oxygenxml.webhelp.xsl.dita2webhelp"
              file="custom_side_TOC.xsl"/>
</plugin>
```

3. Create your customization stylesheet (for example, **custom_side_TOC.xsl**), and edit it to override the template that produces the side TOC:

```
<xsl:template
              match="
              toc:topic
              [not(@toc = 'no')]
              [not(@processing-role = 'resource-only')]"
              mode="toc-pull"
              priority="10">

              <xsl:param name="pathFromMaplist" as="xs:string"/>
              <xsl:param name="children" select="()" as="element()*"/>
              <xsl:param name="parent" select="parent::*" as="element()?"/>

              <xsl:apply-templates select="." mode="sideTocEntry">
              <xsl:with-param name="pathFromMaplist" select="$pathFromMaplist"/>
              <xsl:with-param name="children" select="$children"/>
              </xsl:apply-templates>
</xsl:template>
```

4. Use the **Run DITA OT Integrator** transformation scenario found in the **DITA Map** section in the ***Configure Transformation Scenario(s)*** *dialog box*.
5. Run a **DITA Map WebHelp Responsive** transformation scenario to obtain the customized side TOC.

Use-Case 2: WebHelp XSLT-Parameter extension point to control which topics are displayed in the Side TOC from the transformation scenario

Another possibility to customize what is displayed in the side Table of Contents is to add a transformation parameter that will control the XSLT customization when the transformation scenario is applied. For this use-case, you can use the ***com.oxygenxml.webhelp.xsl.dita2webhelp.param*** *extension point*.

To add this functionality, follow these steps:

1. Create a DITA-OT plugin structure by following the first 3 steps in the *procedure above*.
2. In the customization stylesheet that you created in step 3, declare the `side_toc_only_children` parameter and modify the template to match the topic only when the `side_toc_only_children` parameter is set to *yes*:

```
<xsl:param name="side_toc_only_children" select="'yes'"/>
              ...
              <xsl:template
              match="
              toc:topic
              [not(@toc = 'no')]
              [not(@processing-role = 'resource-only')]
              [$side_toc_only_children = 'yes']"
              ...
```

3. Edit the **plugin.xml** file to specify the **com.oxygenxml.webhelp.xsl.dita2webhelp.param** extension point and a custom parameter file by adding the following line:

```
<feature extension="com.oxygenxml.webhelp.xsl.dita2webhelp.param"
              file="custom_params.xml"/>
```

4. Create a custom parameter file (for example, **custom_params.xml**). It should look like this:

```
<dummy>
              <param name="side_toc_only_children" expression="${side_toc_only_children}"
              if="side_toc_only_children"/>
</dummy>
```

5. Use the **Run DITA OT Integrator** transformation scenario found in the **DITA Map** section in the ***Configure Transformation Scenario(s)*** *dialog box*.
6. Edit a **DITA Map WebHelp Responsive** transformation scenario and in the ***Parameters*** *tab*, specify the desired value (*yes* or *no*) for your custom parameter (`side_toc_only_children`).
7. Run the transformation scenario.

Related Information:

[DITA-OT] XSLT-Import Extension Points

[DITA-OT] XSLT-Parameter Extension Points

Overriding a WebHelp XSLT Stylesheet from an Ant Build File

To create a WebHelp XSLT customization that is only available for a certain DITA OT transformation, the extension plugin should *declare a custom transtype*. The WebHelp XSLT stylesheets can be overridden from an ANT file provided by the DITA-OT extension plugin. From the Ant target associated with the plugin, you will specify a custom XSLT stylesheet that imports the original WebHelp stylesheet and add some customization templates.

The following procedure explains how to create a DITA-OT extension plugin that uses this extension method:

1. In the *`DITA-OT-DIR`*`\plugins\` folder, create a folder for this plugin (for example, `com.oxygenxml.webhelp.responsive.custom`).
2. Create a **plugin.xml** file (in the folder you created in step 1) that specifies a new DITA-OT `transtype` and the build file associated with the plugin. For example:

```
<plugin id="com.oxygenxml.webhelp.responsive.customization">
    <feature extension="dita.conductor.target.relative" file="integrator.xml"/>
    <transtype name="webhelp-responsive-custom" extends="webhelp-responsive"
      desc="WebHelp Responsive Customization"/>
</plugin>
```

3. Create the **integrator.xml** file that will import the actual plugin Ant build file.

```
<project basedir="." name="Webhelp Responsive Customization">
    <import file="build.xml"/>
</project>
```

4. Create the **build.xml** file that overrides the value of properties associated with the *XSLT stylesheets used to produce HTML files*. The following Ant properties can be overridden to specify your customization stylesheets:

   `args.wh.xsl`

   Specify this property if you want to customize the *XSLT stylesheet used to produce an HTML file for each topic*.

   `args.create.main.files.xsl`

   Specify this property if you want to customize the *XSLT stylesheet used to produce the main HTML file*.

   `args.createTocXML.xsl`

   Specify this property if you want to customize the *XSLT stylesheet used to produce the `toc.xml` file*. The toc.xml file contains information extracted from DITA map and it is mainly used to create the WebHelp TOC.

   For example, to customize a WebHelp Responsive transformation type, the build file should look like:

```
<project basedir="." name="Webhelp Responsive Customization">
 <target name="dita2webhelp-responsive-custom">
   <!--
     Override this property if you want to customize the XSLT stylesheet
     used to produce an HTML file for each topic
   -->
   <property
     name="args.wh.xsl"
     value="${dita.plugin.com.oxygenxml.webhelp.responsive.custom.dir}
                                                /xsl/dita2webhelpCustom.xsl"/>
   <!--
     Override this property if you want to customize the XSLT stylesheet
     used to produce the main HTML file.
   -->
   <property
     name="args.create.main.files.xsl"
     value="${dita.plugin.com.oxygenxml.webhelp.responsive.custom.dir}
                                            /xsl/createMainFilesCustom.xsl"/>
   <!--
     Override this property if you want to customize the XSLT stylesheet
     used to produce the toc.xml file.
   -->
   <property
     name="args.createTocXML.xsl"
     value="${dita.plugin.com.oxygenxml.webhelp.responsive.custom.dir}
                                            /xsl/createTocXMLCustom.xsl"/>
   <!--
     Depending on which version of WebHelp you want to customize,
     you need to delegate to different build targets:
     * dita2webhelp-responsive - when you are customizing the Webhelp Responsive
     * dita2webhelp-mobile - when you are customizing the Webhelp Mobile
     * dita2webhelp - when you are customizing the Webhelp Classic
   -->
   <antcall target="dita2webhelp-responsive"/>
 </target>
</project>
```

   Note: Depending on which version of WebHelp you want to customize, you need to call one of the following build targets:

   - **dita2webhelp-responsive** - For WebHelp Responsive (with or without feedback) output.
   - **dita2webhelp** - For WebHelp Classic (with or without feedback) output.
   - **dita2webhelp-mobile** - For WebHelp Classic Mobile output (deprecated).

5. Create an **xsl** directory in the plugin customization directory (that you created in step 1) to store the customized XSLT stylesheets.

Referencing the WebHelp XSLT Stylesheets from Your Customizations

Note that your customization stylesheets should import the original stylesheets that they override. To reference the WebHelp stylesheets, you can use the **plugin:com.oxygenxml.dita-ot.plugin.webhelp** prefix. This prefix is rewritten by an XML catalog to the WebHelp root directory.

Customizing the `dita2webhelp.xsl` Stylesheet

The XSLT stylesheet that customizes the WebHelp **dita2webhelp.xsl** stylesheet should look like:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:math="http://www.w3.org/2005/xpath-functions/math"
   exclude-result-prefixes="xs math"
   version="2.0">
   <!--
     Import the original stylesheet used to produce an HTML file for each topic.
   -->
   <xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/
                                          dita/responsive/dita2webhelp.xsl"/>
   <!--
     Please add your customization templates here.
   -->
</xsl:stylesheet>
```

Depending on which version of WebHelp you want to customize, you need to import one of the following XSLT stylesheets:

- **dita2webhelp-responsive** (WebHelp Responsive) -

  `<xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/responsive/dita2webhelp.xsl"/>`
- **dita2webhelp** (WebHelp Classic) -

  `<xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/desktop/dita2webhelp.xsl"/>`
- **dita2webhelp-mobile** (WebHelp Classic Mobile) -

  `<xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/mobile/dita2webhelp.xsl"/>`

Customizing the `createMainFiles.xsl` Stylesheet

The XSLT stylesheet that customizes the WebHelp **createMainFiles.xsl** stylesheet should look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:math="http://www.w3.org/2005/xpath-functions/math"
    exclude-result-prefixes="xs math"
    version="2.0">
    <!--
        Import the original stylesheet used to produce the main HTML file.
    -->
    <xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/
                                            responsive/createMainFiles.xsl"/>
    <!--
        Please add your customization templates here.
    -->

</xsl:stylesheet>
```

Depending on which version of WebHelp you want to customize, you need to import one of the following XSLT stylesheets:

- **dita2webhelp-responsive** (WebHelp Responsive) -

  `<xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/responsive/createMainFiles.xsl"/>`
- **dita2webhelp** (WebHelp Classic) -

  `<xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/desktop/createMainFiles.xsl"/>`
- **dita2webhelp-mobile** (WebHelp Classic Mobile) -

  `<xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/mobile/createMainFiles.xsl"/>`

Customizing the `tocDita.xsl` File

The XSLT stylesheet that customizes the WebHelp **tocDita.xsl** stylesheet should look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
    xmlns:math="http://www.w3.org/2005/xpath-functions/math"
    exclude-result-prefixes="xs math"
    version="2.0">
    <!--
        Import the original stylesheet used to produce the toc.xml file.
    -->
    <xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/tocDita.xsl"/>
    <!--
        Please add your customization templates here.
    -->

</xsl:stylesheet>
```

Removing the *Previous*/*Next* Links from WebHelp Classic Output

The **Previous** and **Next** links that are created at the top area of each WebHelp Classic page can be hidden with a CSS code.

To remove these links from WebHelp Classic output, follow these steps:

1. Add the following CSS code in a custom CSS stylesheet:

```
.navparent, .navprev, .navnext {
    visibility:hidden;
}
```

2. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.
   a. For a DITA transformation, set the `args.css` parameter to the path of your custom CSS file. Also, set the `args.copycss` parameter to `yes` to automatically copy your custom CSS in the output folder when the transformation scenario is processed.
   b. For a DocBook transformation, set the `html.stylesheet` parameter to the path of your custom CSS file.
3. Run the transformation scenario.

Search Engine Optimization for DITA WebHelp

A **DITA Map WebHelp** transformation scenario can be configured to produce a `sitemap.xml` file that is used by search engines to aid crawling and indexing mechanisms. A *sitemap* lists all pages of a WebHelp system and allows webmasters to provide additional information about each page, such as the date it was last updated, change frequency, and importance of each page in relation to other pages in your WebHelp deployment.

The structure of the `sitemap.xml` file looks like this:

```
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.example.com/topics/introduction.html</loc>
    <lastmod>2014-10-24</lastmod>
    <changefreq>weekly</changefreq>
    <priority>0.5</priority>
  </url>
  <url>
    <loc>http://www.example.com/topics/care.html#care</loc>
    <lastmod>2014-10-24</lastmod>
    <changefreq>weekly</changefreq>
    <priority>0.5</priority>
  </url>
    .  .  .
</urlset>
```

Each page has a `<url>` element structure containing additional information, such as:

- `loc` - the URL of the page. This URL must begin with the protocol (such as `http`), if required by your web server. It is constructed from the value of the `webhelp.sitemap.base.url` parameter from the transformation scenario and the relative path to the page (collected from the `href` attribute of a `topicref` element in the *DITA map*).

  Note: The value must have fewer than 2,048 characters.
- `lastmod` (optional) - the date when the page was last modified. The date format is `YYYY-MM-DD`.
- `changefreq` (optional) - indicates how frequently the page is likely to change. This value provides general information to assist search engines, but may not correlate exactly to how often they crawl the page. Valid values are: `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, and `never`. The first time the `sitemap.xml` file is generated, the value is set based upon the value of the `webhelp.sitemap.change.frequency` parameter in the DITA WebHelp transformation scenario. You can change the value in each `url` element by editing the `sitemap.xml` file.

Note: The value `always` should be used to describe documents that change each time they are accessed. The value `never` should be used to describe archived URLs.

- `priority` (optional) - the priority of this page relative to other pages on your site. Valid values range from 0.0 to 1.0. This value does not affect how your pages are compared to pages on other sites. It only lets the search engines know which pages you deem most important for the crawlers. The first time the `sitemap.xml` file is generated, the value is set based upon the value of the `webhelp.sitemap.priority` parameter in the DITA WebHelp transformation scenario. You can change the value in each `url` element by editing the `sitemap.xml` file.

Creating and Editing the `sitemap.xml` File

Follow these steps to produce a `sitemap.xml` file for your WebHelp system, which can then be edited to fine-tune search engine optimization:

1. **Edit** the transformation scenario you currently use for obtaining your WebHelp output. This opens the **Edit DITA Scenario** dialog box.
2. Open the **Parameters** tab and set a value for the following parameters:
   - `webhelp.sitemap.base.url` - the URL of the location where your WebHelp system is deployed

     Note: This parameter is required for Oxygen XML Editor to generate the `sitemap.xml` file.
   - `webhelp.sitemap.change.frequency` - how frequently the WebHelp pages are likely to change (accepted values are: `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, and `never`)
   - `webhelp.sitemap.priority` - the priority of each page (value ranging from 0.0 to 1.0)
3. Run the transformation scenario.
4. Look for the `sitemap.xml` file in the transformation's output folder. Edit the file to fine-tune the parameters of each page, according to your needs.

Support for Right-to-Left (RTL) Oriented Languages for DITA WebHelp

To activate support for RTL (right-to-left) languages in WebHelp output, edit the *DITA map* and set the `xml:lang` attribute on its root element (`map`). The corresponding attribute value can be set for following RTL languages:

- `ar-eg` - Arabic
- `he-il` - Hebrew
- `ur-pk` - Urdu

WebHelp Skin Builder

The **WebHelp Skin Builder** is a simple, easy-to-use tool, specially designed to assist users to visually customize the look and feel of the WebHelp output. It is implemented as an online tool hosted on the Oxygen XML Editor website and allows you to experiment with various styles and colors over a documentation sample.

To be able to use the **Skin Builder**, you need:

- An Internet connection and unrestricted access to Oxygen XML Editor website.
- A late version web browser.

To start the **Skin Builder**, do one of the following:

- For DocBook or DITA WebHelp systems, use a web browser to go to *https://www.oxygenxml.com/webhelp-skin-builder*.
- For DITA WebHelp systems, you can click the **Online preview** link in the ***Skins tab*** of a DITA OT transformation scenario. In the upper section of the preview, click the **Select Skin** button, then choose **Customize Skin**.

Skin Builder Layout

The left side panel of the **Skin Builder** is divided into 3 sections:

- **Actions** - Contains the following two buttons:
  - **Import** - Opens an **Import CSS** dialog box that allows you to load a CSS stylesheet and apply it over the documentation sample.
  - **Export** - Saves all properties as a CSS file.

- **Settings** - Includes a **Highlight selection** option that helps you identify the areas affected by a particular element customization.
  - When hovering an item in the customizable elements menu, the affected sample area is highlighted with a dotted blue border.
  - When an item in the customizable elements menu is selected, the affected sample area is highlighted with a solid red border.
- **Customize** - Provides a series of customizable elements organized under four main categories:
  - Header
  - TOC Area
  - Vertical Splitter
  - Content

  For each customizable element, you can alter properties such as background color or font face. Any alteration made in the customizable elements menu is applied in real time over the sample area.

Create a Customization Skin

1. The starting point can be either one of the predefined skins or a CSS stylesheet applied over the sample using the **Import** button.
2. Use the elements in the **Customize** section to set properties that modify the look of the skin. By default, all customizable elements display a single property, but you can make more visible by clicking the **+Add** button and choosing from the available properties.

   Note: If you want to revert a particular property to its initial value, press the **Reset** button.
3. When you are happy with the skin customizations you have made, press the **Export** button. All settings will be saved in a CSS file.

Apply a Customization Skin to a DITA Map to WebHelp Classic Transformation Scenario

1. Start Oxygen XML Editor.
2. Load the *DITA map* you want to produce as a WebHelp output.
3. Edit a **DITA Map to WebHelp**-type transformation scenario. Set the previously exported CSS file in the **Custom** section of the **Skins** tab.
4. Run the transformation to obtain the WebHelp output.

Apply a Customization Skin to a DocBook to WebHelp Classic Transformation Scenario

1. Start Oxygen XML Editor.
2. Load the DocBook file you want to produce as a WebHelp output.
3. In the **Parameters** tab, set the `webhelp.skin.css` parameter to point to the previously exported CSS.
4. To customize the logo, use the following parameters: webhelp.logo.image and webhelp.logo.image.target.url.
5. Run the transformation to obtain the WebHelp output.

To see our video demonstration about using the WebHelp Skin Builder, go to *https://www.oxygenxml.com/demo/Skin_Builder.html*.

Related Information:

Skins Tab (DITA OT Transformations) on page 1682

WebHelp Classic Runtime Additional Parameters

A deployed WebHelp system can accept the following GET parameters:

- `log` - The value can be `true` or `false` (default value). When set to `true`, it enables JavaScript debugging.
- `contextId` - The WebHelp JavaScript engine will look for this value in the `context-help-map.xml` mapping file and load the corresponding help page. For more information, see the *Context-Sensitive WebHelp System* topic.

Note: You can use an *anchor* in the `contextId` parameter to jump to a specific section in a document. For example, `contextId=topicID#anchor`.

- `appname` - You can use this parameter in conjunction with `contextId` to search for this value in the corresponding *appname attribute value* in the mapping file.

```
http://localhost/webhelp/index.html?contextId=topicID&appname=myApplication
```

- `toc.visible` - The value can be `true` (default value) or `false`. When set to `false`, the table of contents will be collapsed when you load the WebHelp page.
- `searchQuery` - You can use this parameter to perform a search operation when WebHelp is loaded. For example, if you want to open WebHelp showing all search results for *growing flowers*, the URL should look like this: `http://localhost/webhelp/index.html?searchQuery=growing%20flowers`.

Related Information:

Context-Sensitive WebHelp Classic System on page 1021

Context-Sensitive WebHelp Classic System

Context-sensitive help systems assist users by providing specific informational topics for certain components of a user interface, such as a button or window. This mechanism works based on mappings between a unique ID defined in the topic and a corresponding HTML page.

Generating Context-Sensitive Help

When WebHelp Classic output is generated by Oxygen XML Editor, the transformation process produces an XML mapping file called `context-help-map.xml` and copies it in the output folder of the transformation. This XML file maps an ID to a corresponding HTML page through an `appContext` element, as in the following example:

```
<map productID="oxy-webhelp" productVersion="1.1">
  <appContext helpID="myapp-functionid1" path="tasks/app-help1.html"/>
  <appContext helpID="myapp-functionid2" path="tasks/app-help1.html"/>
    .  .  .
</map>
```

The possible attributes are as follows:

`helpID`

A Unique ID provided by a topic from two possible sources (`resourceid` element or `id` attribute):

`resourceid`

The `resourceid` element is mapped into the `appContext` element and can be specified in either the `topicref` within a *DITA map* or in a `prolog` within a DITA topic. The `resourceid` element accepts the following attributes:

- **`appname`** - A name for the external application that references the topic. If this attribute is not specified, its value is considered to be empty (" ").
- **`appid`** - An ID used by an application to identify the topic.
- **`id`** - Specifies a value that is used by a specific application to identify the topic, but this attribute is ignored if an **`appid`** attribute is used.

Note: Multiple `appid` values can be associated with a single `appname` value (and multiple `appname` values can be associated with a single `appid` value), but the values for both attributes work in combination to specify a specific ID for a specific application, and therefore each combination of values for the `appid` and `appname` attributes should be unique within the context of a single *root map*. For example, suppose that you need two different functions of an application to both open the same WebHelp page.

Example: `resourceid` Specified in a DITA Map

The `resourceid` element can be specified in a `topicmeta` element within a `topicref`.

```
<map title="App Help">
  <topicref href="app-help1.dita" type="task">
     <topicmeta>
       <resourceid appname="myapp" appid="functionid1"/>
       <resourceid appname="myapp" appid="functionid2"/>
     </topicmeta>
```

```
  </topicref>
</map>
```

Example: `resourceid` Specified in a DITA Topic

The `resourceid` element can be specified in a `prolog` element within a DITA topic.

```
<task id="app-help1">
  <title>My App Help</title>
  <prolog>
    <resourceid appname="myapp" appid="functionid1"/>
    <resourceid appname="myapp" appid="functionid2"/>
  </prolog>
...
</task>
```

For more information about the `resourceid` element, see *DITA Specifications: <resourceid>*.

`id`

If a `resourceid` element is not declared in the *DITA map* or DITA topic (as described above), the `id` attribute that is set on the topic root element is mapped into the `appContext` element.

Important: You should ensure that these defined IDs are unique in the context of the entire DITA project. If the IDs are not unique, the transformation scenario will display warning messages in the transformation console output and the help system will not work properly.

`path`

The path to a corresponding WebHelp page. This path is relative to the location of the `context-help-map.xml` mapping file.

`productID` (Applicable only if you are using the WebHelp Classic with Feedback transformation scenario)

The ID of the product for your documentation project.

`productVersion` (Applicable only if you are using the WebHelp Classic with Feedback transformation scenario)

The version of the product for your documentation project.

There are two ways of implementing context-sensitive help in your system:

- The XML mapping file can be loaded by a PHP script on the server side. The script receives the `contextId` value and will look it up in the XML file.
- Invoke one of the WebHelp system files `index.html` or `index_frames.html` and pass the `contextId` parameter with a specific value. The WebHelp system will automatically open the help page associated with the value of the `contextId` parameter.

The following example will open a *frameless* version of the WebHelp system showing the page associated with the ID `dialog1ID`:

```
index.html?contextId=dialog1ID
```

The following example will open a *frameset* version of the WebHelp system showing the page associated with the ID `view1ID`:

```
index_frames.html?contextId=view1ID
```

Tip: You can use an *anchor* in the `contextId` parameter to jump to a specific section in a document. For example, `contextId=topicID#anchor`.

Context-Sensitive Queries

You can use the URL field in your browser to search for topics in a context-sensitive WebHelp system with the assistance of the following parameters:

- `contextId` - The WebHelp JavaScript engine will look for this value in the `context-help-map.xml` mapping file and load the corresponding help page. For more information, see the *Context-Sensitive WebHelp System* topic.

  Note: You can use an *anchor* in the `contextId` parameter to jump to a specific section in a document. For example, `contextId=topicID#anchor`.

- `appname` - You can use this parameter in conjunction with `contextId` to search for this value in the corresponding *appname attribute value* in the mapping file.

```
http://localhost/webhelp/index.html?contextId=topicID&appname=myApplication
```

Related Information:

WebHelp Classic Runtime Additional Parameters on page 1020

Publishing WebHelp Classic Output on a SharePoint Site

Since WebHelp output must be published locally, on the same machine where the WebHelp process is running, to publish your files directly to a SharePoint library you need to map a network drive to connect to SharePoint and change your file extensions to `.aspx`, as described in the steps below.

To publish WebHelp Classic output on a SharePoint site, follow this procedure:

1. Map a network drive to connect to your SharePoint library. For more information, see: *https://support.microsoft.com/en-us/kb/2616712*.
2. To allow browsers to open your published files (rather than downloading them), you need to change the file extensions from `.html` to `.aspx`. Fortunately, this can be done in the transformation scenario.
   a. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.
      a. For a DITA transformation, set the `args.outext` parameter to `.aspx`.
      b. For a DocBook transformation, set the `html.ext` parameter to `.aspx`.
   b. Run the transformation scenario.

WebHelp Classic Mobile System (Deprecated)

Note: The WebHelp Classic Mobile system was deprecated as of version 19.0. It is recommended that you use the more recent ***WebHelp Responsive*** *system* for a flexible mobile variant.

WebHelp is a form of online help that consists of a series of web pages (XHTML format). Its advantages include platform independence, ability to update content continuously, and it can be viewed using a regular web browser. The Oxygen XML Editor WebHelp system includes several variants to suit your specific needs. The **WebHelp Classic Mobile** variant works on multiple platforms (Android, iOS, BlackBerry, Windows Mobile) and is specially designed for mobile devices when feedback from users is not necessary. It is available for DocBook and DITA document types. The functionality of the desktop WebHelp Classic layout is preserved, it is organized in an intuitive layout, and offers table of contents, search capabilities, and index navigation.

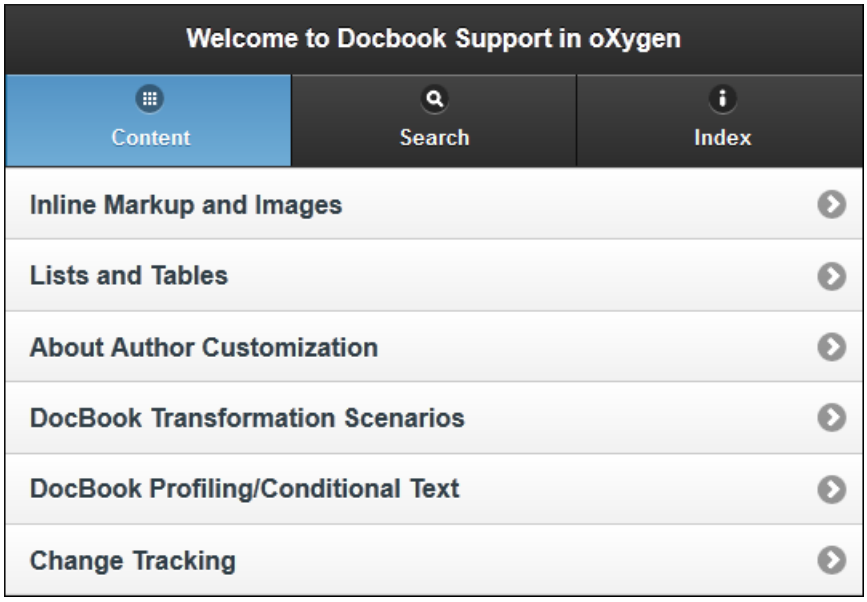


Figure 493: WebHelp Classic Mobile

Important: Due to some security restrictions in certain browsers (Google Chrome and Internet Explorer), WebHelp Classic pages loaded from the local system (through URLs of the `file:///...` format) may not work properly. We recommend that you load WebHelp Classic pages in Google Chrome or Internet Explorer only from a web server (with a URL such as `http://your.server.com/webhelp/index.html` or `http://localhost/web_pages/index.html`).

Customizing WebHelp Classic Mobile Systems

If you are familiar with CSS and coding, you can style your WebHelp output through your own custom stylesheets. You can also customize your output by modifying option and parameters in the transformation scenario. This section includes topics that explain various ways to customize your WebHelp Classic Mobile system output.

Copying Additional Resources to WebHelp Output Directory

To copy additional resources (such as JavaScript, CSS or other resources) to the output directory of a WebHelp system, follow these steps:

1. Place all your resources in the same directory.
2. Edit the WebHelp transformation scenario.
3. Go to the **Parameters** tab.
4. Edit the value for the webhelp.custom.resources parameter and set it to the absolute path of the directory in step 1.
5. Click **OK** to save the changes to the transformation scenario.
   All files from the new directory will be copied to the root of the WebHelp output directory.

Adding Custom HTML Content in WebHelp Classic Output

You can add custom HTML content in the WebHelp Classic output by inserting it in a well-formed XML file that will be referenced in the transformation scenario. This content may include references to additional JavaScript, CSS, and other types of resources, or such resources can be inserted inline within the HTML content that is inserted in the XML file.

To include custom HTML content in the WebHelp Classic output files, follow these steps:

1. Insert the HTML content in a well-formed XML file. There are several things to consider in regards to this XML file:

   a. **Well-Formedness** - If the file is not a *Well-formed XML document* (or fragments are not well-formed), the transformation will fail.

      A common use case is if you want to include several `script` or `link` elements. In this case, the XML fragment has multiple root elements and to make it well-formed, you can wrap it in an `html` element. This element tag will be filtered out and only its children will be copied to the output documents. Similarly, you can wrap your content in `head`, `body`, `html/head`, or `html/body` elements.

   b. **Referencing Resources in the XML File** - You can include references to local resources (such as JavaScript or CSS files) by using the predefined `${oxygen-webhelp-output-dir}` macro to specify their paths relative to the output directory:

      ```
      <html>
        <script type="text/javascript" src="${oxygen-webhelp-output-dir}/js/test.js"/>
        <link rel="stylesheet" type="text/css"
              href="${oxygen-webhelp-output-dir}/css/test.css" />
      </html>
      ```

      To copy the referenced resources to the output directory, follow the procedure in: *Copying Additional Resources to WebHelp Output Directory* on page 966.

   c. **Inline JavaScript or CSS Content** - If you want to include inline JavaScript or CSS content in the XML file, it is important to place this content inside an XML comment, as in the following examples:

      JavaScript:

      ```
      <script type="text/javascript">
        <!--
          /* Include JavaScript code here. */

          function myFunction() {
            return true;
          }
      ```

```
  -->
</script>
```

CSS:

```
<style>
  <!--
    /* Include CSS style rules here. */

    *{
      color:red
    }
  -->
</style>
```

2. Edit the WebHelp Classic transformation scenario.
3. Go to the **Parameters** tab.
4. Edit the value of the webhelp.head.script parameter and set it to the URL of the XML file created in step 1. Your additional content will be included at the end of the `head` element of your output document.

   Note: If you want to include the content in the `body` element, use the webhelp.body.script parameter instead.
5. Click **OK** to save the changes to the transformation scenario.

Related Information:

Copying Additional Resources to WebHelp Output Directory on page 966

Adding a Button in Code Snippet Areas in DITA WebHelp Classic Output

This task will get you started with how to add an action (such as a button or link) in the code snippet areas that are displayed in WebHelp Classic output created from a *DITA map* transformation. You can then attach your code that does the actual processing for the action.

Follow these steps:

1. Open the *`DITA-OT-DIR`*`\plugins\org.dita.xhtml\xsl\xslhtml\dita2htmlImpl.xsl` file.
2. Locate the `<xsl:template match="*[contains(@class, ' topic/pre ')]" mode="pre-fmt">` template to check the default behavior of this template.
3. Open the *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\desktop\fixup.xsl` file.
4. Create a `<xsl:template match="*[contains(@class, ' topic/pre ')]" mode="pre-fmt">` template to override the default processing.
5. This new template will include your code for creating the button. It will have the action code that does the actual processing attached to it (this can be written in JavaScript, for example).

   Example of a *Select all* button:

```
<xsl:template match="*[contains(@class, ' topic/pre ')]" mode="pre-fmt">
    <button onclick="SelectText(element)">Select all</button>
    <script>
      function SelectText(element) {
        var text = document.getElementById(element);
        var range = document.body.createTextRange();
        range.moveToElementText(text);
        range.select();
      }
    </script>
  </xsl:template>
```

Adding a Favicon in WebHelp Systems

You can add a custom *favicon* to your WebHelp system by simply using a parameter in the transformation scenario to point to your *favicon* image. This is available for DITA and DocBook WebHelp systems using **WebHelp Responsive**, **WebHelp Responsive with Feedback**, **WebHelp Classic**, **WebHelp Classic with Feedback**, or **WebHelp Classic Mobile** transformation scenarios.

To add a *favicon*, follow these steps:

1. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.
2. Locate the `webhelp.favicon` parameter and enter the file path that points to the image that will be use as the *favicon*.
3. Run the transformation scenario.

Adding Video and Audio Objects in DITA WebHelp Output

You can insert references to video and audio media resources (such as videos, audio clips, or embedded HTML frames) in your DITA topics and then publish them to WebHelp output. The media objects can be played directly in all HTML5-based outputs, including WebHelp systems.

To add media objects in the WebHelp output generated from DITA documents, follow the procedures below.

Adding Videos to DITA WebHelp Output

1. Edit the DITA topic and insert a reference to the video through one of the following methods:
   - Use the ***Insert Media Object*** *toolbar action*.
   - Drag (or copy) the video file from your system explorer or the ***Project*** *view* and drop (or paste) it into your document.
   - Manually add an `object` element, as in one of the following examples:

     ```
     <object outputclass="video" type="video/mp4" data="MyVideo.mp4"/>
     ```

     or, instead of the `data` attribute, you can specify the video using a parameter like this:

     ```
     <object outputclass="video">
       <param name="src" value="videos/MyVideo.mp4"/>
     </object>
     ```
2. Apply a **DITA to WebHelp** transformation scenario to obtain the output.

   Result: The transformation converts the `object` element to an HTML5 `video` element.

   ```
   <video controls="controls"><source type="video/mp4" src="MyVideo.mp4"></source>
   </video>
   ```

Adding Audio Clips to DITA WebHelp Output

1. Edit the DITA topic and insert a reference to the audio clip through one of the following methods:
   - Use the ***Insert Media Object*** *toolbar action*.
   - Drag (or copy) the audio file from your system explorer or the ***Project*** *view* and drop (or paste) it into your document.
   - Manually add an `object` element, as in one of the following examples:

     ```
     <object outputclass="audio" type="audio/mpeg" data="MyClip.mp3"/>
     ```

     or, instead of the `data` attribute, you can specify the video using a parameter like this:

     ```
     <object outputclass="audio">
       <param name="src" value="audio/MyClip.mp3"/>
     </object>
     ```
2. Apply a **DITA to WebHelp** transformation scenario to obtain the output.

   Result: The transformation converts the `object` element to an HTML5 `audio` element.

   ```
   <audio controls="controls"><source type="audio/mpeg" src="MyClip.mp3"></source>
   </audio>
   ```

Adding Embedded HTML Frames (such as YouTube videos) to DITA WebHelp Output

1. Edit the DITA topic and insert a reference to the embedded object by using the ***Insert Media Object*** *toolbar action* or by manually adding an `object` element, as in one of the following examples:

   ```
   <object outputclass="iframe" data="https://www.youtube.com/embed/m_vv2s5Trn4"/>
   ```

   or, instead of the `data` attribute, you can specify the object using a parameter like this:

   ```
   <object outputclass="iframe">
     <param name="src" value="http://www.youtube.com/embed/m_vv2s5Trn4"/>
   ```

```
</object>
```

2. Apply a **DITA to WebHelp** transformation scenario to obtain the output.

   Result: The transformation converts the `object` element to an HTML5 `iframe` element.

```
<iframe controls="controls" src="https://www.youtube.com/embed/m_vv2s5Trn4">
</iframe>
```

For more information, see the following video demonstration: *https://www.oxygenxml.com/demo/Media_Objects.html*.

Related Information:

Adding Video, Audio, and Embedded HTML Resources in DITA Topics on page 1626

Adding Videos in DocBook WebHelp Classic Output

You can insert references to videos in your DocBook topics and then publish them to **WebHelp Classic** output. The videos can be played directly in all HTML5-based outputs, including WebHelp systems.

To add videos in the **WebHelp Classic** output generated from DocBook documents, follow these steps:

1. Edit the DocBook document and reference the video using an `mediaobject` element, as in the following example:

```
<mediaobject>
  <videoobject>
    <videodata fileref="http://www.youtube.com/watch/v/VideoName"/>
  </videoobject>
</mediaobject>
```

2. Apply a *WebHelp* or *WebHelp with Feedback* transformation scenario to obtain the output.

Changing the Style of WebHelp Mobile Pages

You can change the style for your WebHelp Mobile pages by setting a custom theme created with a third-party tool.

To create a custom theme for WebHelp Mobile pages, use the following procedure:

1. Create a custom theme (the result will be a CSS stylesheet). Use a designer tool, such as the *ThemeRoller for jQuery Mobile*, to create your own custom theme and then download the resulting CSS stylesheet.

   Tip: If you are using ThemeRoller for jQuery Mobile, make sure you use a type C swatch.
2. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.
   a. For a DITA transformation, set the `args.css` parameter to the path of your custom CSS file. Also, set the `args.copycss` parameter to `yes` to automatically copy your custom CSS in the output folder when the transformation scenario is processed.
   b. For a DocBook transformation, set the `html.stylesheet` parameter to the path of your custom CSS file.
3. Make sure that the output folder is empty.
4. Run the transformation scenario.

Change Numbering Styles for Ordered Lists

Ordered lists (`ol`) are usually numbered in XHTML output using numerals. If you want to change the numbering to alphabetical, follow these steps:

1. Define a custom `outputclass` value and set it as an attribute of the ordered list, as in the following example:

```
<ol outputclass="number-alpha">
    <li>A</li>
    <li>B</li>
    <li>C</li>
</ol>
```

2. Add the following code snippet in a custom CSS file:

```
ol.number-alpha{
    list-style-type:lower-alpha;
}
```

3. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.

a. For a DITA transformation, set the `args.css` parameter to the path of your custom CSS file. Also, set the `args.copycss` parameter to `yes` to automatically copy your custom CSS in the output folder when the transformation scenario is processed.
b. For a DocBook transformation, set the `html.stylesheet` parameter to the path of your custom CSS file.

4. Run the transformation scenario.

Changing the Icons in a WebHelp Classic Table of Contents

You can change the icons that appear in a WebHelp Classic table of contents by assigning new image files in a custom CSS file. By default, the icons for the WebHelp Classic table of contents are defined with the following CSS codes (the first example is the icon that appears for a collapsed menu and the second for an expanded menu):

```
.hasSubMenuClosed{
    background: url('../img/book_closed16.png') no-repeat;
    padding-left: 16px;
    cursor: pointer;
}
```

```
.hasSubMenuOpened{
    background: url('../img/book_opened16.png') no-repeat;
    padding-left: 16px;
    cursor: pointer;
}
```

To assign other icons, use the following procedure:

1. Create a custom CSS file that assigns your desired icons to the `.hasSubMenuClosed` and `.hasSubMenuOpened` properties.

```
.hasSubMenuClosed{
    background: url('TOC-my-closed-button.png') no-repeat;
}
```

```
.hasSubMenuOpened{
    background: url('TOC-my-opened-button.png') no-repeat;
}
```

2. It is recommended that you store the image files in the same directory as the default icons.
   a) For DITA transformations: *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\oxygen-webhelp\resources\img\`.
   b) For DocBook transformations: *`[OXYGEN_INSTALL_DIR]`*`\frameworks\docbook\xsl\com.oxygenxml.webhelp\oxygen-webhelp\resources\img\`.
3. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.
   a) For a DITA transformation, set the `args.css` parameter to the path of your custom CSS file. Also, set the `args.copycss` parameter to `yes` to automatically copy your custom CSS in the output folder when the transformation scenario is processed.
   b) For a DocBook transformation, set the `html.stylesheet` parameter to the path of your custom CSS file.
4. Run the transformation scenario.

CSS Styling to Customize WebHelp Output

One way to customize WebHelp output is to use custom CSS styling. This method can be used to make small, simple styling changes or more advanced, precise changes. To implement the styling in your WebHelp output, you simply need to create the custom CSS file and reference it in your transformation scenario.

As a practical example, to hide the horizontal separator line between the content and footer, follow these steps:

1. Create a custom CSS file that contains the following snippet:

```
.footer_separator {
  display:none;
}
```

2. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.

a. For a DITA transformation, set the `args.css` parameter to the path of your custom CSS file. Also, set the `args.copycss` parameter to `yes` to automatically copy your custom CSS in the output folder when the transformation scenario is processed.
b. For a DocBook transformation, set the `html.stylesheet` parameter to the path of your custom CSS file.

3. Run the transformation scenario.

Customize the Appearance of Selected Items in the Table of Contents

The appearance of selected items in the table of contents of WebHelp Classic output can be enhanced.

For example, to highlight the background of the selected item, follow these steps:

1. Locate the `toc.css` file in the following directory:
   a. For DITA transformations: *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\oxygen-webhelp\resources\css`.
   b. For DocBook transformations: *`[OXYGEN_INSTALL_DIR]`*`\frameworks\docbook\xsl\com.oxygenxml.webhelp\oxygen-webhelp\resources\css`.
2. Edit that CSS file, find the `menuItemSelected` class, and change the value of the `background` property.

Note: You can also overwrite the same value from your own custom CSS and then specify the path to your CSS in the transformation scenario (see step 3 in the *Changing the Icons in a WebHelp Classic Table of Contents* topic.

Customizing WebHelp Output with a Custom CSS

By creating your own custom CSS stylesheet, you can customize the look and style of WebHelp output to fit your specific needs.

To use a custom CSS in WebHelp output, follow these steps:

1. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.
   a. For a DITA transformation, set the `args.css` parameter to the path of your custom CSS file. Also, set the `args.copycss` parameter to `yes` to automatically copy your custom CSS in the output folder when the transformation scenario is processed.
   b. For a DocBook transformation, set the `html.stylesheet` parameter to the path of your custom CSS file.
2. Run the transformation scenario.

Disable Caching in WebHelp Classic Output

In cases where a set of WebHelp Classic pages need to be updated on a regular basis to deliver the latest version of the documentation, the WebHelp pages should always be requested from the server upon re-loading it in a Web browser on the client side, rather than re-using an outdated *cached* version in the browser.

To disable caching in WebHelp Classic output, follow this procedure:

1. Edit the following XSL file for DITA or DocBook WebHelp systems:
   - For DITA WebHelp systems, edit the following file: *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\createMainFiles.xsl`.
   - For DocBook WebHelp system, edit the following file: *`[OXYGEN_INSTALL_DIR]`*`\frameworks\docbook\xsl\com.oxygenxml.webhelp\xsl\createMainFiles.xsl`.
2. Locate the following template in the XSL file: `<xsl:template name-"create-toc-common-file">` and add the following code snippet:

```
<meta http-equiv="Pragma" content="no-cache" />
<meta http-equiv="Expires" content="-1" />
```

Note: The code should look like this:

```
  <html>
    <head>
      <xsl:if test="$withFrames">
      <base target="contentwin"/>
      </xsl:if>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
           <!-- Disable caching of WebHelp pages in web browser. -->
      <meta http-equiv="Pragma" content="no-cache" />
      <meta http-equiv="Expires" content="-1" />
```

```
....
```

3. Save your changes to the file.
4. Re-run your WebHelp system transformation scenario.

Editing Scoring Values of Tag Elements in Search Results

The WebHelp **Search** feature is enhanced with a rating mechanism that computes scores for every page that matches the search criteria. HTML tag elements are assigned a scoring value and these values are evaluated for the search results. Oxygen XML Editor includes a properties file that defines the scoring values for tag elements and this file can be edited to customize the values according to your needs.

To edit the scoring values of HTML tag element for enhancing WebHelp search results, follow these steps:

1. Edit the scoring properties file for DITA or DocBook WebHelp systems. The properties file includes instructions and examples to help you with your customization.
   a) For DITA WebHelp systems, edit the following file: *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\indexer\scoring.properties`.
   b) For DocBook WebHelp system, edit the following file: *`[OXYGEN_INSTALL_DIR]`*`\frameworks\docbook\xsl\com.oxygenxml.webhelp\indexer\scoring.properties`.

   The values that can be edited in the `scoring.properties` file:

```
h1 = 10
h2 = 9
h3 = 8
h4 = 7
h5 = 6
h6 = 5
b = 5
strong = 5
em = 3
i=3
u=3
div.toc=-10
title=20
div.ignore=ignored
meta_keywords = 20
meta_indexterms = 20
meta_description = 25
shortdesc=25
```

2. Save your changes to the file.
3. Re-run your WebHelp system transformation scenario.

Exclude Certain DITA Topics from WebHelp Search Results

The WebHelp **Search** engine does not index DITA topics that have the `@search` attribute set to no. This is useful if you have topics in your *DITA map* structure that you do not want to be included in search results for your WebHelp system.

To exclude DITA topics from WebHelp search results, follow these steps:

1. Edit the *DITA map* and for any `topicref` that you want to exclude from search results, set the `search` attribute to no.

   For example:

```
<topicref href="../topics/internal-topic1.dita" search="no"/>
```

2. Save your changes to the *DITA map*.
3. Re-run your WebHelp system transformation scenario.

Flag DITA Content in WebHelp Output

Flagging content in WebHelp output involves defining a set of images that will be used for marking content across your information set.

To flag DITA content, you need to create a filter file that defines properties that will be applied on elements to be flagged. Generally, flagging is supported for *block elements* (such as paragraphs), but not for phrase-level elements within a paragraph. This ensures that the images that will flag the content are easily scanned by the reader, instead of being buried in text.

Follow this procedure:

1. Create a DITA filter file in the directory where you want to add the file. Give the file a descriptive name, such as `audience-flag-build.ditaval`.
2. Define the property of the element you want to be flagged. For example, if you want to flag elements that have the `audience` attribute set to `programmer`, the content of the DITAVAL file should look like the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<val>
  <prop att="audience" val="programmer" action="flag"
 img="D:\resource\delta.gif" alt="sample alt text"/>
</val>
```

   Note that for an element to be flagged, at least one attribute-value pair needs to have a property declared in the DITAVAL file.
3. Specify the DITAVAL file in the **Filters** tab of the transformation scenario.
4. Run the transformation scenario.

Integrating Social Media and Google Tools in WebHelp Output

Oxygen XML Editor includes support for integrating some of the most popular social media sites in WebHelp output.

How to Add a Facebook Like Button in WebHelp Classic Output

To add a Facebook™ *Like* widget to your WebHelp output, follow these steps:

1. Go to the *Facebook Developers* website.
2. Fill-in the displayed form, then click the **Get Code** button.
   A dialog box that contains code snippets is displayed.
3. Copy the two code snippets and paste them into a `<div>` element inside an XML file called `facebook-widget.xml`.

   Make sure you follow these rules:

   - The file must be well-formed.
   - The code for each `script` element must be included in an XML comment.
   - The start and end tags for the XML comment must be on a separate line.

   The content of the XML file should look like this:

```
<div id="facebook">
    <div id="fb-root"/>
    <script>
        <!--
            (function(d, s, id) {
        var js, fjs = d.getElementsByTagName(s)[0];
        if (d.getElementById(id)) return;
        js = d.createElement(s); js.id = id;
        js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
        fjs.parentNode.insertBefore(js, fjs);
        }(document, 'script', 'facebook-jssdk'));
        -->
    </script>
    <div class="fb-like" data-layout="standard" data-action="like"
        data-show-faces="true" data-share="true"/>
</div>
```

4. In Oxygen XML Editor, click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
5. Select an existing WebHelp transformation scenario (depending on your needs, it may be with or without feedback, or the mobile variant) and click the **Duplicate** button to open the **Edit Scenario** dialog box.
6. Switch to the **Parameters** tab and edit the `webhelp.footer.file` parameter to reference the `facebook-widget.xml` file that you created earlier.
7. Click **Ok**.
8. Run the transformation scenario.

Related Information:

DITA Map to WebHelp Output on page 808

How to Add a Google+ Button in WebHelp Classic Output

To add a *Google+* widget to your WebHelp output, follow these steps:

1. Go to the *Google Developers* website.
2. Fill-in the displayed form.
   The preview area on the right side displays the code and a preview of the widget.
3. Copy the code snippet displayed in the preview area and paste it into a `div` element inside an XML file called `google-plus-button.xml`.

   Make sure that the content of the file is well-formed.

   The content of the XML file should look like this:

```
<div id="google-plus">
  <!-- Place this tag in your head or just before your close body tag. -->
  <script src="https://apis.google.com/js/platform.js" async defer></script>

  <!-- Place this tag where you want the +1 button to render. -->
  <div class="g-plusone" data-annotation="inline" data-width="300"></div>
</div>
```

4. In Oxygen XML Editor, click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
5. Select an existing WebHelp transformation scenario (depending on your needs, it may be with or without feedback, or the mobile variant) and click the **Duplicate** button to open the **Edit Scenario** dialog box.
6. Switch to the **Parameters** tab and edit the `webhelp.footer.file` parameter to reference the `google-plus-button.xml` file that you created earlier.
7. Click **Ok**.
8. Run the transformation scenario.

Related Information:

DITA Map to WebHelp Output on page 808

How to Add Tweet Button in WebHelp Classic Output

To add a Twitter™ *Tweet* widget to your WebHelp output, follow these steps:

1. Go to the *Tweet button generator* page.
2. Fill-in the displayed form.
   The **Preview and code** area displays the code.
3. Copy the code snippet displayed in the **Preview and code** area and paste it into a `div` element inside an XML file called `tweet-button.xml`.

   Make sure you follow these rules:

   - The file must be well-formed.
   - The code for each `script` element must be included in an XML comment.
   - The start and end tags for the XML comment must be on a separate line.

   The content of the XML file should look like this:

```
<div id="twitter">
  <a href="https://twitter.com/share" class="twitter-share-button">Tweet</a>
  <script>
    <!--
      !function (d, s, id) {
      var
      js, fjs = d.getElementsByTagName(s)[0], p = /^http:/.test(d.location)
 ? 'http': 'https';
      if (! d.getElementById(id)) {
      js = d.createElement(s);
      js.id = id;
      js.src = p + '://platform.twitter.com/widgets.js';
      fjs.parentNode.insertBefore(js, fjs);
      }
      }
      (document,
      'script', 'twitter-wjs');
     -->
  </script>
```

```
</div>
```

4. In Oxygen XML Editor, click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
5. Select an existing WebHelp transformation scenario (depending on your needs, it may be with or without feedback, or the mobile variant) and click the **Duplicate** button to open the **Edit Scenario** dialog box.
6. Switch to the **Parameters** tab and edit the `webhelp.footer.file` parameter to reference the `tweet-button.xml` file that you created earlier.
7. Click **Ok**.
8. Run the transformation scenario.

Related Information:

DITA Map to WebHelp Output on page 808

How to Integrate Google Analytics in WebHelp Classic Output

To allow your WebHelp system to benefit from Google Analytics reports, follow these steps:

1. Create a new Google Analytics account (if you do not already have one) and log on.
2. Choose the Analytics solution that fits the needs of your website.
3. Follow the on-screen instructions to obtain a **Tracking Code** that contains your *Tracking ID*.

   A **Tracking Code** looks like this:

```
<script>
 (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
 (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
 m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
 })(window,document,'script','//www.google-analytics.com/analytics.js','ga');

 ga('create', 'UA-XXXXXXXX-X', 'auto');
 ga('send', 'pageview');
</script>
```

4. Save the Tracking Code (obtained in the previous step) in a new HTML file called `googleAnalytics.html`.
5. In Oxygen XML Editor, click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
6. Select an existing WebHelp transformation scenario (depending on your needs, it may be with or without feedback, or the mobile variant) and click the **Duplicate** button to open the **Edit Scenario** dialog box.
7. Switch to the **Parameters** tab and edit the `webhelp.footer.file` parameter to reference the `googleAnalytics.html` file that you created earlier.
8. Click **Ok**.
9. Run the transformation scenario.

Related Information:

DITA Map to WebHelp Output on page 808

How to Integrate Google Search in WebHelp Classic Output

You can integrate Google Search into your WebHelp output.

To allow your WebHelp system to use Google Search, follow these steps:

1. Go to the Google Custom Search Engine page using your Google account.
2. Press the **Create a custom search engine** button.
3. Follow the on-screen instructions to create a search engine for your site. At the end of this process you should obtain a code snippet.

   A Google Search script looks like this:

```
<script>
 (function()  {
  var cx =
    '000888210889775888983:8mn4x_mf-yg';
  var gcse = document.createElement('script');
  gcse.type = 'text/javascript';
  gcse.async = true;
```

```
  gcse.src = (document.location.protocol == 'https:' ?
    'https:' : 'http:') +         '//www.google.com/cse/cse.js?cx=' + cx;
  var s = document.getElementsByTagName('script')[0];
  s.parentNode.insertBefore(gcse, s);
 }
 )();
</script>
```

4. Save the script into a well-formed HTML file called `googlecse.html`.
5. In Oxygen XML Editor, click the **Configure Transformation Scenario(s)** action from the toolbar (or the **Document** > **Transformation** menu.
6. Select an existing WebHelp transformation scenario (depending on your needs, it may be with or without feedback, or the mobile variant) and click the **Duplicate** button to open the **Edit Scenario** dialog box.
7. Switch to the **Parameters** tab and edit the `webhelp.google.search.script` parameter to reference the `googlecse.html` file that you created earlier.
8. You can also use the `webhelp.google.search.results` parameter to choose where to display the search results.
   a) Create an HTML file with the following content: `<div class="gcse-searchresults-only" data-queryParameterName="searchQuery" >` (you must use the *HTML5 version for the GCSE*). It is recommended that you set the `data-linkTarget` attribute value to `frm` for *frameless* versions of the WebHelp system or to `data-contentWin` for *frameset* versions of WebHelp. The default value is `_blank` and if you do not specify a value the search results will be loaded in a new window. For more information about other supported attributes, see *Google Custom Search: Supported Attributes*.
   b) Set the value of the `webhelp.google.search.results` parameter to the file path of the file you just created. If this parameter is not specified, the following code is used: `<gcse:searchresults-only linkTarget="frm"></gcse:searchresults-only>`.
9. Click **Ok**.
10. Run the transformation scenario.

Related Information:

Integrating Social Media and Google Tools in WebHelp Output on page 974

Localizing the Interface of WebHelp Output

You can localize the interface of WebHelp output for DITA or DocBook transformations.

Localizing the Interface of WebHelp Output (for DITA Map Transformations)

Static labels that are used in the WebHelp output are kept in translation files in the *`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp/oxygen-webhelp/resources/localization` folder. Translation files have the *strings-lang1-lang2.xml* name format, where *lang1* and *lang2* are ISO language codes. For example, the US English text is kept in the *strings-en-us.xml* file.

To localize the interface of the WebHelp output for *DITA map* transformations, follow these steps:

1. Look for the *strings-[lang1]-[lang2].xml* file in *`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp/oxygen-webhelp/resources/localization` directory (for example, the Canadian French file would be: *strings-fr-ca.xml*). If it does not exist, create one starting from the *strings-en-us.xml* file.
2. Translate all the labels from the above language file. Labels are stored in XML elements that have the following format: `<str name="Label name">Caption</str>`.
3. Run the predefined transformation scenario called **Run DITA OT Integrator** by executing it from the **Apply Transformation Scenario(s)** dialog box. If the *integrator* is not visible, select the **Show all scenarios** action that is available in the **Settings** drop-down menu.
4. Make sure that the new XML file that you created in the previous two steps is listed in the file *`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp/oxygen-webhelp/resources/localization/strings.xml`. In our example for the Canadian French file, it should be listed as: `<lang xml:lang="fr-ca" filename="strings-fr-ca.xml"/>`.
5. Edit any of the **DITA Map to WebHelp** transformation scenarios (with or without feedback, or the mobile version) and set the args.default.language parameter to the code of the language you want to localize (for example, *fr-ca* for Canadian French).

6. Run the transformation scenario to produce the WebHelp output.

Related Information:

Searching Japanese Content in WebHelp Pages on page 978

Localizing the Interface of WebHelp Output (for DocBook Transformations)

Static labels that are used in the WebHelp output are kept in translation files in the `[OXYGEN_INSTALL_DIR]/frameworks/docbook/xsl/com.oxygenxml.webhelp/oxygen-webhelp/resources/localization` folder. Translation files have the *strings-lang1-lang2.xml* name format, where *lang1* and *lang2* are ISO language codes. For example, the US English text is kept in the *strings-en-us.xml* file.

To localize the interface of the WebHelp output for DocBook transformations, follow these steps:

1. Look for the *strings-[lang1]-[lang2].xml* file in `[OXYGEN_INSTALL_DIR]/frameworks/docbook/xsl/com.oxygenxml.webhelp/oxygen-webhelp/resources/localization` directory (for example, the Canadian French file would be: *strings-fr-ca.xml*). If it does not exist, create one starting from the *strings-en-us.xml* file.
2. Translate all the labels from the above language file. Labels are stored in XML elements that have the following format: `<str name="Label name">Caption</str>`.
3. Make sure that the new XML file that you created in the previous two steps is listed in the file `[OXYGEN_INSTALL_DIR]/frameworks/docbook/xsl/com.oxygenxml.webhelp/oxygen-webhelp/resources/localization/strings.xml`. In our example for the Canadian French file, it should be listed as: `<lang xml:lang="fr-ca" filename="strings-fr-ca.xml"/>`.
4. Edit any of the DocBook to WebHelp transformation scenarios (with or without feedback, or the mobile version) and set the l10n.gentext.default.language parameter to the code of the language you want to localize (for example, *fr-ca* for Canadian French).
5. Run the transformation scenario to produce the WebHelp output.

Related Information:

Searching Japanese Content in WebHelp Pages on page 978

Searching Japanese Content in WebHelp Pages

To optimize the indexing of Japanese content in WebHelp pages, the Lucene Kuromoji Japanese analyzer can be used. This analyzer is included in the Oxygen XML Editor installation kit.

Activating Japanese Indexing in DITA WebHelp Systems

To activate the Japanese indexing in your WebHelp system generated from DITA content, follow these steps:

1. Set the language for your content to Japanese with one of the following two methods:
   - Edit a **DITA to WebHelp** transformation scenario and in the ***Parameters*** *tab*, set the value of the `args.default.language` parameter to `ja-jp`.
   - Set the `xml:lang` attribute on the root of the *DITA map* and the referenced topics to `ja-jp`.
2. For the analyzer to work properly, search terms that are entered into the WebHelp search text field must be separated by spaces.
3. Run the **DITA to WebHelp** transformation scenario to generate the output.

Optionally a Japanese user dictionary can be set with the *`webhelp.search.japanese.dictionary` parameter*.

Activating Japanese Indexing in DocBook WebHelp Systems

To activate the Japanese indexing in your WebHelp system generated from DocBook content, follow these steps:

1. Edit a **DocBook to WebHelp** transformation scenario and in the ***Parameters*** *tab*, set the value of the `l10n.gentext.default.language` parameter to `ja`.
2. Run the transformation scenario to generate the output.

Related Information:

Localizing the Interface of WebHelp Output (for DITA Map Transformations) on page 978

Localizing the Interface of WebHelp Output (for DocBook Transformations) on page 1011

Overriding an XSLT Processing Step of the DITA WebHelp Transformation

Since WebHelp output is primarily obtained by running XSLT transformations over the DITA input files (through the **DITA Map WebHelp** transformation scenarios), one customization method would be to override the default XSLT templates that are used by the WebHelp transformations.

There are two methods available to override the XSLT stylesheets implied by the WebHelp transformation.

- The first method is to use one of the ***WebHelp XSLT-import extension points*** that allow you to import an XSLT stylesheet so that it becomes part of the normal build. This method uses the same mechanism as the *DITA-OT XSLT-import extension points*. Note that this method will affect all the outputs generated with the WebHelp system.
- The second method implies that you will *create a DITA-OT extension plugin* with a custom *transtype* and use an Ant build file to define the transformation process. The main difference from the first customization method is that you will not affect all WebHelp transformations. Only the transformations that use the declared plugin *transtype* will be affected.

WebHelp XSLT-Import and XSLT-Parameter Extension Points

The WebHelp XSLT-Import extension points allows you to extend the XSLT stylesheets associated with some of the WebHelp transformation steps. The DITA-OT plug-in installer adds an XSLT import statement in the default WebHelp XSLT so that the XSLT stylesheet referenced by the extension point becomes part of the normal build.

Example:

```
<plugin id="com.oxygenxml.webhelp.extension">
  <feature extension="com.oxygenxml.webhelp.xsl.dita2webhelp" file="xsl/fixup.xsl"/>
</plugin>
```

Attention: The customizations you make by using this extension point will affect all WebHelp transformations. If you want to have a customization that is only available for a certain transformation, please use the *Overriding a WebHelp XSLT Stylesheet from an Ant Build File* method.

XSLT-Import Extension Points

The following extension points are available:

com.oxygenxml.webhelp.xsl.dita2webhelp

Extension point to override the XSLT stylesheet (`dita2webhelpImpl.xsl`) that produces an HTML file for each DITA topic. Depending on the type of WebHelp transformation you are currently using, the path to this stylesheet is as follows:

- ***WebHelp Classic* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\desktop\dita2webhelpImpl.xsl`
- ***WebHelp Classic Mobile* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\mobile\dita2webhelpImpl.xsl`
- ***WebHelp Responsive* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\responsive\dita2webhelpImpl.xsl`

com.oxygenxml.webhelp.xsl.createMainFiles

Extension point to override the XSLT stylesheet (`createMainFilesImpl.xsl`) that produces the WebHelp main HTML page (`index.html`). Depending on the type of WebHelp transformation you are currently using, the path to this stylesheet is as follows:

- ***WebHelp Classic* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\desktop\createMainFilesImpl.xsl`
- ***WebHelp Classic Mobile* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\mobile\createMainFilesImpl.xsl`
- ***WebHelp Responsive* Transformations** - *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\responsive\createMainFilesImpl.xsl`

com.oxygenxml.webhelp.xsl.createTocXML

Extension point to override the XSLT stylesheet (`tocDita.xsl`) that produces the `toc.xml` file. This file contains information extracted from the *DITA map* and it is mainly used to construct the WebHelp Table of Contents and navigational links. The path to this stylesheet is: *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\tocDita.xsl`.

XSLT-Parameter Extension Points

If your customization stylesheet declares one or more XSLT parameters and you want to control their values from the transformation scenario, you can use one of the following XSLT parameter extension points:

com.oxygenxml.webhelp.xsl.dita2webhelp.param

Use this extension point to pass parameters to the stylesheet specified using the ***com.oxygenxml.webhelp.xsl.dita2webhelp*** *extension point*.

com.oxygenxml.webhelp.xsl.createMainFiles.param

Use this extension point to pass parameters to the stylesheet specified using the ***com.oxygenxml.webhelp.xsl.createMainFiles*** *extension point*.

com.oxygenxml.webhelp.xsl.createTocXml.param

Use this extension point to pass parameters to the stylesheet specified using the ***com.oxygenxml.webhelp.xsl.createTocXml*** *extension point*.

Related Information:

[DITA-OT] XSLT-Import Extension Points

[DITA-OT] XSLT-Parameter Extension Points

Example: Customizing Side TOC Using XSLT-Import/Parameter Extension Points

This topic provides some common use-cases to demonstrate how to use the WebHelp XSLT-Import extension points to customize the Table of Contents displayed on the right side of the WebHelp output (the default transformation generates a mini table of contents for the current topic and it contains links to the children of current topic, its siblings, and all of its ancestors). The first use-case uses an XSLT-Import extension point while the second uses an XSLT-Parameter extention point.

Use Case 1: WebHelp XSLT-Import extension point to change which topics are displayed in the Side TOC

Suppose you want to customize the side TOC to only include the current topic and its child topics (while excluding its siblings and ancestors).

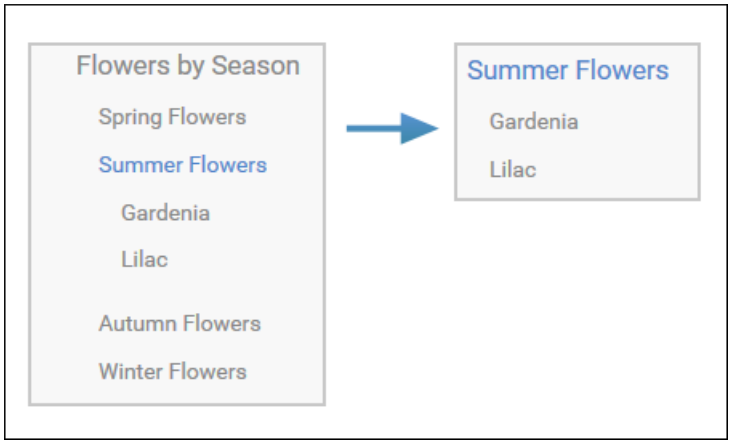


Figure 494: Example: Filtered Side Table of Contents

The default XSLT template responsible for this functionality is defined in: *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp\xsl\dita\responsive\navigationLinks.xsl`.

```
<xsl:template
          match="
          toc:topic
```

```
          [not(@toc = 'no')]
          [not(@processing-role = 'resource-only')]"
          mode="toc-pull" priority="10">
```

This template computes the link for the current topic along with the links for the sibling topics, and then propagates them recursively to the parent topic. For this specific use-case, you need to override this template so that it will produce the link for the current topic along with just the child links that the template already receives.

You can implement this functionality with a WebHelp extension plugin that uses the ***com.oxygenxml.webhelp.xsl.dita2webhelp*** *extension point*. This extension point allows you to specify a customization stylesheet that will override the template described above.

To add this functionality as a DITA-OT plugin, follow these steps:

1. In the *`DITA-OT-DIR`*`\plugins\` folder, create a folder for this plugin (for example, `com.oxygenxml.webhelp.custom.sidetoc`).
2. Create a **plugin.xml** file (in the folder you created in step 1) that specifies the extension point and your customization stylesheet. For example:

```
<plugin id="com.oxygenxml.webhelp.custom.sidetoc">
  <feature extension="com.oxygenxml.webhelp.xsl.dita2webhelp"
             file="custom_side_TOC.xsl"/>
</plugin>
```

3. Create your customization stylesheet (for example, **custom_side_TOC.xsl**), and edit it to override the template that produces the side TOC:

```
<xsl:template
              match="
              toc:topic
              [not(@toc = 'no')]
              [not(@processing-role = 'resource-only')]"
              mode="toc-pull"
              priority="10">

              <xsl:param name="pathFromMaplist" as="xs:string"/>
              <xsl:param name="children" select="()" as="element()*"/>
              <xsl:param name="parent" select="parent::*" as="element()?"/>

              <xsl:apply-templates select="." mode="sideTocEntry">
              <xsl:with-param name="pathFromMaplist" select="$pathFromMaplist"/>
              <xsl:with-param name="children" select="$children"/>
              </xsl:apply-templates>
</xsl:template>
```

4. Use the **Run DITA OT Integrator** transformation scenario found in the **DITA Map** section in the ***Configure Transformation Scenario(s)*** *dialog box*.
5. Run a **DITA Map WebHelp Responsive** transformation scenario to obtain the customized side TOC.

Use-Case 2: WebHelp XSLT-Parameter extension point to control which topics are displayed in the Side TOC from the transformation scenario

Another possibility to customize what is displayed in the side Table of Contents is to add a transformation parameter that will control the XSLT customization when the transformation scenario is applied. For this use-case, you can use the ***com.oxygenxml.webhelp.xsl.dita2webhelp.param*** *extension point*.

To add this functionality, follow these steps:

1. Create a DITA-OT plugin structure by following the first 3 steps in the *procedure above*.
2. In the customization stylesheet that you created in step 3, declare the `side_toc_only_children` parameter and modify the template to match the topic only when the `side_toc_only_children` parameter is set to *yes*:

```
<xsl:param name="side_toc_only_children" select="'yes'"/>
              ...
              <xsl:template
              match="
              toc:topic
              [not(@toc = 'no')]
              [not(@processing-role = 'resource-only')]
              [$side_toc_only_children = 'yes']"
              ...
```

3. Edit the **plugin.xml** file to specify the **com.oxygenxml.webhelp.xsl.dita2webhelp.param** extension point and a custom parameter file by adding the following line:

```
<feature extension="com.oxygenxml.webhelp.xsl.dita2webhelp.param"
              file="custom_params.xml"/>
```

4. Create a custom parameter file (for example, **custom_params.xml**). It should look like this:

```
<dummy>
              <param name="side_toc_only_children" expression="${side_toc_only_children}"
              if="side_toc_only_children"/>
</dummy>
```

5. Use the **Run DITA OT Integrator** transformation scenario found in the **DITA Map** section in the ***Configure Transformation Scenario(s)*** *dialog box*.
6. Edit a **DITA Map WebHelp Responsive** transformation scenario and in the ***Parameters*** *tab*, specify the desired value (*yes* or *no*) for your custom parameter (`side_toc_only_children`).
7. Run the transformation scenario.

Related Information:

[DITA-OT] XSLT-Import Extension Points

[DITA-OT] XSLT-Parameter Extension Points

Overriding a WebHelp XSLT Stylesheet from an Ant Build File

To create a WebHelp XSLT customization that is only available for a certain DITA OT transformation, the extension plugin should *declare a custom transtype*. The WebHelp XSLT stylesheets can be overridden from an ANT file provided by the DITA-OT extension plugin. From the Ant target associated with the plugin, you will specify a custom XSLT stylesheet that imports the original WebHelp stylesheet and add some customization templates.

The following procedure explains how to create a DITA-OT extension plugin that uses this extension method:

1. In the *`DITA-OT-DIR`*`\plugins\` folder, create a folder for this plugin (for example, `com.oxygenxml.webhelp.responsive.custom`).
2. Create a **plugin.xml** file (in the folder you created in step 1) that specifies a new DITA-OT `transtype` and the build file associated with the plugin. For example:

```
<plugin id="com.oxygenxml.webhelp.responsive.customization">
    <feature extension="dita.conductor.target.relative" file="integrator.xml"/>
    <transtype name="webhelp-responsive-custom" extends="webhelp-responsive"
      desc="WebHelp Responsive Customization"/>
</plugin>
```

3. Create the **integrator.xml** file that will import the actual plugin Ant build file.

```
<project basedir="." name="Webhelp Responsive Customization">
    <import file="build.xml"/>
</project>
```

4. Create the **build.xml** file that overrides the value of properties associated with the *XSLT stylesheets used to produce HTML files*. The following Ant properties can be overridden to specify your customization stylesheets:

   `args.wh.xsl`

   Specify this property if you want to customize the *XSLT stylesheet used to produce an HTML file for each topic*.

   `args.create.main.files.xsl`

   Specify this property if you want to customize the *XSLT stylesheet used to produce the main HTML file*.

   `args.createTocXML.xsl`

   Specify this property if you want to customize the *XSLT stylesheet used to produce the `toc.xml` file*. The toc.xml file contains information extracted from DITA map and it is mainly used to create the WebHelp TOC.

   For example, to customize a WebHelp Responsive transformation type, the build file should look like:

```
<project basedir="." name="Webhelp Responsive Customization">
 <target name="dita2webhelp-responsive-custom">
   <!--
     Override this property if you want to customize the XSLT stylesheet
     used to produce an HTML file for each topic
```

```
    -->
    <property
      name="args.wh.xsl"
      value="${dita.plugin.com.oxygenxml.webhelp.responsive.custom.dir}
                                                /xsl/dita2webhelpCustom.xsl"/>
    <!--
      Override this property if you want to customize the XSLT stylesheet
      used to produce the main HTML file.
    -->
    <property
      name="args.create.main.files.xsl"
      value="${dita.plugin.com.oxygenxml.webhelp.responsive.custom.dir}
                                              /xsl/createMainFilesCustom.xsl"/>
    <!--
      Override this property if you want to customize the XSLT stylesheet
      used to produce the toc.xml file.
    -->
    <property
      name="args.createTocXML.xsl"
      value="${dita.plugin.com.oxygenxml.webhelp.responsive.custom.dir}
                                              /xsl/createTocXMLCustom.xsl"/>
    <!--
      Depending on which version of WebHelp you want to customize,
      you need to delegate to different build targets:
      * dita2webhelp-responsive - when you are customizing the Webhelp Responsive
      * dita2webhelp-mobile - when you are customizing the Webhelp Mobile
      * dita2webhelp - when you are customizing the Webhelp Classic
    -->
    <antcall target="dita2webhelp-responsive"/>
  </target>
 </project>
```

Note: Depending on which version of WebHelp you want to customize, you need to call one of the following build targets:

- **dita2webhelp-responsive** - For WebHelp Responsive (with or without feedback) output.
- **dita2webhelp** - For WebHelp Classic (with or without feedback) output.
- **dita2webhelp-mobile** - For WebHelp Classic Mobile output (deprecated).

5. Create an **xsl** directory in the plugin customization directory (that you created in step 1) to store the customized XSLT stylesheets.

Referencing the WebHelp XSLT Stylesheets from Your Customizations

Note that your customization stylesheets should import the original stylesheets that they override. To reference the WebHelp stylesheets, you can use the **plugin:com.oxygenxml.dita-ot.plugin.webhelp** prefix. This prefix is rewritten by an XML catalog to the WebHelp root directory.

Customizing the `dita2webhelp.xsl` Stylesheet

The XSLT stylesheet that customizes the WebHelp **dita2webhelp.xsl** stylesheet should look like:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:math="http://www.w3.org/2005/xpath-functions/math"
   exclude-result-prefixes="xs math"
   version="2.0">
   <!--
     Import the original stylesheet used to produce an HTML file for each topic.
   -->
   <xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/
                                        dita/responsive/dita2webhelp.xsl"/>
   <!--
     Please add your customization templates here.
   -->
</xsl:stylesheet>
```

Depending on which version of WebHelp you want to customize, you need to import one of the following XSLT stylesheets:

- **dita2webhelp-responsive** (WebHelp Responsive) -

  `<xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/responsive/dita2webhelp.xsl"/>`
- **dita2webhelp** (WebHelp Classic) -

  `<xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/desktop/dita2webhelp.xsl"/>`
- **dita2webhelp-mobile** (WebHelp Classic Mobile) -

```
<xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/mobile/
dita2webhelp.xsl"/>
```

Customizing the `createMainFiles.xsl` Stylesheet

The XSLT stylesheet that customizes the WebHelp **createMainFiles.xsl** stylesheet should look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:math="http://www.w3.org/2005/xpath-functions/math"
    exclude-result-prefixes="xs math"
    version="2.0">
    <!--
        Import the original stylesheet used to produce the main HTML file.
    -->
    <xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/
                                            responsive/createMainFiles.xsl"/>
    <!--
        Please add your customization templates here.
    -->

</xsl:stylesheet>
```

Depending on which version of WebHelp you want to customize, you need to import one of the following XSLT stylesheets:

- **dita2webhelp-responsive** (WebHelp Responsive) -

  ```
  <xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/
  responsive/createMainFiles.xsl"/>
  ```
- **dita2webhelp** (WebHelp Classic) -

  ```
  <xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/desktop/
  createMainFiles.xsl"/>
  ```
- **dita2webhelp-mobile** (WebHelp Classic Mobile) -

  ```
  <xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/mobile/
  createMainFiles.xsl"/>
  ```

Customizing the `tocDita.xsl` File

The XSLT stylesheet that customizes the WebHelp **tocDita.xsl** stylesheet should look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:math="http://www.w3.org/2005/xpath-functions/math"
    exclude-result-prefixes="xs math"
    version="2.0">
    <!--
        Import the original stylesheet used to produce the toc.xml file.
    -->
    <xsl:import href="plugin:com.oxygenxml.dita-ot.plugin.webhelp:xsl/dita/tocDita.xsl"/>
    <!--
        Please add your customization templates here.
    -->

</xsl:stylesheet>
```

Removing the *Previous*/*Next* Links from WebHelp Classic Output

The **Previous** and **Next** links that are created at the top area of each WebHelp Classic page can be hidden with a CSS code.

To remove these links from WebHelp Classic output, follow these steps:

1. Add the following CSS code in a custom CSS stylesheet:

   ```
   .navparent, .navprev, .navnext {
       visibility:hidden;
   }
   ```
2. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.

a. For a DITA transformation, set the `args.css` parameter to the path of your custom CSS file. Also, set the `args.copycss` parameter to `yes` to automatically copy your custom CSS in the output folder when the transformation scenario is processed.
b. For a DocBook transformation, set the `html.stylesheet` parameter to the path of your custom CSS file.

3. Run the transformation scenario.

Search Engine Optimization for DITA WebHelp

A **DITA Map WebHelp** transformation scenario can be configured to produce a `sitemap.xml` file that is used by search engines to aid crawling and indexing mechanisms. A *sitemap* lists all pages of a WebHelp system and allows webmasters to provide additional information about each page, such as the date it was last updated, change frequency, and importance of each page in relation to other pages in your WebHelp deployment.

The structure of the `sitemap.xml` file looks like this:

```
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.example.com/topics/introduction.html</loc>
    <lastmod>2014-10-24</lastmod>
    <changefreq>weekly</changefreq>
    <priority>0.5</priority>
  </url>
  <url>
    <loc>http://www.example.com/topics/care.html#care</loc>
    <lastmod>2014-10-24</lastmod>
    <changefreq>weekly</changefreq>
    <priority>0.5</priority>
  </url>
   .  .  .
</urlset>
```

Each page has a `<url>` element structure containing additional information, such as:

- `loc` - the URL of the page. This URL must begin with the protocol (such as `http`), if required by your web server. It is constructed from the value of the `webhelp.sitemap.base.url` parameter from the transformation scenario and the relative path to the page (collected from the `href` attribute of a `topicref` element in the *DITA map*).

  Note: The value must have fewer than 2,048 characters.
- `lastmod` (optional) - the date when the page was last modified. The date format is `YYYY-MM-DD`.
- `changefreq` (optional) - indicates how frequently the page is likely to change. This value provides general information to assist search engines, but may not correlate exactly to how often they crawl the page. Valid values are: `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, and `never`. The first time the `sitemap.xml` file is generated, the value is set based upon the value of the `webhelp.sitemap.change.frequency` parameter in the DITA WebHelp transformation scenario. You can change the value in each `url` element by editing the `sitemap.xml` file.

  Note: The value `always` should be used to describe documents that change each time they are accessed. The value `never` should be used to describe archived URLs.
- `priority` (optional) - the priority of this page relative to other pages on your site. Valid values range from 0.0 to 1.0. This value does not affect how your pages are compared to pages on other sites. It only lets the search engines know which pages you deem most important for the crawlers. The first time the `sitemap.xml` file is generated, the value is set based upon the value of the `webhelp.sitemap.priority` parameter in the DITA WebHelp transformation scenario. You can change the value in each `url` element by editing the `sitemap.xml` file.

Creating and Editing the `sitemap.xml` File

Follow these steps to produce a `sitemap.xml` file for your WebHelp system, which can then be edited to fine-tune search engine optimization:

1. **Edit** the transformation scenario you currently use for obtaining your WebHelp output. This opens the **Edit DITA Scenario** dialog box.
2. Open the **Parameters** tab and set a value for the following parameters:
   - `webhelp.sitemap.base.url` - the URL of the location where your WebHelp system is deployed

     Note: This parameter is required for Oxygen XML Editor to generate the `sitemap.xml` file.

- `webhelp.sitemap.change.frequency` - how frequently the WebHelp pages are likely to change (accepted values are: `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, and `never`)
- `webhelp.sitemap.priority` - the priority of each page (value ranging from 0.0 to 1.0)

3. Run the transformation scenario.
4. Look for the `sitemap.xml` file in the transformation's output folder. Edit the file to fine-tune the parameters of each page, according to your needs.

Support for Right-to-Left (RTL) Oriented Languages for DITA WebHelp

To activate support for RTL (right-to-left) languages in WebHelp output, edit the *DITA map* and set the `xml:lang` attribute on its root element (`map`). The corresponding attribute value can be set for following RTL languages:

- `ar-eg` - Arabic
- `he-il` - Hebrew
- `ur-pk` - Urdu

WebHelp Classic Runtime Additional Parameters

A deployed WebHelp system can accept the following GET parameters:

- `log` - The value can be `true` or `false` (default value). When set to `true`, it enables JavaScript debugging.
- `contextId` - The WebHelp JavaScript engine will look for this value in the `context-help-map.xml` mapping file and load the corresponding help page. For more information, see the *Context-Sensitive WebHelp System* topic.

  Note: You can use an *anchor* in the `contextId` parameter to jump to a specific section in a document. For example, `contextId=topicID#anchor`.
- `appname` - You can use this parameter in conjunction with `contextId` to search for this value in the corresponding *`appname` attribute value* in the mapping file.

  ```
  http://localhost/webhelp/index.html?contextId=topicID&appname=myApplication
  ```
- `toc.visible` - The value can be `true` (default value) or `false`. When set to `false`, the table of contents will be collapsed when you load the WebHelp page.
- `searchQuery` - You can use this parameter to perform a search operation when WebHelp is loaded. For example, if you want to open WebHelp showing all search results for *growing flowers*, the URL should look like this: `http://localhost/webhelp/index.html?searchQuery=growing%20flowers`.

Related Information:

Context-Sensitive WebHelp Classic System on page 1021

Context-Sensitive WebHelp Classic System

Context-sensitive help systems assist users by providing specific informational topics for certain components of a user interface, such as a button or window. This mechanism works based on mappings between a unique ID defined in the topic and a corresponding HTML page.

Generating Context-Sensitive Help

When WebHelp Classic output is generated by Oxygen XML Editor, the transformation process produces an XML mapping file called `context-help-map.xml` and copies it in the output folder of the transformation. This XML file maps an ID to a corresponding HTML page through an `appContext` element, as in the following example:

```
<map productID="oxy-webhelp" productVersion="1.1">
  <appContext helpID="myapp-functionid1" path="tasks/app-help1.html"/>
  <appContext helpID="myapp-functionid2" path="tasks/app-help1.html"/>
    .  .  .
</map>
```

The possible attributes are as follows:

`helpID`

A Unique ID provided by a topic from two possible sources (`resourceid` element or `id` attribute):

`resourceid`

The `resourceid` element is mapped into the `appContext` element and can be specified in either the `topicref` within a *DITA map* or in a `prolog` within a DITA topic. The `resourceid` element accepts the following attributes:

- **`appname`** - A name for the external application that references the topic. If this attribute is not specified, its value is considered to be empty (" ").
- **`appid`** - An ID used by an application to identify the topic.
- **`id`** - Specifies a value that is used by a specific application to identify the topic, but this attribute is ignored if an **`appid`** attribute is used.

Note: Multiple `appid` values can be associated with a single `appname` value (and multiple `appname` values can be associated with a single `appid` value), but the values for both attributes work in combination to specify a specific ID for a specific application, and therefore each combination of values for the `appid` and `appname` attributes should be unique within the context of a single *root map*. For example, suppose that you need two different functions of an application to both open the same WebHelp page.

Example: `resourceid` Specified in a DITA Map

The `resourceid` element can be specified in a `topicmeta` element within a `topicref`.

```
<map title="App Help">
  <topicref href="app-help1.dita" type="task">
     <topicmeta>
       <resourceid appname="myapp" appid="functionid1"/>
       <resourceid appname="myapp" appid="functionid2"/>
     </topicmeta>
  </topicref>
</map>
```

Example: `resourceid` Specified in a DITA Topic

The `resourceid` element can be specified in a `prolog` element within a DITA topic.

```
<task id="app-help1">
  <title>My App Help</title>
  <prolog>
    <resourceid appname="myapp" appid="functionid1"/>
    <resourceid appname="myapp" appid="functionid2"/>
  </prolog>
...
</task>
```

For more information about the `resourceid` element, see *DITA Specifications: <resourceid>*.

`id`

If a `resourceid` element is not declared in the *DITA map* or DITA topic (as described above), the `id` attribute that is set on the topic root element is mapped into the `appContext` element.

Important: You should ensure that these defined IDs are unique in the context of the entire DITA project. If the IDs are not unique, the transformation scenario will display warning messages in the transformation console output and the help system will not work properly.

`path`

The path to a corresponding WebHelp page. This path is relative to the location of the `context-help-map.xml` mapping file.

`productID` (Applicable only if you are using the WebHelp Classic with Feedback transformation scenario)

The ID of the product for your documentation project.

`productVersion` (Applicable only if you are using the WebHelp Classic with Feedback transformation scenario)

The version of the product for your documentation project.

There are two ways of implementing context-sensitive help in your system:

- The XML mapping file can be loaded by a PHP script on the server side. The script receives the `contextId` value and will look it up in the XML file.

- Invoke one of the WebHelp system files `index.html` or `index_frames.html` and pass the `contextId` parameter with a specific value. The WebHelp system will automatically open the help page associated with the value of the `contextId` parameter.

The following example will open a *frameless* version of the WebHelp system showing the page associated with the ID `dialog1ID`:

```
index.html?contextId=dialog1ID
```

The following example will open a *frameset* version of the WebHelp system showing the page associated with the ID `view1ID`:

```
index_frames.html?contextId=view1ID
```

Tip: You can use an *anchor* in the `contextId` parameter to jump to a specific section in a document. For example, `contextId=topicID#anchor`.

Context-Sensitive Queries

You can use the URL field in your browser to search for topics in a context-sensitive WebHelp system with the assistance of the following parameters:

- `contextId` - The WebHelp JavaScript engine will look for this value in the `context-help-map.xml` mapping file and load the corresponding help page. For more information, see the *Context-Sensitive WebHelp System* topic.

  Note: You can use an *anchor* in the `contextId` parameter to jump to a specific section in a document. For example, `contextId=topicID#anchor`.
- `appname` - You can use this parameter in conjunction with `contextId` to search for this value in the corresponding *appname attribute value* in the mapping file.

  ```
  http://localhost/webhelp/index.html?contextId=topicID&appname=myApplication
  ```

Related Information:
WebHelp Classic Runtime Additional Parameters on page 1020

Publishing WebHelp Classic Output on a SharePoint Site

Since WebHelp output must be published locally, on the same machine where the WebHelp process is running, to publish your files directly to a SharePoint library you need to map a network drive to connect to SharePoint and change your file extensions to `.aspx`, as described in the steps below.

To publish WebHelp Classic output on a SharePoint site, follow this procedure:

1. Map a network drive to connect to your SharePoint library. For more information, see: *https://support.microsoft.com/en-us/kb/2616712*.
2. To allow browsers to open your published files (rather than downloading them), you need to change the file extensions from `.html` to `.aspx`. Fortunately, this can be done in the transformation scenario.
   a. *Edit the WebHelp transformation scenario* and open the **Parameters** tab.
      a. For a DITA transformation, set the `args.outext` parameter to `.aspx`.
      b. For a DocBook transformation, set the `html.ext` parameter to `.aspx`.
   b. Run the transformation scenario.

Using the Oxygen XML WebHelp Plugin to Automate Output

Oxygen XML WebHelp plugin allows you to use a command line interface script to obtain WebHelp output from DITA and DocBook documents. Note that the Oxygen XML WebHelp plugin is a standalone product with its own licensing terms and cannot be used with a Oxygen XML Editor license.

The WebHelp output files created with the Oxygen XML WebHelp plugin are the same as the output files produced when you run DITA or DocBook to WebHelp transformation scenarios from within Oxygen XML Editor.

When an automated process is required due to the amount of output needed, do the following:

1. Install the Oxygen XML WebHelp plugin.
2. Acquire a Oxygen XML WebHelp license from *https://www.oxygenxml.com/buy_webhelp.html*.
3. Integrate the Oxygen XML WebHelp plugin with *DITA* or *DocBook*.

Oxygen XML WebHelp Plugin for DITA

To transform DITA documents using the Oxygen XML WebHelp plugin, first integrate the plugin with the DITA Open Toolkit. The purpose of the integration is to add the following transformation types to the DITA Open Toolkit:

- **webhelp-responsive** - The transformation that produces **WebHelp Responsive** and **WebHelp Responsive with Feedback** output for desktop and mobile devices.
- **webhelp** - The transformation that produces **WebHelp Classic** output for desktop.
- **webhelp-feedback** - The transformation that produces feedback-enabled **WebHelp Classic with Feedback** for desktop.
- **webhelp-mobile** - The transformations that produces **WebHelp Classic Mobile** output for mobile devices.

Integrating the Oxygen XML WebHelp Plugin with the DITA Open Toolkit

The requirements of the Oxygen XML WebHelp plugin for the DITA Open Toolkit are as follows:

- Java Virtual Machine 1.6 or newer if you want to use DITA-OT 1.8.5 or Java Virtual Machine 1.8 or newer if you want to use DITA-OT 2.4.4.
- DITA Open Toolkit 1.8.5 or 2.4.4 (includes Saxon 9.x libraries).

  Note: The Oxygen XML WebHelp Plugin has been tested with these two specific versions (**DITA-OT 1.8.5** or **DITA-OT 2.4.4**) and therefore they are the recommended versions.

To integrate the Oxygen XML WebHelp plugin with the DITA Open Toolkit, follow these steps:

1. Download and install a *Java Virtual Machine* version compatible with the DITA-OT version.
2. Download and unpack the *DITA Open Toolkit* version 1.8.5 or 2.4.4.
3. Go to *Oxygen XML WebHelp website*, download the latest DITA-OT version of the installation kit, and unzip it.
4. Copy all *plugin* directories from the unpacked archive to the `plugins` directory of the DITA OT distribution. This is necessary to enable certain functionality. For example, the `com.oxygenxml.highlight` directory adds syntax highlight capabilities to your WebHelp output for `codeblock` sections that contain source code snippets (XML, Java, JavaScript).
5. If you have not already done so, *copy your license key into a `licensekey.txt` file* and place it in the *`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp` directory.
6. In the home directory of the DITA Open Toolkit, run `ant -f integrator.xml`.

Related Information:

Licensing the Oxygen XML WebHelp Plugin for DITA OT on page 1046
Upgrading the Oxygen XML WebHelp Plugin for DITA OT on page 1047
Customization Example: Apply Custom Styling to an External Transformation on page 1051
DITA OT PDF Customization Plugin

Licensing the Oxygen XML WebHelp Plugin for DITA OT

To register the license for the Oxygen XML WebHelp plugin for the DITA Open Toolkit, follow these steps:

1. Open the *`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp` directory and create a file called `licensekey.txt`.
2. In this file, copy your license key that you purchased for your Oxygen XML WebHelp plugin.

   The WebHelp transformation process reads the Oxygen XML WebHelp license key from this file. In case the file does not exit, or it contains an invalid license, an error message will be displayed.

Related Information:

Integrating the Oxygen XML WebHelp Plugin with the DITA Open Toolkit on page 1046
Upgrading the Oxygen XML WebHelp Plugin for DITA OT on page 1047

Upgrading the Oxygen XML WebHelp Plugin for DITA OT

To upgrade your Oxygen XML WebHelp plugin for the DITA Open Toolkit, follow these steps:

1. Navigate to the `plugins` directory of your DITA OT distribution and delete the old Oxygen XML WebHelp plugin files (`oxygen_custom.xsl`, `oxygen_custom_html.xsl`) and directories (`com.oxygenxml.highlight`, `com.oxygenxml.media`, `com.oxygenxml.webhelp`).
2. Go to *Oxygen XML WebHelp website*, download the latest DITA-OT version of the installation kit, and unzip it.
3. Copy all *plugin* directories from the unpacked archive to the `plugins` directory of the DITA OT distribution. This is necessary to enable certain functionality. For example, the `com.oxygenxml.highlight` directory adds syntax highlight capabilities to your WebHelp output for `codeblock` sections that contain source code snippets (XML, Java, JavaScript).
4. In the home directory of the DITA Open Toolkit, run `ant -f integrator.xml`.

Related Information:

Integrating the Oxygen XML WebHelp Plugin with the DITA Open Toolkit on page 1046

Licensing the Oxygen XML WebHelp Plugin for DITA OT on page 1046

Running an External DITA Transformation Using the Oxygen XML WebHelp Plugin

There are two main ways to run a DITA to WebHelp (***webhelp-responsive, webhelp, webhelp-feedback, webhelp-mobile***) transformation using the Oxygen XML WebHelp plugin:

Startup Script from WebHelp Plugin Method

The first method involves running the `ditaWebhelp.bat` or `ditaWebhelp.sh` startup script that comes bundled in the WebHelp plugin folder:

- `WEBHELP_PLUGIN_DIR\ditaWebhelp.bat` script file (Windows based systems)
- `WEBHELP_PLUGIN_DIR\ditaWebhelp.sh` script file (Unix/Linux based systems)

Before using the script to generate output, you must customize it to specify the paths to the Java VM, DITA Open Toolkit, and also to set the transformation type. To do this, open the script file and edit the following variables and parameters:

- `JVM_INSTALL_DIR` - Specifies the path to the Java Virtual Machine installation directory on your disk.
- `DITA_OT_INSTALL_DIR` - Specifies the path to DITA Open Toolkit installation directory on your disk.

  Note: The scripts reference the `dost-patches-DITA-1.8.jar` *JAR* file containing DITA OT 1.8.5-specific patches. If you use DITA OT 1.7, please update that reference to `dost-patches-DITA-1.7.jar`. If you use DITA OT 2.4.4, no patches are needed, so just remove the reference.
- `TRANSTYPE` - Specifies the type of the transformation you want to apply. You can set it to `webhelp-responsive`, `webhelp`, `webhelp-feedback` or `webhelp-mobile`.
- `DITA_MAP_BASE_DIR` - Specifies the path to the directory where the input *DITA map* file is located.
- `DITAMAP_FILE` - Specifies the input *DITA map* file.
- `DITAVAL_FILE` - Specifies the `.ditaval` input filter that the transformation process applies to the input *DITA map* file.
- `DITAVAL_DIR` - Specifies the path to the directory where the `.ditaval` file is located.
- `-Doutput.dir` - Specifies the output directory of the transformation.

Startup Script from DITA OT Distribution Method

The second method involves using the startup script that comes bundled with each DITA OT distribution.

DITA OT 1.8.5

For DITA Open Toolkit 1.8.5, the general instructions for running the startup script can be found here: *http://www.dita-ot.org/1.8/readme/building-output-using-cl-tool.html*.

DITA OT 2.4.4

For DITA Open Toolkit 2.4.4, the general instructions for running the startup script can be found here: *http://www.dita-ot.org/2.3/parameters/dita-command-arguments.html*.

DITA OT Startup Script Examples:

DITA-OT-DIR`\bin\[executable] -i [path_to_input.ditamap] -f [transformation_type]`

For example:

- *DITA-OT-DIR*`\bin\dita.bat -i c:\mySample.ditamap -f webhelp-responsive` (Windows based systems)
- *DITA-OT-DIR*`\bin\dita -i c:\mySample.ditamap -f webhelp-responsive` (Unix/Linux based systems)

  Note: For the `-format` (`-f`) argument, use one of the following *transformation types*: `webhelp-responsive`, `webhelp`, `webhelp-feedback` or `webhelp-mobile`.

The downside of this approach is the fact that you will lose certain small fixes and patches that Oxygen XML Editor adds to the automated DITA OT processing. For example, you may lose the ability to process DITA topics that contain entity references inside them.

Related Information:

Integrating the Oxygen XML WebHelp Plugin with the DITA Open Toolkit on page 1046
Customization Example: Apply Custom Styling to an External Transformation on page 1051
DITA OT PDF Customization Plugin

Additional Oxygen XML WebHelp Plugin Parameters for DITA

You can append the following parameters to the command line that runs the transformation:

`-Dwebhelp.copyright`

Adds a small copyright text that appears at the end of the **Table of Contents** pane.

`-Dwebhelp.custom.resources`

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

`-Dwebhelp.favicon`

The file path that points to an image to be used as a *favicon* in the WebHelp output.

`-Dwebhelp.footer.file` (not available for WebHelp Responsive systems)

Path to an XML file that includes the footer content for your WebHelp output pages. You can use this parameter to integrate social media features (such as widgets for Facebook™, Twitter™, Google Analytics, or Google+™). The file must be well-formed, each widget must be in separate `div` or `span` element, and the code for each `script` element is included in an XML comment (also, the start and end tags for the XML comment must be on a separate line). The following code exert is an example for adding a Facebook™ widget:

```
<div id="facebook">
  <div id="fb-root"/>
  <script>
    <!-- (function(d, s, id) { var js, fjs = d.getElementsByTagName(s)[0];
      if (d.getElementById(id)) return;
      js = d.createElement(s); js.id = id;
      js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
      fjs.parentNode.insertBefore(js, fjs); }
       (document, 'script', 'facebook-jssdk')); -->
    </script>
    <div data-share="true" data-show-faces="true" data-action="like"
       data-layout="standard" class="fb-like"/>
</div>
```

`-Dwebhelp.footer.include` (not available for WebHelp Responsive systems)

Specifies whether or not to include footer in each WebHelp page. Possible values: `yes`, `no`. If set to `no`, no footer is added to the WebHelp pages. If set to `yes` and the `webhelp.footer.file` parameter has a value, then the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default Oxygen XML Editor footer is inserted in each WebHelp page.

`-Dwebhelp.google.search.results`

A file path that specifies the location of a well-formed XHTML file containing the Google Custom Search Engine element `gcse:searchresults-only`. You can use all supported attributes for this element. It is recommend to set the `linkTarget` attribute to `frm` for frameless (*iframe*) version of WebHelp or to `contentWin` for the frameset version of WebHelp. The default value for this attribute is `_blank` and the

search results will be loaded in a new window. If this parameter is not specified, the following code will be used `<gcse:searchresults-only linkTarget="frm"></gcse:searchresults-only>`

`-Dwebhelp.google.search.script`

A file path that specifies the location of a well-formed XHTML file containing the Custom Search Engine script from Google.

`-Dwebhelp.logo.image.target.url` (not available for WebHelp Classic Mobile systems)

Specifies a target URL that is set on the logo image. When you click the logo image, you will be redirected to this address.

`-Dwebhelp.logo.image` (not available for WebHelp Classic Mobile systems)

Specifies a path to an image displayed as a logo in the left side of the output header.

`-Dwebhelp.product.id` (available only for Feedback-enabled systems)

This parameter specifies a short name for the documentation target, or product (for example, `mobile-phone-user-guide`, `hvac-installation-guide`).

Note: You can deploy documentation for multiple products on the same server.

Restriction: The following characters are not allowed in the value of this parameter: `< > / \ ' ( ) { } = ; * % + &`.

`-Dwebhelp.product.version` (available only for Feedback-enabled systems)

Specifies the documentation version number (for example, 1.0, 2.5, etc.). New user comments are bound to this version.

Note: Multiple documentation versions can be deployed on the same server.

Restriction: The following characters are not allowed in the value of this parameter: `< > / \ ' ( ) { } = ; * % + &`.

`-Dwebhelp.search.japanese.dictionary`

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Editor uses for indexing Japanese content in the WebHelp pages.

`-Dwebhelp.search.ranking`

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the search results that are displayed on the **Search** tab (default setting is `true`).

`-Dwebhelp.show.changes.and.comments`

When set to `yes`, user comments, replies to comments, and tracked changes are published in the WebHelp output. The default value is no.

`-Dwebhelp.sitemap.base.url`

Base URL for all the `loc` elements in the generated `sitemap.xml` file. The value of a `loc` element is computed as the relative file path from the `href` attribute of a `topicref` element from the *DITA map*, appended to this base URL value. The `loc` element is mandatory in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `sitemap.xml` file is not generated.

`-Dwebhelp.sitemap.change.frequency`

The value of the `changefreq` element in the generated `sitemap.xml` file. The `changefreq` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `changefreq` element is not added in `sitemap.xml`. Allowed values: <empty string> (default), `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, `never`.

`-Dwebhelp.sitemap.priority`

The value of the `priority` element in the generated `sitemap.xml` file. It can be set to any fractional number between 0.0 (least important priority) and 1.0 (most important priority). For example, 0.3, 0.5, or 0.8. The `priority` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `priority` element is not added in `sitemap.xml`.

`-Dwebhelp.skin.css` (available only for WebHelp Classic and WebHelp Classic with Feedback systems)

Path to a CSS file that sets the style theme in the output WebHelp pages. It can be one of the predefined skin CSS from the `OXYGEN_INSTALL_DIR\frameworks\docbook\xsl\com.oxygenxml.webhelp`

`\predefined-skins` directory, or it can be a custom skin CSS generated with the *Oxygen Skin Builder* web application.

Parameters Specific to WebHelp Responsive Output (available only for WebHelp Responsive and WebHelp Responsive with Feedback systems)

`-Dwebhelp.enable.search.autocomplete`

Specifies if the *Autocomplete* feature is enabled in the WebHelp search text field. The default value is `yes`.

`-Dwebhelp.fragment.after.body`

In the generated output it displays a given XHTML fragment after the page body. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`-Dwebhelp.fragment.after.logo_and_title`

In the generated output it displays a given XHTML fragment after the logo and title. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`-Dwebhelp.fragment.after.main.page.search`

In the generated output it displays a given XHTML fragment after the search field. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`-Dwebhelp.fragment.after.toc_or_tiles`

In the generated output it displays a given XHTML fragment after the table of contents or tiles in the main page. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`-Dwebhelp.fragment.after.top_menu`

In the generated output it displays a given XHTML fragment after the top menu. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`-Dwebhelp.fragment.before.body`

In the generated output it displays a given XHTML fragment before the page body. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`-Dwebhelp.fragment.before.logo_and_title`

In the generated output it displays a given XHTML fragment before the logo and title. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`-Dwebhelp.fragment.before.main.page.search`

In the generated output it displays a given XHTML fragment before the search field. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`-Dwebhelp.fragment.before.toc_or_tiles`

In the generated output it displays a given XHTML fragment before the table of contents or tiles in the main page. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`-Dwebhelp.fragment.before.top_menu`

In the generated output it displays a given XHTML fragment before the top menu. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`-Dwebhelp.fragment.footer`

In the generated output it displays a given XHTML fragment as the page footer. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`-Dwebhelp.fragment.head`

In the generated output it displays a given XHTML fragment as a page header. The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`-Dwebhelp.fragment.welcome`

In the generated output it displays a given XHTML fragment as a welcome message (or title). The value of the parameter can be either an XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`-Dwebhelp.merge.nested.topics.related.links`

Specifies if the related links from nested topics will be merged with the links in the parent topic. Thus the links will be moved from the topic content to the related links component and all of the links from the same group

(for example, *Related Tasks*, *Related References*, *Related Information*) are merged into a single group. The default value is `yes`.

`-Dwebhelp.show.breadcrumb`

Specifies if the breadcrumb component will be presented in the output. The default value is `yes`.

`-Dwebhelp.show.child.links`

Specifies if child links will be generated in the output for all topics that have subtopics. The default value is `no`.

`-Dwebhelp.show.indexterms.link`

Specifies if an icon that links to the index will be presented in the output. The default value is `yes`.

`-Dwebhelp.show.main.page.tiles`

Specifies if the tiles component will be presented in the main page of the output. For a *tree* style layout, this parameter should be set to `no`.

`-Dwebhelp.show.main.page.toc`

Specifies if the table of contents will be presented in the main page of the output. The default value is `yes`.

`-Dwebhelp.show.navigation.links`

Specifies if navigation links will be presented in the output. The default value is `yes`.

`-Dwebhelp.show.print.link`

Specifies if a print link or icon will be presented within each topic in the output. The default value is `yes`.

`-Dwebhelp.show.side.toc`

Specifies if a side table of contents will be presented on the right side of each topic in the output. The default value is `yes`.

`-Dwebhelp.show.top.menu`

Specifies if a menu will be presented at the topic of the main page in the output. The default value is `yes`.

`-Dwebhelp.top.menu.depth`

Specifies the maximum depth level of the topics that will be included in the top menu. The default value is 2.

Note: Note that the `fix.external.refs.com.oxygenxml` parameter is not supported in Oxygen XML WebHelp plugin. This parameter is normally used to specify whether or not the application tries to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references.

Further Customization

If you need to further customize the transformation process, you can append other DITA-OT parameters as well. Any parameter that you want to append must follow the `-D` model of the above parameters. For example, to append the `args.csspath` parameter, use:

```
-Dargs.csspath=[CSS_FILE_PATH]
```

where `[CSS_FILE_PATH]` is the location of the directory that contains the CSS file.

Related Information:

DITA OT Parameter Reference

Customization Example: Apply Custom Styling to an External Transformation

Suppose that you want to apply custom styling to an external DITA transformation using the Oxygen XML WebHelp plugin for DITA OT. The following procedure provides an example of how you could achieve this using a custom CSS file and some of the *Oxygen XML WebHelp plugin parameters*.

External Transformation Customization Example

To apply your own custom styling to an external DITA transformation using Oxygen XML WebHelp plugin, follow these steps:

1. Download and install a *Java Virtual Machine* version compatible with the DITA-OT version.
2. Download and unpack the *DITA Open Toolkit* version 1.8.5 or 2.4.4.

3. Go to *Oxygen XML WebHelp website*, download the latest DITA-OT version of the installation kit, and unzip it.
4. Copy all *plugin* directories from the unpacked archive to the `plugins` directory of the DITA OT distribution. This is necessary to enable certain functionality. For example, the `com.oxygenxml.highlight` directory adds syntax highlight capabilities to your WebHelp output for `codeblock` sections that contain source code snippets (XML, Java, JavaScript).
5. If you have not already done so, *copy your license key into a `licensekey.txt` file* and place it in the *`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp` directory.
6. In the home directory of the DITA Open Toolkit, run `ant -f integrator.xml`.
7. Create a new directory for your custom template (*`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp/templates/dita/MyCustomTemplate`).
8. Create a new directory for your custom skin (*`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp/templates/dita/MyCustomTemplate/variants/tree/MyCustomSkin`).
9. Copy the *`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp/templates/dita/bootstrap/variants/tree/light/skin.css` file to the custom skin directory you just created (*`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp/templates/dita/MyCustomTemplate/variants/tree/MyCustomSkin`).
10. Copy the *`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp/templates/dita/bootstrap/variants/tree/light/resources` directory to the custom skin directory you just created (*`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp/templates/dita/MyCustomTemplate/variants/tree/MyCustomSkin`).
11. Use your system command console to run a command in the *`DITA-OT-DIR`*`/bin` folder that will invoke the external transformation and include the *Oxygen XML WebHelp plugin parameters* that you desire for your customization.

    For example: `dita.bat -i c:\myMap.ditamap -f webhelp-responsive -Dwebhelp.responsive.template.name=MyCustomTemplate -Dwebhelp.responsive.variant.name=tree -Dwebhelp.responsive.skin.name=MyCustomSkin -Dargs.breadcrumbs=yes -Dwebhelp.logo.image=c:\Mylogo.png -Dwebhelp.show.side.toc=no -Dwebhelp.show.top.menu=yes -o c:\MyOutputDirectory`
12. Check the output directory to make sure it now contains your custom skin directory.
13. Modify the *`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp/templates/dita/MyCustomTemplate/variants/tree/MyCustomSkin/skin.css` file according to your needs and check the output to make sure your custom styles are applied properly.

Related Information:

Integrating the Oxygen XML WebHelp Plugin with the DITA Open Toolkit on page 1046

Configuring the Comment Database for DITA WebHelp Systems with Feedback

If you run the ***webhelp-responsive*** *or* ***webhelp-feedback*** transformation using the WebHelp plugin, you need to configure the database that holds the user comments.

The instructions for configuring the database are presented in the `installation.html` file, located at: `[DITA_MAP_BASE_DIR]/out/[TRANSFORM_TYPE]/oxygen-webhelp/resources`. The `installation.html` file is created by the transformation process.

Building Oxygen XML WebHelp Output on Jenkins

This procedure assumes that you have already *integrated, configured, and registered the WebHelp plugin with the DITA Open Toolkit*.

To integrate WebHelp output with the Jenkins continuous integration tool, follow these steps:

1. Create a Maven project to incorporate the DITA-OT that already integrates Oxygen XML WebHelp.
2. Go to the root of your Maven project and edit the `pom.xml` file to include the following fragment:

```
<properties>
  <oxygen-webhelp>${basedir}/tools/DITA-OT1.8.5/
       plugins/com.oxygenxml.webhelp</oxygen-webhelp>
</properties>

<plugin>
  <artifactId>exec-maven-plugin</artifactId>
  <groupId>org.codehaus.mojo</groupId>
  <executions>
```

```
    <execution><!-- Run our version calculation script -->
      <id>Version Calculation</id>
      <phase>generate-sources</phase>
      <goals>
        <goal>exec</goal>
      </goals>
      <configuration>
        <executable>${oxygen-webhelp}/ditaWebhelp.bat</executable>
      </configuration>
    </execution>
  </executions>
</plugin>
```

Note: In the fragment above it is assumed that you are using DITA-OT version 1.8.5. If you are using another version, please adjust the path accordingly.

3. Go to the Jenkins top page and *create a new Jenkins job*. Configure this job to suit your particular requirements, such as the build frequency and location of the Maven project.

Building Oxygen XML WebHelp Output on Travis CI

This topic assumes you have a DITA project hosted on a GitHub public or private repository.

The goal of this tutorial is to help you setup a Travis continuous integration job that automatically publishes your DITA project to *GitHub pages* after every commit. The published website will contain a feedback link on each page that would allow a contributor to easily suggest changes to the documentation by creating a pull request on GitHub with just a few clicks.

Enable the Travis CI Build

1. *Sign in to Travis CI* with your GitHub account, accepting the GitHub *access permissions confirmation*.
2. Once you are signed in, and you have synchronized your GitHub repositories, go to your *profile page* and enable Travis CI for the repository you want to build.

Configure the Travis CI Build in your GitHub Project

1. Checkout your GitHub project locally.
2. Copy the `.travis` folder from *here* to the root directory of your project.
3. In the root of your GitHub project, add a file called `.travis.yml` with the following content:

```
language: dita
install:
  - echo "Installed"
script:
  - sh .travis/publish.sh
after_success:
  - sh .travis/deploy.sh
env:
  global:
    - DITAMAP=/path/to/your/ditamap/file
    - DITAVAL=/path/to/your/ditaval/file
    - ANT_OPTS=-Xmx1024M
```

Note: Replace `/path/to/your/ditamap/file` and `/path/to/your/ditaval/file` with the appropriate paths to your *DITA map* and `ditaval` files.

4. Create a GitHub personal access token by following *this procedure*.
5. *Define an environment variable in the repository settings* that has the name `GH_TOKEN` and the value equal with the GitHub personal access token created earlier.

Register Your License Key

1. Edit your `.gitignore` file (or create it if it does not already exist) and add the following line:

```
licenseKey.txt
```

2. Copy your WebHelp license to the root of your GitHub project in a file called `licenseKey.txt`.

   Important: The `licenseKey.txt` file should not be committed to GitHub as it contains a license key that is issued only to you.

3. *Encrypt the license key file* and **add** it to the `.travis.yml` configuration file. This way only the Travis CI server will be able to decrypt it during the build process.

Commit to GitHub

1. Commit the following files and folders and push the commit to GitHub:

```
git add .gitignore licenseKey.txt.enc .travis.yml .travis/
git commit -m "Set up the Travis CI publishing system"
git push
```

2. Create a `gh-pages` branch in your GitHub project where the WebHelp output will be published. You can follow the procedure *here*.

Oxygen XML WebHelp Plugin for DocBook

To transform DocBook documents using the Oxygen XML WebHelp plugin, first integrate the plugin with the DocBook XSL distribution. The purpose of the integration is to add the following transformation types to the DocBook XSL distribution:

- **webhelp** - The transformation that produces **WebHelp Classic** output for desktop.
- **webhelp-feedback** - The transformation that produces feedback-enabled **WebHelp Classic with Feedback** for desktop.
- **webhelp-mobile** - The transformations that produces **WebHelp Classic Mobile** output for mobile devices.
- **webhelp-responsive** - The transformation that produces **WebHelp Responsive** and **WebHelp Responsive with Feedback** output for desktop and mobile devices.

Integrating the Oxygen XML WebHelp Plugin with the DocBook XSL Distribution

The WebHelp plugin transformations run as an Ant build script. The requirements are:

- Ant 1.8 or later
- Java Virtual Machine 1.6 or later
- DocBook XSL 1.78.1 or later
- Saxon 6.5.5
- Saxon 9.1.0.8

To integrate the Oxygen XML WebHelp plugin with the DocBook XSL distribution, follow these steps:

1. Download and install a *Java Virtual Machine* 1.6 or later.
2. Download and install *Ant 1.8.* or later.
3. Go to *Oxygen XML WebHelp website*, download the *DocBook XSL 1.78.1* (or later) installation kit, and unzip it in the DocBook XSL directory (*[OXYGEN_INSTALL_DIR]*`/frameworks/docbook/xsl`).
   The DocBook XSL directory should now contain a new subdirectory named `com.oxygenxml.webhelp` and two new files, `oxygen_custom.xsl` and `oxygen_custom_html.xsl`.
4. If you have not already done so, *copy your license key into a `licensekey.txt` file* and place it in the *[OXYGEN_INSTALL_DIR]*`/frameworks/docbook/xsl/com.oxygenxml.webhelp` directory.
5. Download and unzip *saxon6-5-5.zip* on your computer.
6. Download and unzip *saxonb9-1-0-8j.zip* on your computer.

Licensing the Oxygen XML WebHelp Plugin for DocBook

To register the license for the Oxygen XML WebHelp plugin for the DocBook XSL distribution, follow these steps:

1. Create a `licenseKey.txt` file in the `com.oxygenxml.webhelp` subdirectory of the DocBook XSL directory.
2. In this file, copy the license key that you purchased for your Oxygen XML WebHelp plugin.

   The WebHelp transformation process reads the Oxygen XML Editor license key from this file. If the file does not exit, or it contains an invalid license, an error message is displayed.

Upgrading the Oxygen XML WebHelp Plugin for DocBook

To upgrade your Oxygen XML WebHelp plugin for DocBook, follow these steps:

1. Navigate to the *[OXYGEN_INSTALL_DIR]*`/frameworks/docbook/xsl` directory and delete the old Oxygen XML WebHelp plugin files (`oxygen_custom.xsl`, `oxygen_custom_html.xsl`) and directory (`com.oxygenxml.webhelp`).

2. Go to *Oxygen XML WebHelp website*, download the latest DocBook version of the installation kit, and unzip it in the DocBook XSL directory (`[OXYGEN_INSTALL_DIR]/frameworks/docbook/xsl`).
   The DocBook XSL directory should now contain a new subdirectory named `com.oxygenxml.webhelp` and two new files, `oxygen_custom.xsl` and `oxygen_custom_html.xsl`.

Running an External DocBook Transformation Using the WebHelp Plugin

To run a DocBook to WebHelp (***webhelp, webhelp-feedback, webhelp-mobile***) transformation using the Oxygen XML WebHelp plugin, use:

- The `docbook.bat` script file for Windows based systems.
- The `docbook.sh` script file for Unix/Linux based systems.

  Note: You can call these files in an automated process or from the command line.

The `docbook.bat` and the `docbook.sh` files are located in the home directory of the Oxygen XML WebHelp Plugin. Before using them to generate an WebHelp system, customize them to match the paths to the JVM, DocBook XSL distribution and Saxon engine, and also to set the transformation type. To do this, open a script file and edit the following variables:

- `JVM_INSTALL_DIR` - Specifies the path to the Java Virtual Machine installation directory on your disk.
- `ANT_INSTALL_DIR` - Specifies the path to the installation directory of Ant.
- `SAXON_6_DIR` - Specifies the path to the installation directory of Saxon 6.5.5.
- `SAXON_9_DIR` - Specifies the path to the installation directory of Saxon 9.1.0.8.
- `DOCBOOK_XSL_DIR` - Specifies the path to the installation directory of the DocBook XSL distribution.
- `TRANSTYPE` - Specifies the type of the transformation you want to apply. You can set it to `webhelp`, `webhelp-feedback` and `webhelp-mobile`.
- `INPUT_DIR` - Specifies the path to the input directory, containing the input XML file.
- `XML_INPUT_FILE` - Specifies the name of the input XML file.
- `OUTPUT_DIR` - Specifies the path to the output directory where the transformation output is generated.
- `DOCBOOK_XSL_DIR_URL` - Specifies the path to the directory of the DocBook XSL distribution in URL format.

Additional Oxygen XML WebHelp Plugin Parameters for DocBook

You can append the following parameters to the command line that runs the transformation:

`-Dwebhelp.copyright`

Adds a small copyright text that appears at the end of the **Table of Contents** pane.

`-Dwebhelp.favicon`

The file path that points to an image to be used as a *favicon* in the WebHelp output.

`-Dwebhelp.footer.file`

Path to an XML file that includes the footer content for your WebHelp output pages. You can use this parameter to integrate social media features (such as widgets for Facebook™, Twitter™, Google Analytics, or Google+™). The file must be well-formed, each widget must be in separate `div` or `span` element, and the code for each `script` element is included in an XML comment (also, the start and end tags for the XML comment must be on a separate line). The following code exert is an example for adding a Facebook™ widget:

```
<div id="facebook">
  <div id="fb-root"/>
  <script>
    <!-- (function(d, s, id) { var js, fjs = d.getElementsByTagName(s)[0];
      if (d.getElementById(id)) return;
      js = d.createElement(s); js.id = id;
      js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
      fjs.parentNode.insertBefore(js, fjs); }
       (document, 'script', 'facebook-jssdk')); -->
    </script>
    <div data-share="true" data-show-faces="true" data-action="like"
       data-layout="standard" class="fb-like"/>
</div>
```

`-Dwebhelp.footer.include`

Specifies whether or not to include footer in each WebHelp page. Possible values: `yes`, no. If set to no, no footer is added to the WebHelp pages. If set to `yes` and the `webhelp.footer.file` parameter has a value,

then the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default Oxygen XML Editor footer is inserted in each WebHelp page.

`-Dwebhelp.product.id` (available only for Feedback-enabled systems)

This parameter specifies a short name for the documentation target, or product (for example, `mobile-phone-user-guide`, `hvac-installation-guide`).

Note: You can deploy documentation for multiple products on the same server.

Restriction: The following characters are not allowed in the value of this parameter: `< > / \ ' ( ) { } = ; * % + &`.

`-Dwebhelp.product.version` (available only for Feedback-enabled systems)

Specifies the documentation version number (for example, 1.0, 2.5, etc.). New user comments are bound to this version.

Note: Multiple documentation versions can be deployed on the same server.

Restriction: The following characters are not allowed in the value of this parameter: `< > / \ ' ( ) { } = ; * % + &`.

If you need to further customize your transformation, other DocBook XSL parameters can be appended. Any parameter that you want to append must follow the `-D` model of the above parameters. For example, you can append the `html.stylesheet` parameter in the following form:

```
-Dhtml.stylesheet=/path/to/directory/of/stylesheet.css
```

Configuring the Comment Database for DocBook WebHelp Classic with Feedback

If you run the ***webhelp-feedback*** transformation using the WebHelp plugin, you need to configure the database that holds the user comments. The instructions for configuring the database are presented in the `installation.html` file, located at: `[OUTPUT_DIR]/oxygen-webhelp/resources/installation.html`. The `installation.html` file is created by the transformation process.

Working with XPath Expressions

10

Topics:

- *XPath Toolbar*
- *XPath Builder View*
- *XPath Expression Results View*
- *XPath and XML Catalogs*
- *XPath Prefix Mapping*

XPath is a language for addressing specific parts of an XML document. XPath, such as the Document Object Model (DOM), models an XML document as a tree of nodes. An XPath expression is a mechanism for navigating through and selecting nodes from the XML document. An XPath expression is, in a way, analogous to an SQL query used to select records from a database.

There are various types of nodes, including element nodes, attribute nodes, and text nodes. XPath defines a way to compute a string-value for each type of node.

XPath defines a library of standard functions for working with strings, numbers and boolean expressions.

Examples:

- `child::*` - Selects all children of the root node.
- `.//name` - Selects all elements having the name "name", descendants of the current node.
- `/catalog/cd[price>10.80]` - Selects all the `cd` elements that have a price element with a value larger than 10.80.

To find out more about XPath, go to *http://www.w3.org/TR/xpath*.

Related Information:

Content Completion in XPath Expressions on page 515

Finding and Replacing Text in Multiple Files on page 476

Find/Replace Dialog Box on page 472

XPath Toolbar

XPath is a query language for selecting nodes from an XML document. To use XPath expressions effectively, you need a good understanding of *the XPath Core Function Library*.

XPath Toolbar

Oxygen XML Editor provides an XPath toolbar to let you query XML documents fast and easy using XPath expressions.

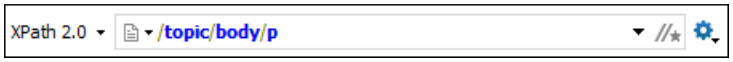


Figure 495: XPath Toolbar

The XPath toolbar includes the following features:

XPath version chooser drop-down menu

You can choose the XPath version from the drop-down menu available in the left side of the toolbar. Available options include **XPath 1.0**, **XPath 2.0**, **XPath 2.0 SA**, **XPath 3.0**, **XPath 3.0 SA**.

Note: The results returned by XPath 2.0 SA and XPath 3.0 SA have a location limited to the line number of the start element (there are no column information and no end specified).

Warning: Oxygen XML Editor uses Saxon to execute XPath 3.0 expressions, but implements a part of the 3.0 functions. When using a function that is not implemented, Oxygen XML Editor can return a compilation error.

XPath scope menu

Oxygen XML Editor allows you to define a scope for which the XPath operation will be executed. You can choose where the XPath expression will be executed:

- **Current file** - Current selected file only.
- **Project** - All the files in the project.
- **Selected project resources** - The files selected in the project.
- **All opened files** - All files opened in the application.
- **Current DITA Map hierarchy** - All resources referenced in the currently selected *DITA map*, opened in the ***DITA Maps Manager*** *view*.
- **Opened archive** - Files open in the ***Archive Browser*** *view*.
- **Working sets** - The selected *working sets*.

At the bottom of the scope menu the following scope configuration actions are available:

- **Configure XPath working sets** - Allows you to configure and manage collections of files and folders, encapsulated in logical containers called *working sets*.
- **XPath file filter** - You can filter the files from the selected scope for which the XPath expression will be executed. By default, the XPath expression will be executed only on XML files, but you can also define a set of patterns that will filter out files from the current scope. If you select the **Include archive** option, the XPath expression will be also executed on the files in any archive (including EPUB and DocX) found at the current scope.

History drop-down list

The XPath combo box keeps a history of the last 15 expressions that were used so that you can easily choose them again.

Switch to XPath Builder View

Opens the ***XPath Builder*** *view*.

Settings menu

The following actions are available in this drop-down menu:

- **Update on cursor move** - When selected and you navigate through a document, the XPath expression corresponding to the XML node at the current cursor position is displayed.
- **Evaluate as you type** - When you select this option, the XPath expression you are composing is evaluated in real time.

  Note: The **Evaluate as you type** option and the automatic validation are disabled when you edit *huge documents* or when the scope is other than **Current file**.
- **Options** - Opens the Preferences page of the currently selected processing engine.

Note: During the execution of an XPath expression, the XPath toolbar displays a **Stop** button. Press this button to stop the XPath execution.

When you type expressions longer than 60 characters, a dialog box opens that offers you the possibility to switch to the ***XPath Builder*** *view*.

Related Information:

XPath Expression Results View on page 1061

XPath Builder View

The **XPath/XQuery Builder** view allows you to compose complex XPath expressions and execute them over the currently edited XML document. For XPath 2.0 / 3.0, you can use the `doc( )` function to specify the source file for which the expressions are executed. When you connect to a database, the expressions are executed over that database. If you are using the **XPath/XQuery Builder** view and the current file is an XSLT document, Oxygen XML Editor executes the expressions over the XML document in the associated scenario.

If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu. You can also open it simply by pressing the **Switch to XPath Builder View** button that is located on the *XPath toolbar*.

The upper part of the view contains the following actions:

XPath version chooser drop-down menu

A drop-down menu that allows you to select the type of the expression you want to execute. You can choose between:

- XPath 1.0 (Xerces-driven)
- XPath 2.0, XPath 2.0SA, XPath 3.0, XPath 3.0SA, XQuery 1.0, XQuery 3.0, Saxon-HE XQuery, Saxon-PE XQuery, or Saxon-EE XQuery (all of them are Saxon-driven)
- Custom connection to XML databases that can execute XQuery expressions

  Note: The results returned by XPath 2.0 SA and XPath 3.0 SA have a location limited to the line number of the start element (there are no column information and no end specified).

  Note: Oxygen XML Editor uses Saxon to execute XPath 3.0 expressions. Since Saxon implements a part of the 3.0 functions, when using a function that is not implemented, Oxygen XML Editor returns a compilation error.

Execute XPath button

Use this button to start the execution of the XPath or XQuery expression you are editing. The result of the execution is displayed in the ***Results** view*.

Favorites button

Allows you to save certain expressions that you can later reuse. To add an expression as a favorite, press the star button and enter a name for it. The star turns yellow to confirm that the expression was saved. Expand the drop-down menu next to the star button to see all your favorites. Oxygen XML Editor automatically groups favorites in folders named after the method of execution.

History drop-down menu

Keeps a list of the last 15 executed XPath expressions. Use the **Clear history** action from the bottom of the list to remove them.

Settings drop-down menu

Contains the following three options:

- **Update on cursor move** - When selected and you navigate through a document, the XPath expression corresponding to the XML node at the current cursor position is displayed.
- **Evaluate as you type** - When you select this option, the XPath expression you are composing is evaluated in real time.

  Note: The **Evaluate as you type** option and the automatic validation are disabled when you edit *huge documents* or when the scope is other than **Current file**.
- **Options** - Opens the Preferences page of the currently selected processing engine.

XPath scope menu

Oxygen XML Editor allows you to define a scope for which the XPath operation will be executed. You can choose where the XPath expression will be executed:

- **Current file** - Current selected file only.

- **Project** - All the files in the project.
- **Selected project resources** - The files selected in the project.
- **All opened files** - All files opened in the application.
- **Current DITA Map hierarchy** - All resources referenced in the currently selected *DITA map*, opened in the ***DITA Maps Manager*** *view*.
- **Opened archive** - Files open in the ***Archive Browser*** *view*.
- **Working sets** - The selected *working sets*.

At the bottom of the scope menu the following scope configuration actions are available:

- **Configure XPath working sets** - Allows you to configure and manage collections of files and folders, encapsulated in logical containers called *working sets*.
- **XPath file filter** - You can filter the files from the selected scope for which the XPath expression will be executed. By default, the XPath expression will be executed only on XML files, but you can also define a set of patterns that will filter out files from the current scope. If you select the **Include archive** option, the XPath expression will be also executed on the files in any archive (including EPUB and DocX) found at the current scope.

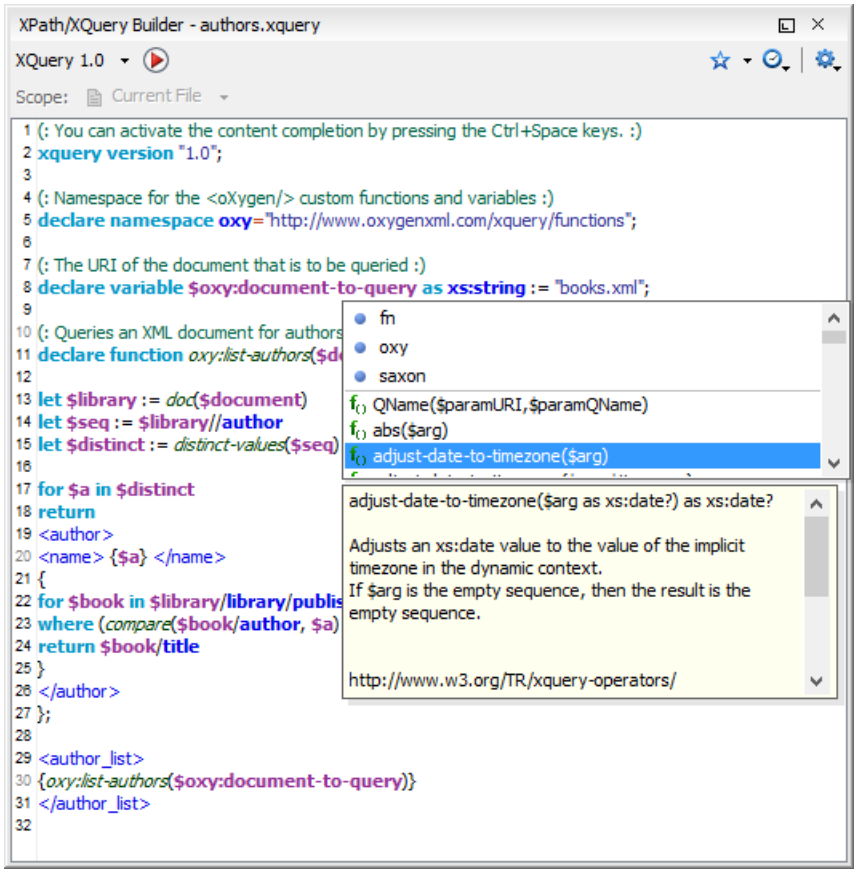


Figure 496: XPath/XQuery Builder View

While you edit an XPath or XQuery expression, Oxygen XML Editor assists you with the following features:

- *Content Completion Assistant* - It offers context-dependent proposals and takes into account the cursor position in the document you are editing. The set of functions proposed by the *Content Completion Assistant* also depends on the engine version. Select the engine version from the drop-down menu available in the toolbar.

- Syntax Highlighting - Allows you to identify the components of an expression. To customize the colors of the components of the expression, *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to ***Editor** > **Syntax Highlight***.
- Automatic validation of the expression as you type.

  Note: When you type invalid syntax a red serrated line underlines the invalid fragments.
- Function signature and documentation balloon, when the cursor is located inside a function.

The usual edit actions (**Cut**, **Copy**, **Paste**, **Select All**, **Undo**, **Redo**) are available in the contextual menu of the top editable part of the view.

Related Information:

XPath Expression Results View on page 1061

XPath Expression Results View

When you run an XPath expression, Oxygen XML Editor displays the results of its execution in the ***Results** view*.

This view contains the following columns:

- **Description** - The result thatOxygen XML Editor displays when you run an XPath expression.
- **XPath location** - The path to the matched node.
- **Resource** - The name of the document that you run the XPath expression on.
- **System ID** - The path to the document itself.
- **Location** - The location of the result in the document.

To arrange the results depending on a column, click its header. To group the results by their resource, or by their system ID, right-click the header of any column in the **Results** view and select **Group by "Resource"** or **Group by "System ID"**. If no information regarding location is available, Oxygen XML Editor displays **Not available** in the **Location** column. Oxygen XML Editor displays the results in a valid XPath expression format.

```
- /node[value]/node[value]/node[value] -
```

The **Results** view also includes various toolbar and contextual menu actions. For more information, see *Results View* on page 203.

Example:

The following snippets are taken from a DocBook book based on the DocBook XML DTD. The book contains a number of chapters. To return all the chapter nodes of the book, enter `//chapter` in the XPath expression field and press **(Enter)**. This action returns all the `chapter` nodes of the DocBook book in the **Results View**. Click a record in the **Results View** to locate and highlight its corresponding chapter element and all its children nodes in the document you are editing.

To find all `example` nodes contained in the `sect2` nodes of a DocBook XML document, use the following XPath expression: `//chapter/sect1/sect2/example`. Oxygen XML Editor adds a result in the **Results View** for each `example` node found in any `sect2` node.

For example, if the result of the above XPath expression is:

```
- /chapter[1]/sect1[3]/sect2[7]/example[1]
```

it means that in the edited file, the `example` node is located in the first chapter, third section level one, seventh section level 2.

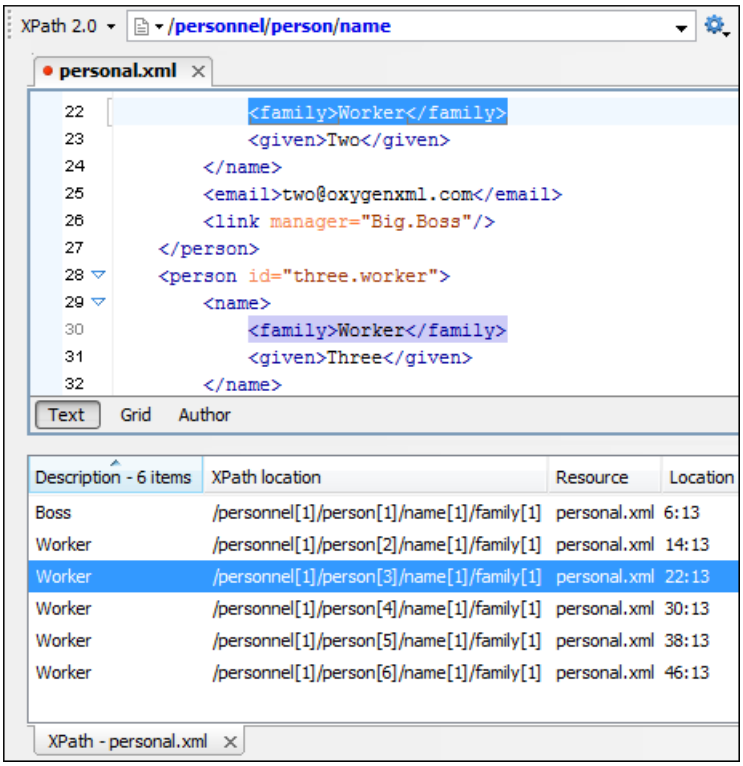


Figure 497: XPath Results Highlighted in Editor Panel with Character Precision

XPath and XML Catalogs

The evaluation of the XPath expression tries to resolve the locations of documents referenced in the expression through *XML Catalogs*. These catalogs are configured in the ***XML Catalog*** *preferences* pages and the ***XML Parser*** *preferences*.

Example:

As an example, consider the evaluation of the `collection(URIofCollection)` function (XPath 2.0). To resolve the references from the files returned by the `collection()` function with an *XML catalog*, specify the class name of the catalog-enabled parser for parsing these collection files. The class name is `ro.sync.xml.parser.CatalogEnabledXMLReader`. Specify it as it follows:

```
let $docs := collection(iri-to-uri(
   "file:///D:/temp/test/XQuery-catalog/mydocsdir?recurse=yes;select=*.xml;
   parser=ro.sync.xml.parser.CatalogEnabledXMLReader"))
```

XPath Prefix Mapping

To define default mappings between prefixes (that you can use in the *XPath toolbar*) and namespace URIs go to the ***XPath*** *preferences page* and enter the mappings in the **Default prefix-namespace mappings** table. The same preferences panel allows you to configure the default namespace used in XPath 2.0 expressions.

Important: If you define a default namespace, Oxygen XML Editor binds this namespace to the first free prefix from the list: `default`, `default1`, `default2`, and so on. For example, if you define the default namespace `xmlns="something"` and the prefix `default` is not associated with another namespace, you can match tags without prefix in an XPath expression typed in the XPath toolbar by using the prefix `default`. To find all the `level` elements when you define a default namespace in the root element, use this expression: `//default:level` in the XPath toolbar.

Working with Archives

11

Topics:

- *Browsing Archives*
- *Working with Archive Files*

Oxygen XML Editor includes a useful ***Archive Browser** view* that offers the means to work with files directly from various types of archives (for example, opening and saving files directly in archives, or browsing and modifying archive structures). The archive support is available for all ZIP-type archives, including:

- ZIP archives
- EPUB books
- *JAR archives*
- Office Open XML (OOXML) files
- Open Document Format (ODF) files
- *IDML files*

You can transform, validate, and perform many other operations on files directly from an archive. For instance, you can transform, or validate files directly from OOXML or ODF packages, and the structure and content of the ZIP archives can be opened, edited, and saved, similar to any other ZIP archive browsing tool. Also, when browsing for a URL in various dialog boxes, you can use the **Browse for archived file** action to browse and select files from a particular archive.

Browsing Archives

Oxygen XML Editor includes a helper view called the **Archive Browser** that allows you to view the contents and structure of an archive, and it offers a variety of toolbar and contextual menu actions. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

To open an archive in the **Archive Browser** view, use one of the following methods:

- Open an archive from the ***Project** view*.
- Select an archive in one of the file chooser dialog boxes in Oxygen XML Editor (such as the **Open** dialog box).
- Drag an archive from a system file explorer and drop it in the **Archives Browser** view.

When displaying an archive, the **Archive Browser** view locks the archive file. It is then automatically unlocked when the **Archive Browser** view is closed.

Tip: If a file is not recognized by Oxygen XML Editor as a supported archive type, you can add it from the ***Archive** preferences page*.

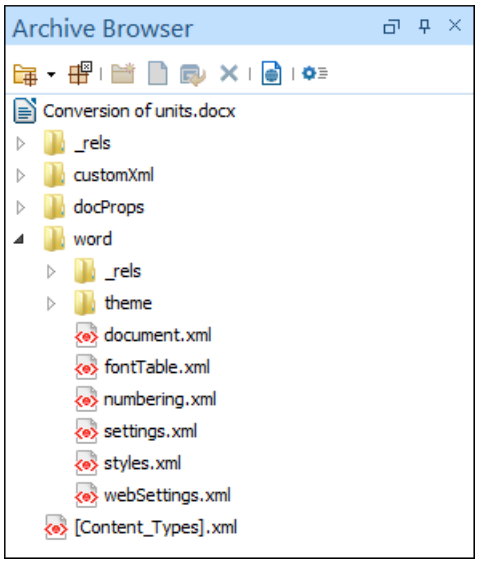


Figure 498: Archive Browser

Archive Browser Toolbar Actions

The following actions are available on the **Archive Browser** toolbar:

Open Archive menu

Provides access to the **Open Archive** action that opens a new archive in the browser. If the extension is not known as an archive extension, you will be directed to the ***Archive*** *preferences page* to add a new extension. The sub-menu keeps a list of recently open archive files and a **Clear history** action that allows you to delete the list.

Close

Closes the browsed archive and unlocks the archive file.

Validate (available for EPUB archives only)

Checks the EPUB archive to see if its content and structure is valid.

New folder

Creates a folder as child of the selected folder in the browsed archive.

New file

Creates a file as child of the selected folder in the browsed archive.

Add files

Adds existing files as children of the selected folder in the browsed archive.

Note: You can also add files in the archive by dragging them from the file browser or the ***Project*** *view* and dropping them in the **Archive Browser** view.

Delete

Deletes the selected resource in the browsed archive.

Open in System Application

Opens the selected resource in the default system application that is associated with that type of file.

Archive Options

Opens the ***Archive*** *preferences page*.

Archive Browser Contextual Menu Actions

The following additional actions are available from the contextual menu for resources in the **Archive Browser** view:

Open

Opens a resource from the archive in the editor.

Extract

Extracts a resource from the archive in a specified folder.

New folder

Creates a folder as child of the selected folder in the browsed archive.

New file

Creates a file as child of the selected folder in the browsed archive.

Add files

Adds existing files as children of the selected folder in the browsed archive.

Note: On OS X, the **Add file** action is also available and it allows you to add one file at a time.

Rename

Renames a resource in the archive.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to search for and replace specific pieces of text inside the archive.

Cut

Cuts the selected archive resource.

Copy

Copies the selected archive resource.

Paste

Pastes a file or folder into the archive.

Delete

Removes a file or folder from archive.

Preview

Previews an image contained in the archive. See the *Image Preview* topic for more details.

Copy location

Copies the URL location of the selected resource.

Refresh

Refreshes the selected resource.

Properties

Shows the properties of the selected resource.

Working with Archive Files

Oxygen XML Editor includes support for working with various types of archives, including the following:

- **EPUB** - An e-book file format that can be used on many types of devices, such as smart phones, tablets, e-readers, or computers.
- **OOXML** - An XML-based file format for representing spreadsheets, charts, presentations, and word processing documents.

- **ODF** - An free and open-source XML-based file format for electronic office documents, such as spreadsheets, charts, presentations, and word processing documents.

When these types of files are opened in the ***Archive Browser*** *view*, their internal components are expanded:

- Document content (XHTML and image files).
- Packaging files.
- Container files.

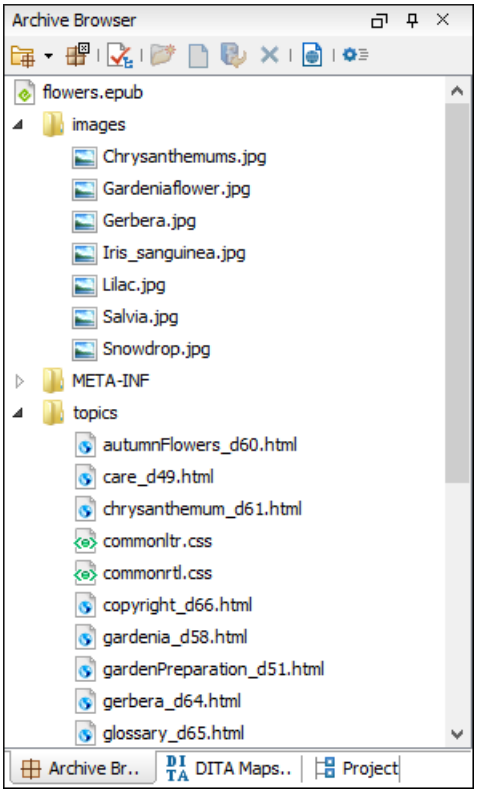


Figure 499: EPUB File Displayed in the Archive Browser View

When an archive is expanded in the ***Archive Browser*** *view*, you can add or delete files that compose the archive structure. All changes made to the structure of an archive are saved immediately. You can open files from within the archive to *edit them in the main editing pane and save changes* back to the archive. You can also use the **Open in System Application** action to open the archive in the default system application that is associated with that type of file.

EPUB-Specific Validation

When working with EPUB archives, the ***Archive Browser*** includes a **Validate** action on the toolbar that checks the EPUB archive to make sure the structure and content are valid. Oxygen XML Editor uses the open-source *EpubCheck* validator to perform the validation. This validator detects many types of errors, including OCF container structure, OPF and OPS mark-up, as well as internal reference consistency.

To watch our video demonstration about EPUB support in Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/Epub.html*.

Related Information:

The Archive Browser View on page 1063
EPUB Document Type (Framework) on page 852

Creating an Archive

To create an archive from scratch, follow these steps:

1. Go to **File** > **New** or click **New** on the main toolbar.
2. Choose your particular type of archive template. For example, select one of the **ODF**, **OOXML**, or **EPUB** templates.
3. Click **Create** and choose the name and location of the file.
4. Click **Save**.
   A skeleton archive is saved on disk and opened in the ***Archive Browser*** *view*.

   Tip: Use toolbar and contextual menu actions to edit, add, and remove resources from the archive. For EPUB archives, you can use the **Validate** action to verify the integrity of the EPUB archive.

Editing and Saving Files Inside an Archive

You can open files directly from an archive in the ***Archive Browser*** *view* and then edit them in the main editor pane. To open a file, simply double-click it or select **Open** from the contextual menu.

When saving the file back to the archive, you are prompted to choose if you want the application to make a backup copy of the archive before saving the new content. If you choose **Never ask me again**, you will not be asked again to make backup copies. You can re-enable the pop-up message from the *Messages preferences page*.

Migrating Archives to DITA or TEI

Certain types of archives can be converted to DITA or TEI. For example, OOXML (Office Open XML) archive files with the DOCX file extension can be migrated to DITA or TEI.

To migrate DOCX files to DITA or TEI, follow these steps:

1. Open and expand the archive in the ***Archive Browser***.
2. Open the **document.xml** file contained in the archive.
3. Run one of the following predefined transformation scenarios:
   a) **DOCX DITA** to migrate to DITA.
   b) **DOCX TEI P5** to migrate to TEI.
4. You may need to do some manual reconfiguring to map DOCX styles to DITA or TEI content.

   Tip: Oxygen XML Editor also includes a predefined transformation scenario called **ODT TEI P5** for converting ODF archive files with the ODT file extension to TEI and a similar process can be used to migrate ODT files to TEI.

Databases and CMS Integration

12

Topics:

- *Working with Databases*
- *Content Management System (CMS) Integration*

Oxygen XML Editor provides support for connecting and integrating with various databases and content management systems. This section includes information about the database-related features in Oxygen XML Editor. It explains how to connect with the supported databases, presents the actions that are available for each type, and includes information about SharePoint and Documentum (CMS) integration.

Working with Databases

XML is a storage and interchange format for structured data and is supported by all major database systems. Oxygen XML Editor offers the means for managing the interaction with some of the most commonly used databases (both *Relational* and *Native XML* databases). Through this interaction, Oxygen XML Editor helps users with browsing, content editing, importing from databases, using XQuery with databases, SQL execution, and generating XML Schema from a database structure.

The types of connections that are supported in Oxygen XML Editor include:

- *IBM DB2*
- *Microsoft SQL Server*
- *Oracle Database*
- *PostgreSQL*
- *Berkeley DB XML*
- *eXist*
- *MarkLogic*
- *Documentum xDB (X-Hive/DB)*
- *MySQL*
- *Generic JDBC*
- *JDBC-ODBC*
- *BaseX*
- *WebDAV*
- *Microsoft SharePoint*

Related Information:

Integration with Microsoft SharePoint on page 1126

Data Source Explorer View

The **Data Source Explorer** view displays your database connections. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

You can connect to a database simply by expanding the connection node (click the connection) . The database structure can be expanded to resource level, or even all the way to column level for tables inside relational databases. Oxygen XML Editor supports multiple simultaneous database connections and the connection tree in the **Data Source Explorer** view provides an easy method for browsing them.

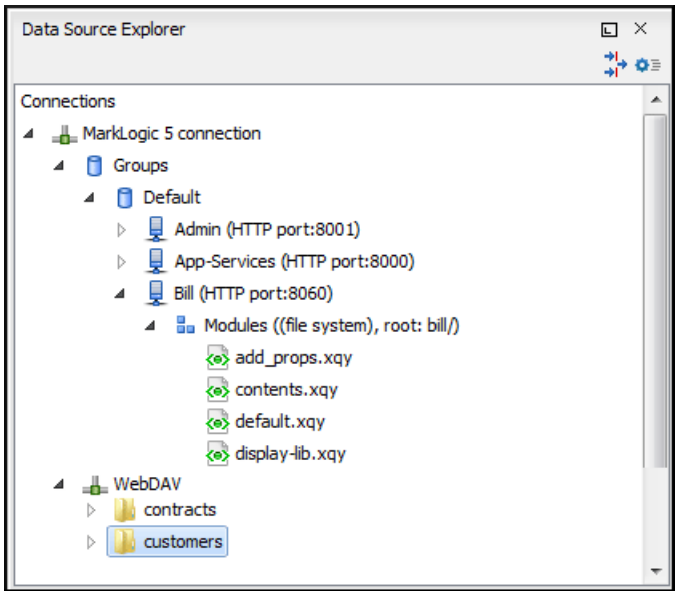


Figure 500: Data Source Explorer View

The objects (nodes) that are displayed in the **Data Source Explorer** view depend on the connection type and structure of the database. Various contextual menu actions are available for each hierarchical level and for some connections you can add or move resources in a container by simply dragging them from the ***Project*** *view*, a file browsing application, or another database.

Toolbar Actions

The following actions are available in the toolbar of this view:

Filters

Opens the ***Data Sources / Table Filters*** *preferences page*, allowing you to decide which table types are displayed in the **Data Source Explorer** view.

Configure Database Sources

Opens the ***Data Sources*** *preferences page* where you can configure both data sources and connections.

Database-Specific Contextual Menu Actions

Each specific type of database will also include its own specific contextual menu actions in the **Data Source Explorer** view. The actions depend on the type of database, the type of node, or the hierarchical level of the node in which the contextual menu is invoked.

For more information on the specific actions that are available, see the topics in this section for each specific type of database.

Related Information:

Data Sources Preferences on page 138

Table Explorer View

Relational databases tables in the ***Data Source Explorer*** *view* can be displayed and edited in the **Table Explorer** view by selecting the **Edit** action from the contextual menu of a **Table** node or by double-clicking one of its fields. To modify the content of a cell, double-click it and start typing. When editing is complete, Oxygen XML Editor attempts to update the database with the new cell content.

Table Explorer

	ID [SMALLINT]	NAME [VARCHAR]	DEPT [SMALLINT]	JOB [CHAR]	YEARS [SMALLINT]	SALARY [DECIMAL]	COMM [DECIMAL]
0	10	Sanders	20	Mgr	7	98357.50	*(null)*
1	20	Pernal	20	Sales	8	78171.25	612.45
2	30	Marenghi	38	Mgr	5	77506.75	*(null)*
3	40	O'Brien	38	Sales	6	78006.00	846.55
4	50	Hanes	15	Mgr	10	80659.80	*(null)*
5	60	Quigley	38	Sales	*(null)*	66808.30	650.25
6	70	Rothman	15	Sales	7	76502.83	1152.00
7	80	James	20	Clerk	*(null)*	43504.60	128.20
8	90	Koonitz	42	Sales	6	38001.75	1386.70
9	100	Plotz	42	Mgr	7	78352.80	*(null)*

Figure 501: Table Explorer View

You can sort the content of a table by one of its columns by clicking its column header.

Note the following:

- The first column is an index (not part of the table structure)
- Every column header contains the field name and its data type
- The primary key columns are marked with this symbol:
- Multiple tables are presented in a tabbed manner

For performance issues, you can set the maximum number of cells that are displayed in the **Table Explorer** view (using the ***Limit the number of cells*** *option in the* ***Data Sources*** *Preferences page*). If a table that has more cells than the value set in the options is displayed in the **Table Explorer** view, a warning dialog box informs you that the table is only partially shown.

You are notified if the value you have entered in a cell is not valid (and thus cannot be updated).

- If the content of the edited cell does not belong to the data type of the column, the cell is marked by a red square and remains in an editing state until a correct value is inserted. For example, in the following figure `propID` contains `LONG` values. If a character or string is inserted, the cell will look like this:

Table Explorer

	propID [LONG]	setting [VARCHAR]	value [VARCHAR]
0	8	imagePath	/home/bogdan/projects/camera/public_html/Camera/img
1	abc	maxBadPass	3
2	3	pageRefresh	5
3	4	lastUpdate	June 7th, 2002
4	5	adminEmail	bogdan@oxygenxml.com
5	7	moviePath	/home/bogdan/projects/camera/public_html/Camera/movie
6	6	timeoutReceivingData	300
7	13	firmwareUploadPort	3002
8	11	internalFirmwareUploadHost	10.0.0.16
9	9	uploadPath	/home/bogdan/projects/camera/public_html/Camera/tsk
10	10	imageStreamPath	/home/bogdan/projects/camera/public_html/Camera/imgStream

testcase | user | settings

Figure 502: Cell Containing an Invalid Value

- If the constraints of the database are not met (for instance, primary key constraints), an information dialog box will appear, notifying you of the reason the database has not been updated. For example, in the table below, trying to set the second record in the primary key `propID` column to 8, results in a duplicate entry error since that value has already been used in the first record:

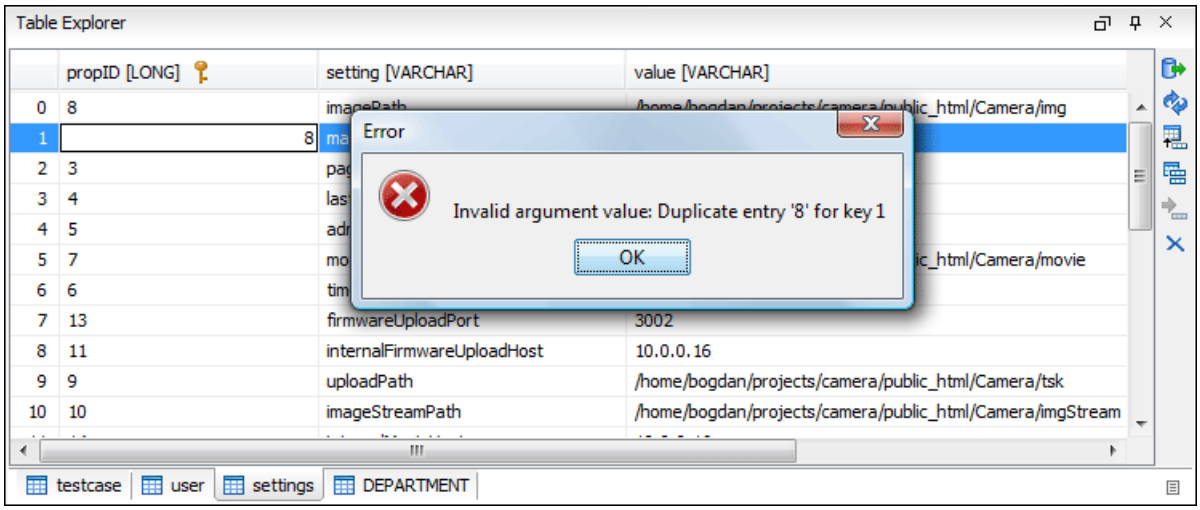


Figure 503: Duplicate Entry for Primary Key

Table Explorer Contextual Menu Actions

Common editing actions (**Cut**, **Copy**, **Paste**, **Select All**, **Undo**, **Redo**) are available in the contextual menu of an edited cell.

The contextual menu, available on every cell in the **Table Explorer** view, also includes the following actions:

Set NULL

Sets the content of the cell to **null**. This action is not available for columns that cannot have a value of **null**.

Insert row

Inserts an empty row in the table.

Duplicate row

Makes a copy of the selected row and adds it in the **Table Explorer** view. Note that the new row will not be inserted in the database table until all conflicts are resolved.

Commit row

Commits the selected row.

Delete row

Deletes the selected row.

Copy

Copies the content of the cell.

Paste

Pastes copied content into the selected cell.

Table Explorer Toolbar Actions

The toolbar of the **Table Explorer** view also includes the following actions:

Export to XML

Opens the **Export Criteria** dialog box (a thorough description of this dialog box can be found in the *Import from database* chapter) .

Refresh

Performs a refresh for the sub-tree of the selected node.

Insert row

Inserts an empty row in the table.

Duplicate row

Makes a copy of the selected row and adds it in the **Table Explorer** view. Note that the new row will not be inserted in the database table until all conflicts are resolved.

Commit row

Commits the selected row.

Delete row

Deletes the selected row.

Related Information:

Data Source Explorer View on page 1068

Database Connection Support

Oxygen XML Editor offers support for a variety of *Relational* and *Native XML* database connections. The database drivers and connections for various types of database are configured in the ***Data Sources** preferences page* and once configured, the database connections can be viewed and managed in the ***Data Source Explorer** view*. Oxygen XML Editor also includes a ***Database** perspective* that helps you to manage databases.

The database support in Oxygen XML Editor offers a variety of capabilities, including:

- Browsing the structure of databases in the ***Data Source Explorer** view*.
- Viewing relational tables in the ***Table Explorer** view*.
- Executing SQL queries against databases.
- Calling stored procedures with input and output parameters.
- XQuery execution with databases.
- Exporting data from databases to XML.

Relational Database Support

Relational databases use a relational model and are based on tables linked by a common key. Oxygen XML Editor offers support for the most commonly used relational databases, including:

- IBM DB2
- Oracle 11g
- Microsoft SQL Server
- PostgreSQL
- MySQL

Oxygen XML Editor also offers generic support (table browsing and execution of SQL queries) for any JDBC-compliant database (for example, *MariaDB*).

To watch our video demonstration about the integration between the relational databases and Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/Author_Database_Integration.html*.

Native XML Database Support

Native XML databases have an XML-based internal model and their fundamental unit of storage is XML. They use XML as an interface to specify documents as tree structured data that may contain unstructured text, but on disk the data is stored as optimized binary files. This makes query and retrieval processes faster. Oxygen XML Editor offers support for the most commonly used native XML databases, including:

- Berkeley DB XML
- eXist
- MarkLogic
- Documentum xDB (X-Hive/DB) 10
- Oracle XML DB
- Base X

To watch our video demonstration about the integration between the native XML databases and Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/Author_Database_XML_Native.html*.

Related Information:

WebDAV Connections on page 1111

Integration with Microsoft SharePoint on page 1126

IBM DB2 Database Connections

Oxygen XML Editor includes support for IBM DB2 database connections. Oxygen XML Editor allows you to browse the structure of an IBM DB2 database in the ***Data Source Explorer** view*, open tables in the ***Table Explorer** view*, and perform various operations on the resources in the repository.

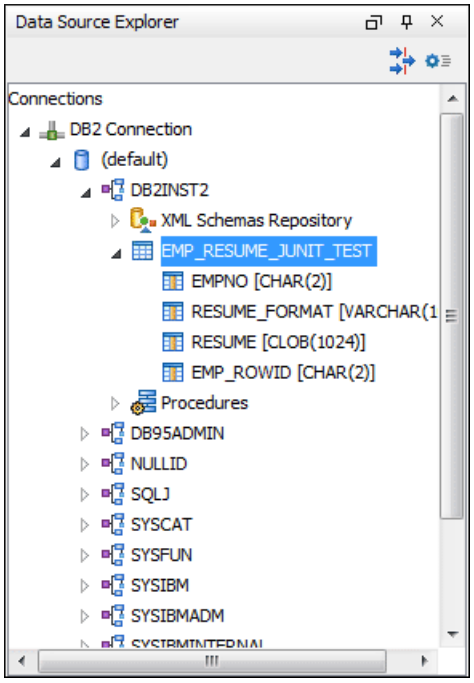


Figure 504: IBM DB2 Database Connection

Configuring an IBM DB2 Database Connection

To configure the support for the IBM DB2 database, follow this procedure:

1. Go to the *IBM website* and in the *DB2 Clients and Development Tools* category select the *DB2 Driver for JDBC and SQLJ* download link. Fill out the download form and download the zip file. Unzip the zip file and use the `db2jcc.jar` and `db2jcc_license_cu.jar` files in Oxygen XML Editor for *configuring a DB2 data source*.
2. *Configure IBM DB2 Data Source drivers*.
3. *Configure an IBM DB2 Server Connection*.
4. To view your connection, go to the ***Data Source Explorer** view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

How to Configure IBM DB2 Data Source Drivers

Available in the Enterprise edition only.

To configure a data source for connecting to an IBM DB2 server, follow these steps:

1. Go to the *IBM website* and in the *DB2 Clients and Development Tools* category select the *DB2 Driver for JDBC and SQLJ* download link. Fill out the download form and download the zip file.
2. Unzip the downloaded archive.
3. *Open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Data Sources**.
4. Click the **New** button in the **Data Sources** panel.

   The dialog box for configuring a data source is opened.

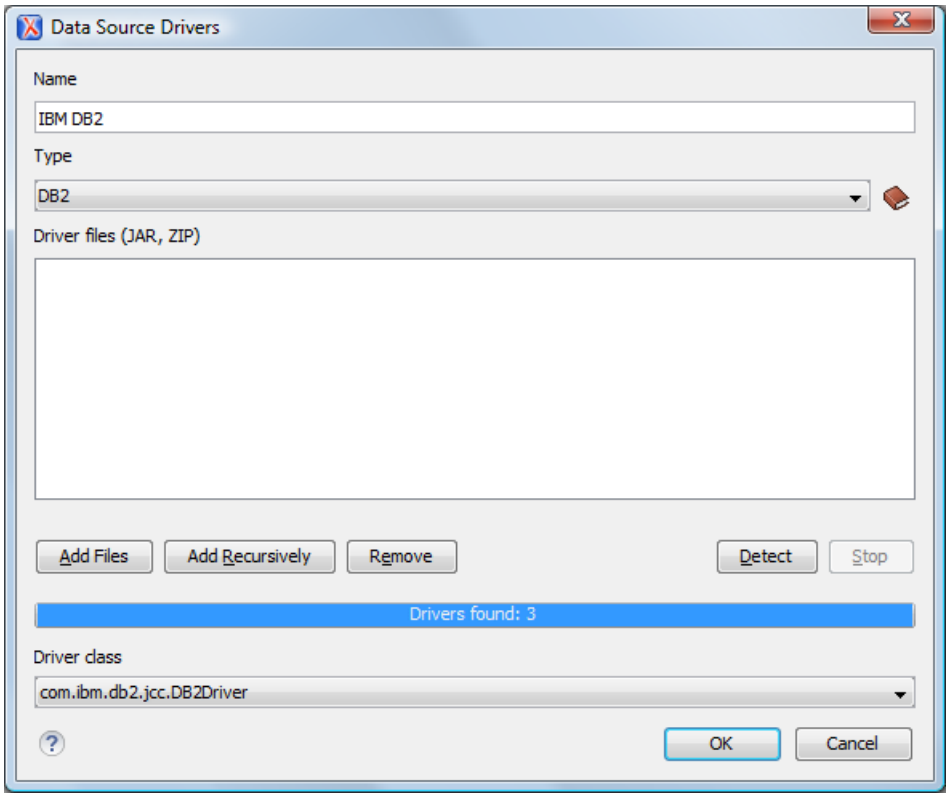


Figure 505: Data Source Drivers Configuration Dialog Box

5. Enter a unique name for the data source.
6. Select *DB2* in the driver **Type** drop-down menu.
7. Click the **Add Files** button and select the IBM DB2 driver files from the archive that you downloaded and unzipped.

   The IBM DB2 driver files are:

   - `db2jcc.jar`
   - `db2jcc_license_cisuz.jar`
   - `db2jcc_license_cu.jar`
8. Select the most appropriate **Driver class**.
9. Click the **OK** button to finish the data source configuration.
10. Continue on to *configure your IBM DB2 connection*.

    To watch our video demonstration about running XQuery against an IBM DB2 Pure XML database, go to *https://www.oxygenxml.com/demo/DB2.html*.

How to Configure an IBM DB2 Connection

The support to create an IBM DB2 connection is available in the Enterprise edition only.

To configure a connection to an IBM DB2 server, follow these steps:

1. *Open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Data Sources**.
2. Click the **New** button in the **Connections** panel.

   The dialog box for configuring a database connection is displayed.

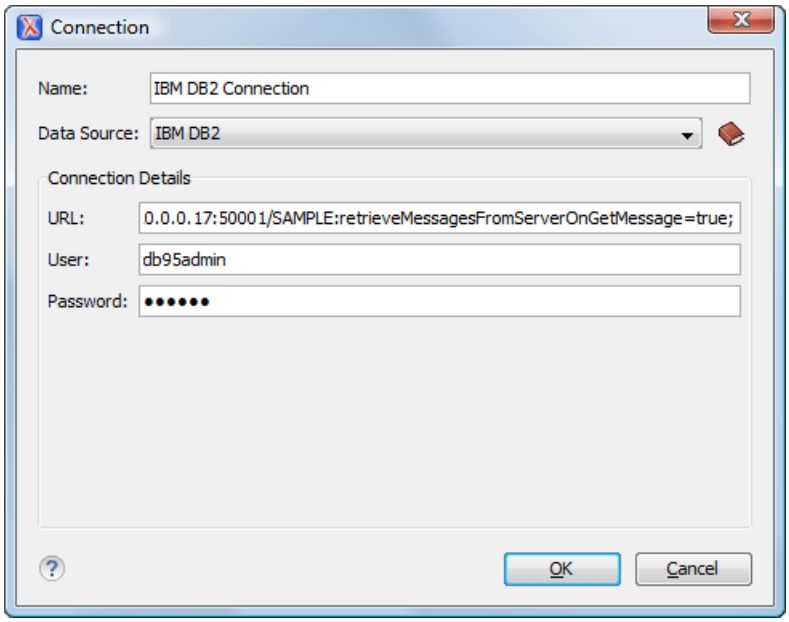


Figure 506: Connection Configuration Dialog Box

3. Enter a unique name for the connection.
4. Select an *IBM DB2* data source in the **Data Source** drop-down menu.
5. Enter the connection details.
   a) Enter the URL to the installed IBM DB2 engine.
   b) Enter the user name to access the IBM DB2 engine.
   c) Enter the password to access the IBM DB2 engine.
6. Click the **OK** button to finish the connection configuration.
7. To view your connection, go to the ***Data Source Explorer*** *view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

To watch our video demonstration about running XQuery against an IBM DB2 Pure XML database, go to *https://www.oxygenxml.com/demo/DB2.html*.

IBM DB2 Contextual Menu Actions

General Contextual Menu Actions

For relational databases, the following general actions are available in the contextual menu of the ***Data Source Explorer*** *view*, depending on the node in which it is invoked:

Refresh

Performs a refresh on the selected node.

Disconnect (available on Connection nodes)

Closes the current database connection. If a table is already open, you are warned to close it before proceeding.

Configure Database Sources (available on Connection nodes)

Opens the ***Data Sources*** *preferences page* where you can configure both data sources and connections.

Edit (available on Table nodes)

Opens the selected table in the ***Table Explorer*** *view*.

Export to XML (available on Table nodes)

Opens the **Export Criteria** dialog box (a thorough description of this dialog box can be found in the *Import from Database* chapter).

Database-Specific Contextual Menu Actions

In addition to the general contextual menu actions in the ***Data Source Explorer*** *view*, the various nodes in IBM DB2 connections include the following additional contextual menu actions:

XML Schema Repository Level Nodes

Register

Opens a dialog box for adding a new schema file in the DB XML repository. In this dialog box, you enter a collection name and the necessary schema files. Schema dependencies management can be done by using the **Add** and **Remove** buttons.

Schema Level Nodes

Unregister

Removes the selected schema from the XML Schema Repository.

View

Opens the selected schema in Oxygen XML Editor.

Microsoft SQL Server Database Connections

Oxygen XML Editor includes support for Microsoft SQL Server database connections. Oxygen XML Editor allows you to browse the structure of a SQL Server database in the ***Data Source Explorer*** *view*, open tables in the ***Table Explorer*** *view*, and perform various operations on the resources in the repository.

Configuring a Microsoft SQL Server Connection

To configure the support for a Microsoft SQL Server database, follow this procedure:

1. Download the appropriate MS SQL JDBC driver from the Microsoft website. For SQL Server 2008 R2 and older go to *http://www.microsoft.com/en-us/download/details.aspx?id=21599*. For SQL Server 2012 and 2014 go to *http://www.microsoft.com/en-us/download/details.aspx?id=11774*.
2. *Configure MS SQL Server Data Source drivers*.
3. *Configure a MS SQL Server Connection*.
4. To view your connection, go to the ***Data Source Explorer*** *view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

How to Configure Microsoft SQL Server Data Source Drivers

Available in the Enterprise edition only.

To configure a data source for connecting to a Microsoft SQL server, follow these steps:

1. Download the appropriate MS SQL JDBC driver from the Microsoft website. For SQL Server 2008 R2 and older, go to *http://www.microsoft.com/en-us/download/details.aspx?id=21599*. For SQL Server 2012 and 2014, go to *http://www.microsoft.com/en-us/download/details.aspx?id=11774*.
2. *Open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Data Sources**.
3. Click the **New** button in the **Data Sources** panel.

   The dialog box for configuring a data source is opened.

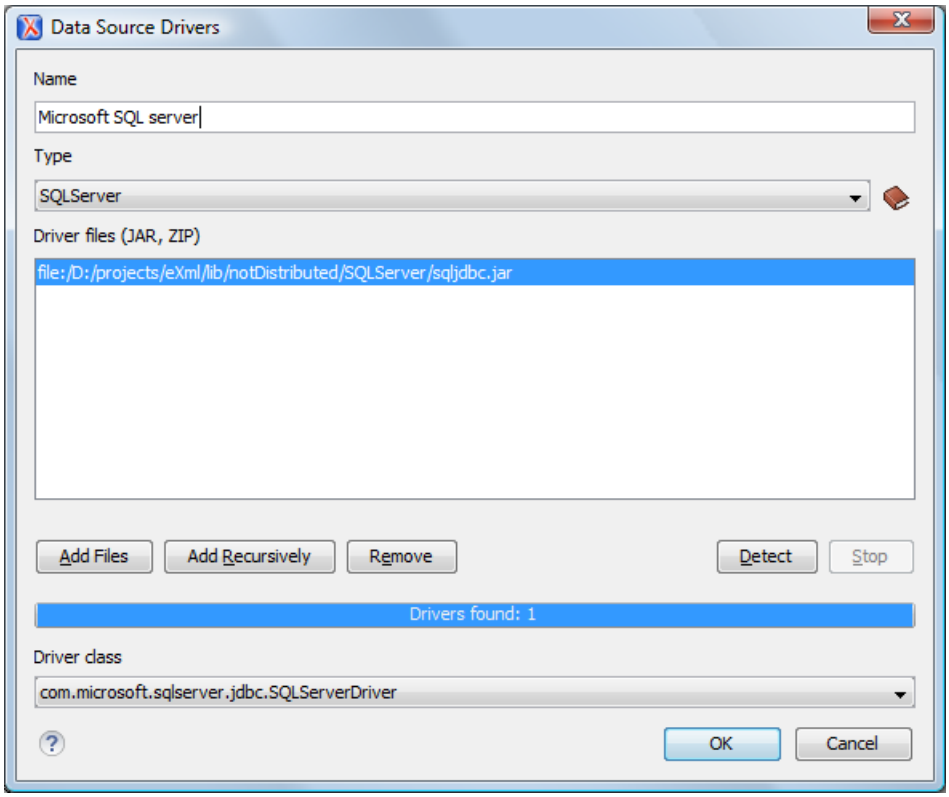


Figure 507: Data Source Drivers Configuration Dialog Box

4. Enter a unique name for the data source.
5. Select *SQLServer* in the driver **Type** drop-down menu.
6. Click the **Add Files** button and select the Microsoft SQL Server driver file that you downloaded.
   The SQL Server driver file is called `sqljdbc.jar`.
7. Select the most appropriate **Driver class**.
8. Click the **OK** button to finish the data source configuration.
9. Continue on to *configure your Microsoft SQL Server connection*.

How to Configure a Microsoft SQL Server Connection

The support to configure a Microsoft SQL Server connection is available in the Enterprise edition only.

To configure a connection to a Microsoft SQL Server, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Data Sources**.
2. Click the + **New** button in the **Connections** panel.
   The dialog box for configuring a database connection is displayed.

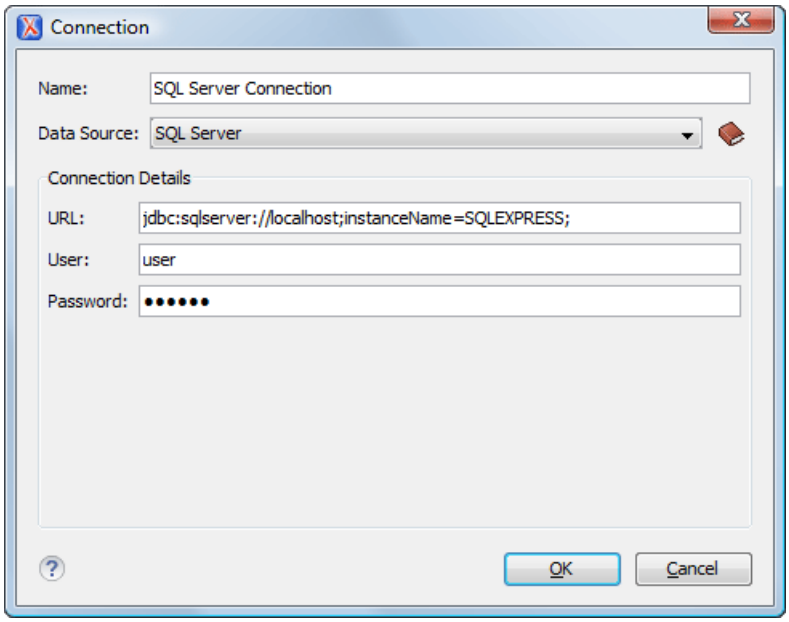


Figure 508: Connection Configuration Dialog Box

3. Enter a unique name for the connection.
4. Select the *SQL Server* data source in the **Data Source** drop-down menu.
5. Enter the connection details.
   a) Enter the URL of the SQL Server server.

      If you want to connect to the server using Windows integrated authentication, you must add `;integratedSecurity=true` to the end of the URL. The URL will look like this:

      ```
      jdbc:sqlserver://localhost;instanceName=SQLEXPRESS;integratedSecurity=true;
      ```

      Note: For integrated authentication, leave the **User** and **Password** fields empty.
   b) Enter the user name for the connection to the SQL Server.
   c) Enter the password for the connection to the SQL Server.
6. Click the **OK** button to finish the connection configuration.
7. To view your connection, go to the ***Data Source Explorer*** *view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

Microsoft SQL Server Contextual Menu Actions

General Contextual Menu Actions

For relational databases, the following general actions are available in the contextual menu of the ***Data Source Explorer*** *view*, depending on the node in which it is invoked:

Refresh

Performs a refresh on the selected node.

Disconnect (available on Connection nodes)

Closes the current database connection. If a table is already open, you are warned to close it before proceeding.

Configure Database Sources (available on Connection nodes)

Opens the ***Data Sources*** *preferences page* where you can configure both data sources and connections.

Edit (available on Table nodes)

Opens the selected table in the ***Table Explorer*** *view*.

Export to XML (available on Table nodes)

Opens the **Export Criteria** dialog box (a thorough description of this dialog box can be found in the *Import from Database* chapter).

Database-Specific Contextual Menu Actions

In addition to the general contextual menu actions in the ***Data Source Explorer** view*, the resource level nodes in Microsoft SQL Server connections include the following additional contextual menu action:

XML Schema Repository Level Nodes

Register

Opens a dialog box for adding a new schema file in the DB XML repository. In this dialog box, you enter a collection name and the necessary schema files. Schema dependencies management can be done by using the **Add** and **Remove** buttons.

Schema Level Nodes

Add

Adds a new schema to the XML Schema files.

Unregister

Removes the selected schema from the XML Schema Repository.

View

Opens the selected schema in Oxygen XML Editor.

Oracle Database Connections

The Oracle database is a common relational type of database system. Oxygen XML Editor comes with built-in support for the 11g version of the database system. The Oracle database also includes a Oracle XML DB component that adds native XML support. Oxygen XML Editor allows you to browse Oracle repositories in the ***Data Source Explorer** view*, open tables in the ***Table Explorer** view*, and perform various operations on the resources in the repository.

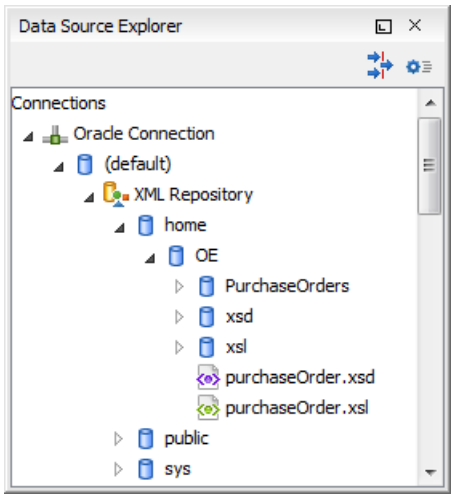


Figure 509: Oracle Database Connection

Related Information:
Using XQuery with Oracle XML DB

Configuring an Oracle 11g Database Connection

To configure the support for a Oracle 11g database, follow this procedure:

1. Go to *http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html* and download the Oracle 11g JDBC driver called `ojdbc6.jar`.

2. *Configure Oracle 11g Data Source drivers*.
3. *Configure an Oracle 11g Connection*.
4. To view your connection, go to the ***Data Source Explorer** view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

How to Configure Oracle 11g Data Source Drivers

Available in the Enterprise edition only.

To configure a data source for connecting to an Oracle 11g server, follow these steps:

1. Go to *http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html* and download the Oracle 11g JDBC driver called `ojdbc6.jar`.
2. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Data Sources**.
3. Click the **New** button in the **Data Sources** panel.

   The dialog box for configuring a data source is opened.

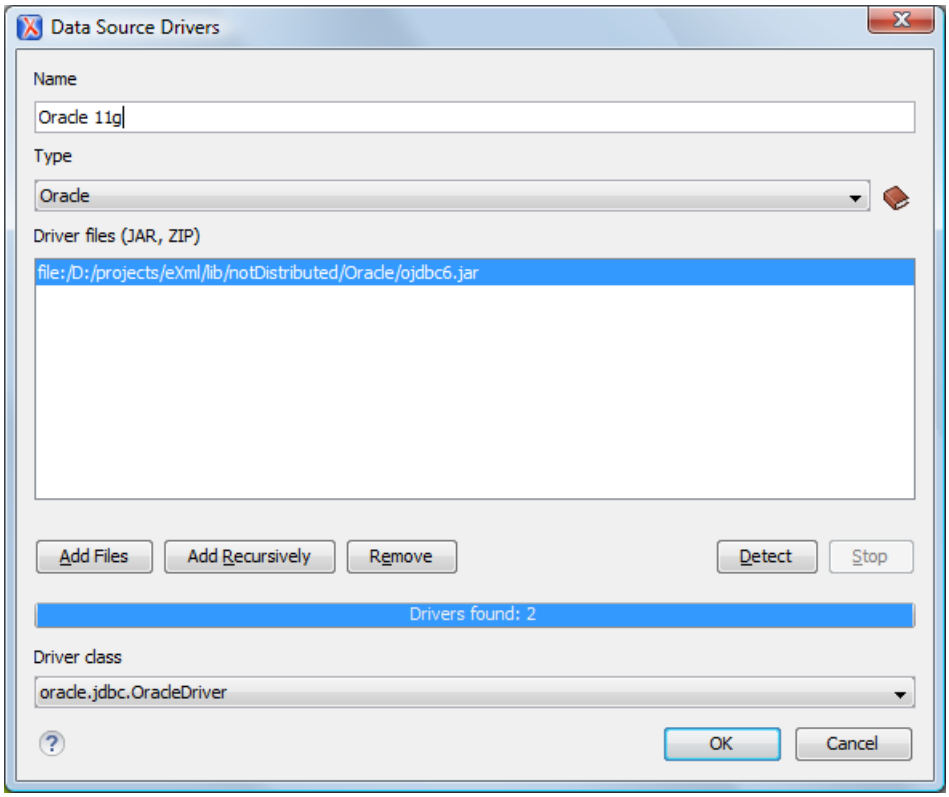


Figure 510: Data Source Drivers Configuration Dialog Box

4. Enter a unique name for the data source.
5. Select *Oracle* in the driver **Type** drop-down menu.
6. Click the **Add Files** button and select the Oracle driver file that you downloaded.

   The Oracle driver file is called `ojdbc5.jar`.
7. Select the most appropriate **Driver class**.
8. Click the **OK** button to finish the data source configuration.
9. Continue on to *configure your Oracle connection*.

How to Configure an Oracle 11g Connection

Available in the Enterprise edition only.

To configure a connection to an Oracle 11g server, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Data Sources**.

2. Click the **New** button in the **Connections** panel.

   The dialog box for configuring a database connection is displayed.

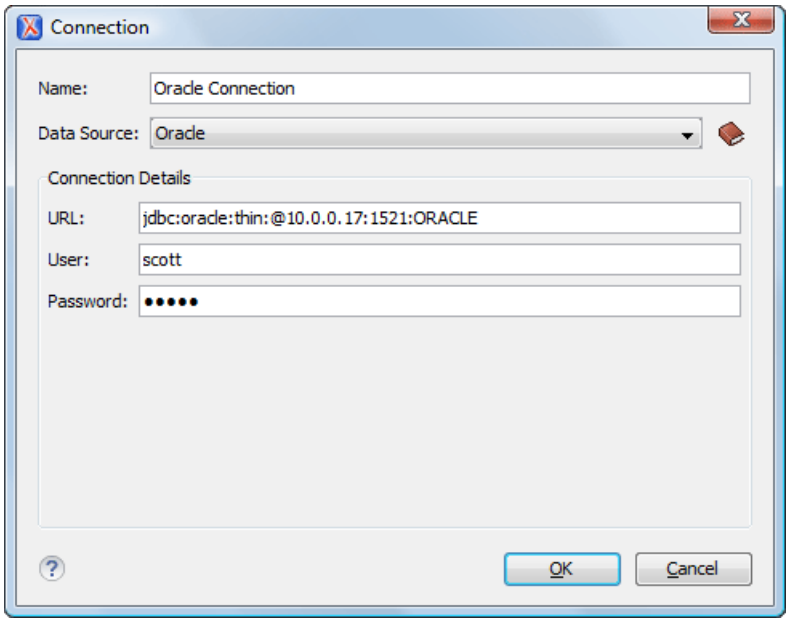


Figure 511: Connection Configuration Dialog Box

3. Enter a unique name for the connection.
4. Select the *Oracle 11g* data source in the **Data Source** drop-down menu.
5. Enter the connection details.
   a) Enter the URL of the Oracle server.
   b) Enter the user name for the connection to the Oracle server.
   c) Enter the password for the connection to the Oracle server.
6. Click the **OK** button to finish the connection configuration.
7. To view your connection, go to the ***Data Source Explorer*** *view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

Oracle Database Contextual Menu Actions

General Contextual Menu Actions

For relational databases, the following general actions are available in the contextual menu of the ***Data Source Explorer*** *view*, depending on the node in which it is invoked:

Refresh

Performs a refresh on the selected node.

Disconnect (available on Connection nodes)

Closes the current database connection. If a table is already open, you are warned to close it before proceeding.

Configure Database Sources (available on Connection nodes)

Opens the ***Data Sources*** *preferences page* where you can configure both data sources and connections.

Edit (available on Table nodes)

Opens the selected table in the ***Table Explorer*** *view*.

Export to XML (available on Table nodes)

Opens the **Export Criteria** dialog box (a thorough description of this dialog box can be found in the *Import from Database* chapter).

Database-Specific Contextual Menu Actions

In addition to the general contextual menu actions in the ***Data Source Explorer** view*, the various nodes in Oracle database connections include the following additional contextual menu actions:

XML Schema Repository Level Nodes

Register

Opens a dialog box for adding a new schema file in the XML repository. To add an XML Schema, enter the schema URI and location on your file system. *Local* scope means that the schema is visible only to the user who registers it. *Global* scope means that the schema is public.

Note: Registering a schema may involve dropping/creating types. Hence you need type-related privileges such as DROP TYPE, CREATE TYPE, and ALTER TYPE. You need privileges to delete and register the XML schemas involved in the registering process. You need all privileges on XMLType tables that conform to the registered schemas. For XMLType columns, the ALTER TABLE privilege is needed on corresponding tables. If there are schema-based XMLType tables or columns in other database schemas, you need privileges such as the following:

- `CREATE ANY TABLE`
- `CREATE ANY INDEX`
- `SELECT ANY TABLE`
- `UPDATE ANY TABLE`
- `INSERT ANY TABLE`
- `DELETE ANY TABLE`
- `DROP ANY TABLE`
- `ALTER ANY TABLE`
- `DROP ANY INDEX`

To avoid having to grant all these privileges to the schema owner, Oracle recommends that the registration be performed by a DBA if there are XML schema-based XMLType table or columns in other user database schemas.

XML Repository Level Nodes

Add container

Adds a new child container to the current one.

Add resource

Adds a new resource to the folder.

Container Level Nodes

Add container

Adds a new child container to the current one.

Add resource

Adds a new resource to the folder.

Delete

Deletes the current container.

Properties

Shows various properties of the current container.

Resource Level Nodes

Open

Opens the selected resource in the editor.

Open in System Application

When you use this action, Oxygen XML Editor downloads the selected resource to a local temporary folder and opens the selected resource in the system application that is currently set as the default application

associated with that type of resource. You can then edit the resource, save it, and when you switch the focus back to the **Data Source Explorer** view, Oxygen XML Editor will detect that there was a change and will ask if you want to upload the edited resource to the server.

Rename

Renames the current resource

Move

Moves the current resource to a new container (also available through drag and drop).

Delete

Deletes the current container.

Copy location

Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

Properties

Shows various properties of the current container.

Compare

Compares two selected resources using the ***Compare Files*** *tool*.

PostgreSQL Database Connections

Oxygen XML Editor includes support for PostgreSQL database connections. Oxygen XML Editor allows you to browse the structure of a PostgreSQL database in the ***Data Source Explorer*** *view*, open tables in the ***Table Explorer*** *view*, and perform various operations on the resources in the repository.

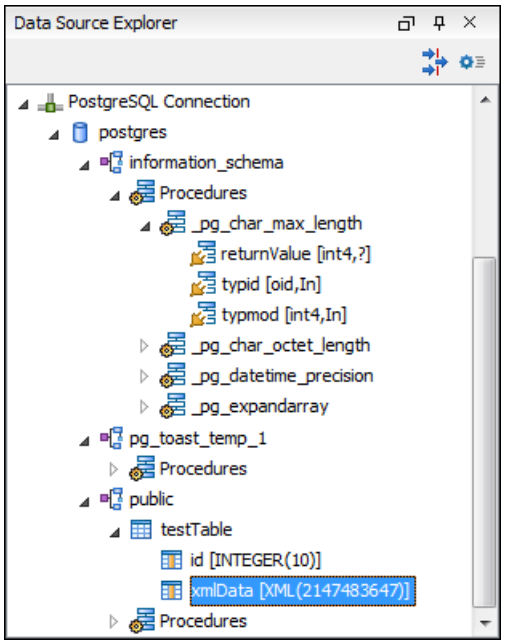


Figure 512: PostgreSQL Database Connection

Configuring a PostgreSQL Database Connection

To configure the support for a PostgreSQL database, follow this procedure:

1. Go to *http://jdbc.postgresql.org/download.html* and download the PostgreSQL 8.3 JDBC3 driver.
2. *Configure PostgreSQL Data Source drivers*.
3. *Configure a PostgreSQL Connection*.
4. To view your connection, go to the ***Data Source Explorer*** *view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

How to Configure PostgreSQL 8.3 Data Source Drivers

To configure a data source for connecting to a PostgreSQL server, follow these steps:

1. Go to *http://jdbc.postgresql.org/download.html* and download the PostgreSQL 8.3 JDBC3 driver.
2. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Data Sources**.
3. Click the **New** button in the **Data Sources** panel.

   The dialog box for configuring a data source is opened.

4. Enter a unique name for the data source.
5. Select *PostgreSQL* in the driver **Type** drop-down list.
6. Click the **Add Files** button and select the PostgreSQL driver file that you downloaded.
   The PostgreSQL driver file is called `postgresql-8.3-603.jdbc3.jar`.
7. Select the most appropriate **Driver class**.
8. Click the **OK** button to finish the data source configuration.
9. Continue on to *configure your PostgreSQL connection*.

How to Configure a PostgreSQL 8.3 Connection

To configure a connection to a PostgreSQL 8.3 server, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Data Sources**.
2. Click the **New** button in the **Connections** panel.

   The dialog box for configuring a database connection is displayed.

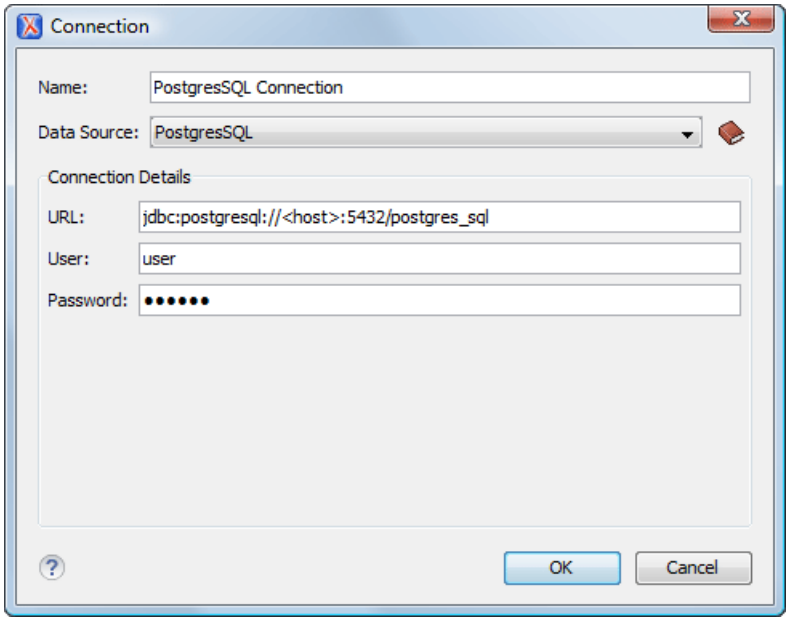


Figure 513: Connection Configuration Dialog Box

3. Enter a unique name for the connection.
4. Select the *PostgreSQL 8.3* data source in the **Data Source** drop-down menu.
5. Enter the connection details.
   a) Enter the URL of the PostgreSQL 8.3 server.
   b) Enter the user name for the connection to the PostgreSQL 8.3 server.
   c) Enter the password for the connection to the PostgreSQL 8.3 server.
6. Click the **OK** button to finish the connection configuration.
7. To view your connection, go to the ***Data Source Explorer** view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

PostgreSQL Contextual Menu Actions

General Contextual Menu Actions

For relational databases, the following general actions are available in the contextual menu of the ***Data Source Explorer*** *view*, depending on the node in which it is invoked:

Refresh

Performs a refresh on the selected node.

Disconnect (available on Connection nodes)

Closes the current database connection. If a table is already open, you are warned to close it before proceeding.

Configure Database Sources (available on Connection nodes)

Opens the ***Data Sources*** *preferences page* where you can configure both data sources and connections.

Edit (available on Table nodes)

Opens the selected table in the ***Table Explorer*** *view*.

Export to XML (available on Table nodes)

Opens the **Export Criteria** dialog box (a thorough description of this dialog box can be found in the *Import from Database* chapter).

Database-Specific Contextual Menu Actions

In addition to the general contextual menu actions in the ***Data Source Explorer*** *view*, the resource level nodes in PostgreSQL connections include the following additional contextual menu action:

Resource Level Nodes

Compare

Compares two selected resources using the ***Compare Files*** *tool*.

Berkeley DB XML Database Connections

Oxygen XML Editor includes support for Berkeley DB XML database connections. Oxygen XML Editor allows you to browse the structure of a Berkeley DB XML database in the ***Data Source Explorer*** *view* and perform various operations on the resources in the repository.

Oracle Berkeley DB XML is an open source, embeddable XML database with XQuery-based access to documents stored in containers and indexed based on their content. It is built on top of the Oracle Berkeley DB and inherits its features and attributes, along with native XML support. A detailed description can be found at: *http://www.oracle.com/us/products/database/berkeley-db/xml/overview/index.html*.

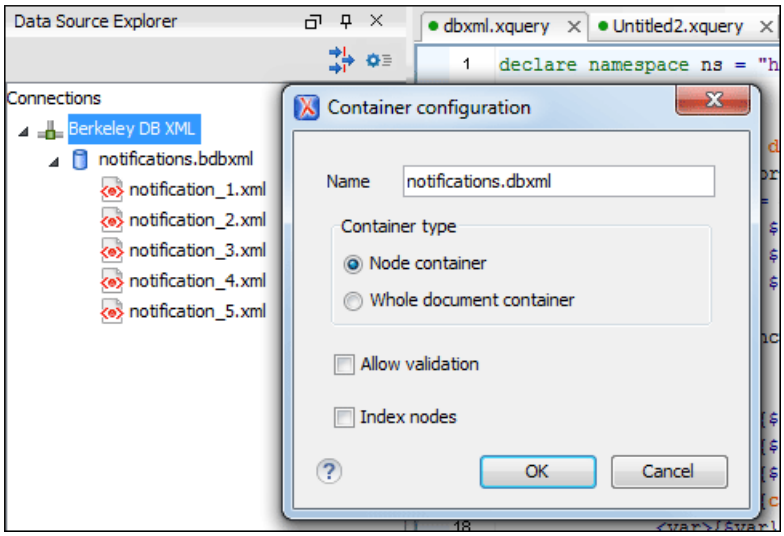


Figure 514: Berkeley DB XML Connection

Configuring a Berkeley DB XML Database Connection

Follow this procedure to configure the support for a Berkeley DB XML database:

1. *Configure Berkeley DB XML Data Source drivers*.
2. *Configure a Berkeley DB XML Connection*.
3. To view your connection, go to the ***Data Source Explorer** view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

How to Configure Berkeley DB XML Data Source Drivers

For this procedure, you need to already have a Berkeley DB XML database installed on your system.

Oxygen XML Editor supports Berkeley DB XML versions 2.3.10, 2.4.13, 2.4.16 & 2.5.16. To configure a data source for a Berkeley DB XML database, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Data Sources**.
2. Click the ✚ **New** button in the **Data Sources** panel.
3. Enter a unique name for the data source.
4. Select *Berkeley DBXML* from the driver **Type** drop-down menu.
5. Click the **Add Files** button to add the Berkeley DB driver files.

   The driver files for the Berkeley DB database (and their locations) are as follows:

   - `db.jar` (check for it in `[DBXML_DIR]/lib` or `[DBXML_DIR]/jar`)
   - `dbxml.jar` (check for it in `[DBXML_DIR]/lib` or `[DBXML_DIR]/jar`)

   Where `[DBXML_DIR]` is the Berkeley DB XML database root directory. For example, in Windows it is: `C:\Program Files\Oracle\Berkeley DB XML <version>`.
6. Click the **OK** button to finish the data source configuration.
7. Continue on to *configure your Berkeley DB XML connection*.

How to Configure a Berkeley DB XML Connection

Oxygen XML Editor supports Berkeley DB XML versions 2.3.10, 2.4.13, 2.4.16 & 2.5.16. To configure a connection to a Berkeley DB XML database, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Data Sources**.
2. Click the ✚ **New** button in the **Connections** panel.
3. Enter a unique name for the connection.
4. Select a previously configured Berkeley data source from the **Data Source** drop-down menu.
5. Enter the connection details.

a) Set the path to the Berkeley DB XML `database` directory in the **Environment home directory field**. Use a directory with write access. DO NOT use the installation directory where Berkeley DB XML is installed if you do not have write access to that directory.
b) Select the **Verbosity** level: *DEBUG*, *INFO*, *WARNING*, or *ERROR*.
c) Optionally, you can select the **Join existing environment** checkbox.

   If selected, an attempt is made to join an existing environment in the specified home directory and all the original environment settings are preserved. If that fails, try reconfiguring the connection with this option unchecked.

6. Click the **OK** button to finish the connection configuration.
7. To view your connection, go to the ***Data Source Explorer*** *view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

Berkeley DB XML Contextual Menu Actions

While browsing Berkeley DB XML connections in the ***Data Source Explorer*** *view*, the various nodes include the following contextual menu actions:

Connection Level Nodes

Configure Database Sources

Opens the ***Data Sources*** *preferences page* where you can configure both data sources and connections.

Disconnect

Stops the connection.

New Collection

Opens a **Container configuration** dialog box that allows you to adds a new container in the repository.

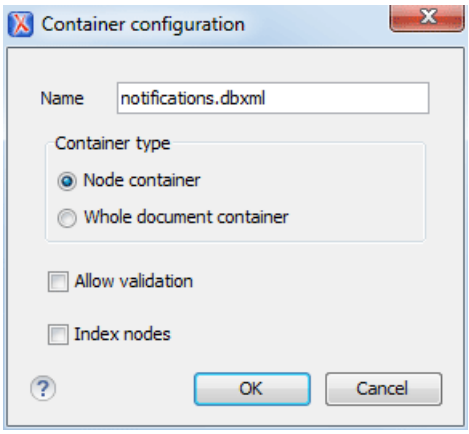


Figure 515: Container Configuration Dialog Box

This dialog box allows you to configure the following:

- **Name** - The name of the new container.
- **Container type** - At creation time, every container must have a type defined for it. This container type identifies how XML documents are stored in the container. As such, the container type can only be determined at container creation time. You cannot change it when subsequent container opens. You can select one of the following types:
  - **Node container** - XML documents are stored as individual nodes in the container. Each record in the underlying database contains a single leaf node, its attributes and attribute values (if any), and its text nodes (if any). Berkeley DB XML also keeps the information it requires to reassemble the document from the individual nodes stored in the underlying databases. This is the default selection and is the preferred container type.
  - **Whole document container** - The container contains entire documents. The documents are stored without any manipulation of line breaks or whitespace.
- **Allow validation** - If selected, documents will be validated when they are loaded into the container. The default behavior is to not validate documents.

- **Index nodes** - If selected, indices for the container will return nodes rather than documents. The default is to index at the document level. This property has no meaning if the container type is **Whole document container**.

Refresh

Performs a refresh on the selected node.

Properties

Shows various properties of the current container.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Container Level Nodes

Import Files

Allows you to add a new file on the connection, in the current folder.

Export

Allows you to export the folder on the remote connection to a local folder.

Cut

Removes the current selection and places it in the clipboard.

Paste

Pastes the copied selection.

Rename

Renames the current resource

Delete

Deletes the current container.

Edit indices

Opens a **Container Indices** dialog box that allows you to configure indices properties for the selected Berkeley container.

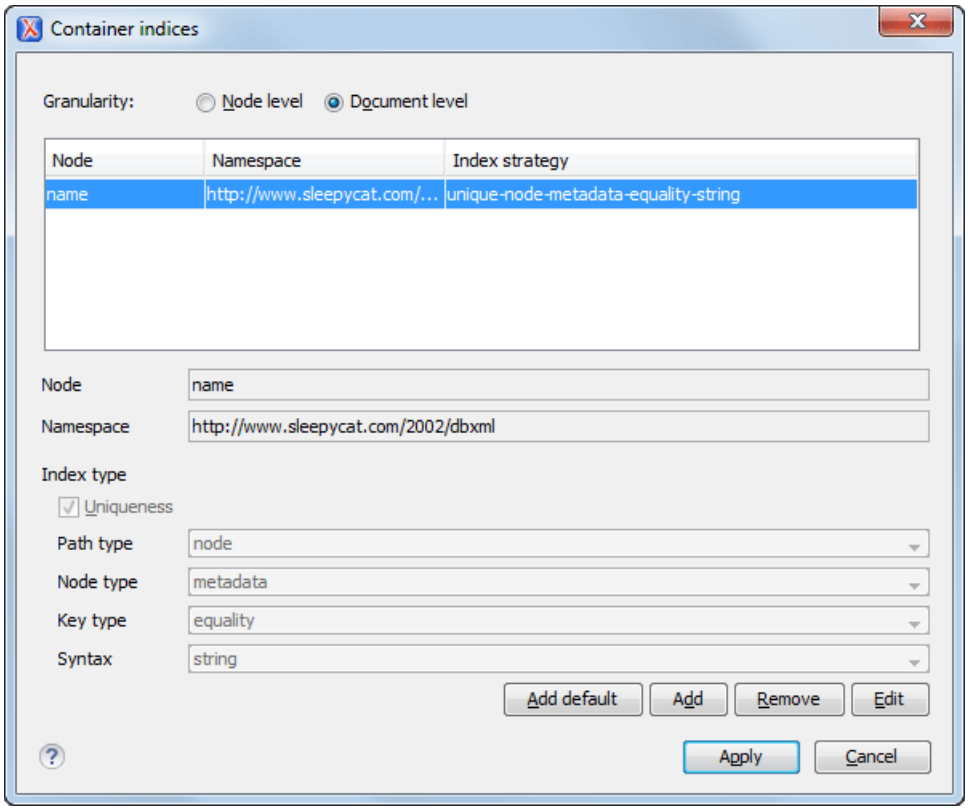


Figure 516: Container Indices Dialog Box

This dialog box allows you to configure the following properties:

- **Granularity** - A measure of the level of details of your data in the database. You can select one of the following:
  - **Document level** - Good option for retrieving large documents.
  - **Node level** - Good option for retrieving nodes from within documents.
- **Node** - The name of the node.
- **Namespace** - The index namespace.
- **Index type**:
  - **Uniqueness** - Indicates whether or not the indexed value must be unique within the container.
  - **Path type** - Drop-down menu that allows you to select from the following:
    - **node** - Indicates that you want to index a single node in the path.
    - **edge** - Indicates that you want to index the portion of the path where two nodes meet.
  - **Node type** - Drop-down menu that allows you to select from the following:
    - **element** - An element node in the document content.
    - **attribute** - An attribute node in the document content.
    - **metadata** - A node found only in the metadata content of a document.
  - **Key type** - Drop-down menu that allows you to select from the following:
    - **equality** - Improves the performances of tests that look for nodes with a specific value.
    - **presence** - Improves the performances of tests that look for the existence of a node regardless of its value.
    - **substring** - Improves the performance of tests that look for a node whose value contains a given sub-string.
  - **Syntax** - The syntax describes the type of data the index contains and is mostly used to determine how indexed values are compared. The default value is `string`.

Refresh

Performs a refresh on the selected node.

Properties

Shows various properties of the current container.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Resource Level Nodes

Open

Opens the selected resource in the editor.

Open in System Application

When you use this action, Oxygen XML Editor downloads the selected resource to a local temporary folder and opens the selected resource in the system application that is currently set as the default application associated with that type of resource. You can then edit the resource, save it, and when you switch the focus back to the **Data Source Explorer** view, Oxygen XML Editor will detect that there was a change and will ask if you want to upload the edited resource to the server.

Cut

Removes the current selection and places it in the clipboard.

Copy location

Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

Rename

Renames the current resource

Delete

Deletes the current container.

Refresh

Performs a refresh on the selected node.

Properties

Shows various properties of the current container.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Compare

Compares two selected resources using the ***Compare Files*** *tool*.

Debugging with Berkeley DB XML

The Berkeley DB XML database added a debugging interface starting with version 2.5. The current version is supported in the Oxygen XML Editor XQuery Debugger. *The same restrictions and peculiarities* apply for the Berkeley debugger as for the MarkLogic debugger.

eXist Database Connections

Oxygen XML Editor includes support for eXist database connections. Oxygen XML Editor allows you to browse the structure of a eXist database in the ***Data Source Explorer*** *view* and perform various operations on the resources in the repository.

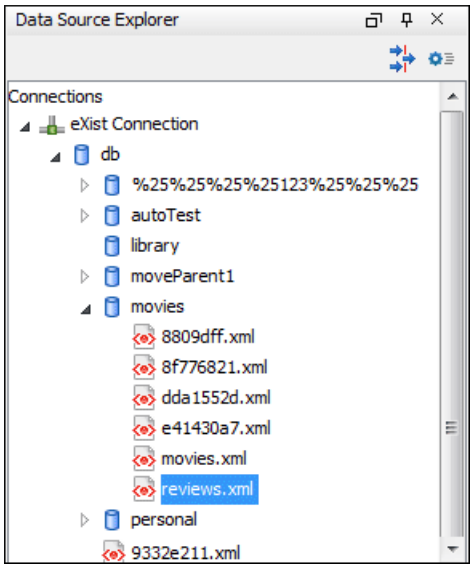


Figure 517: eXist Database Connection

Configuring an eXist Database Connection

There are two ways to configure the support for an eXist database:

1. Use the dedicated **Create eXist-db XML connection** connection wizard.
   a. *Open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Data Sources** and click the **Create eXist-db XML connection** link.
   b. Enter your connection details in the connection wizard and click **OK**.

      Important: To create an eXist connection using this wizard, Oxygen XML Editor expects the `exist/webstart/exist.jnlp` path to be accessible at the provided **Host** and **Port**.
   c. To view your connection, go to the ***Data Source Explorer** view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.
2. Use the **Data Sources** preferences page to configure your connection.
   a. *Configure eXist Data Source drivers*.
   b. *Configure an eXist Connection*.
   c. To view your connection, go to the ***Data Source Explorer** view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

How to Configure eXist Data Source Drivers

For this procedure, you need to already have an eXist database server installed.

Oxygen XML Editor supports eXist database server versions up to and including version 3.0. To configure a data source for an eXist database, follow these steps:

1. *Open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Data Sources**.
2. Click the **New** button in the **Data Sources** panel.
3. Enter a unique name for the data source.
4. Select *eXist* from the driver **Type** drop-down menu.
5. Click the **Add Files** button to add the eXist driver files.

   The following driver files should be added and they are found in the installation directory of the eXist database server. Make sure you copy the files from the installation of the eXist server where you want to connect from Oxygen XML Editor.

   - `exist.jar`
   - `lib/core/xmldb.jar`
   - `lib/core/xmlrpc-client-3.1.x.jar`
   - `lib/core/xmlrpc-common-3.1.x.jar`

- `lib/core/ws-commons-util-1.0.x.jar`
- `lib/core/slf4j-api-1.x.x.jar` (if available)
- `lib/core/slf4j-log4j12-1.x.x.jar` (if available)

The version number from the driver file names may be different for your eXist server installation.

6. Click the **OK** button to finish the data source configuration.
7. Continue on to *configure your eXist connection*.

   To watch our video demonstration about running XQuery against an eXist XML database, go to *https://www.oxygenxml.com/demo/eXist_Database.html*.

How to Configure an eXist Connection

To configure a connection to an eXist database, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Data Sources**.
2. Click the **New** button in the **Connections** panel.
3. Enter a unique name for the connection.
4. Select a previously configured eXist data source from the **Data Source** drop-down menu.
5. Enter the connection details.
   a) Set the URI to the installed eXist engine in the **XML DB URI** field.
   b) Set the user name in the **User** field.
   c) Set the password in the **Password** field.
   d) Enter the start collection in the **Collection** field.

      eXist organizes all documents in hierarchical collections. Collections are like directories. They are used to group related documents together. This text field allows the user to set the default collection name.
6. Click the **OK** button to finish the connection configuration.
7. To view your connection, go to the ***Data Source Explorer** view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

   To watch our video demonstration about running XQuery against an eXist XML database, go to *https://www.oxygenxml.com/demo/eXist_Database.html*.

eXist Contextual Menu Actions

While browsing eXist database connections in the ***Data Source Explorer** view*, the various nodes include the following contextual menu actions:

Connection Level Nodes

Configure Database Sources

Opens the ***Data Sources** preferences page* where you can configure both data sources and connections.

Disconnect

Stops the connection.

Refresh

Performs a refresh on the selected node.

Find/Replace in Files

Opens the ***Find/Replace in Files** dialog box* that allows you to find and replace text in multiple files from the connection.

Container Level Nodes

New File

Creates a new file on the connection, in the current folder.

New Collection

Creates a new collection on the connection.

Import Folders

Imports folders on the server.

Import Files

Allows you to add a new file on the connection, in the current folder.

Export

Allows you to export the folder on the remote connection to a local folder.

Cut

Removes the current selection and places it in the clipboard.

Paste

Pastes the copied selection.

Refresh

Performs a refresh on the selected node.

Properties

Shows various properties of the current container.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Resource Level Nodes

Open

Opens the selected resource in the editor.

Open in System Application

When you use this action, Oxygen XML Editor downloads the selected resource to a local temporary folder and opens the selected resource in the system application that is currently set as the default application associated with that type of resource. You can then edit the resource, save it, and when you switch the focus back to the **Data Source Explorer** view, Oxygen XML Editor will detect that there was a change and will ask if you want to upload the edited resource to the server.

Save As

Allows you to save the selected resource as a file on disk.

Cut

Removes the current selection and places it in the clipboard.

Copy

Copies the current selection into the clipboard.

Copy location

Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

Rename

Renames the current resource

Delete

Deletes the current container.

Refresh

Performs a refresh on the selected node.

Properties

Shows various properties of the current container.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Compare

Compares two selected resources using the ***Compare Files*** *tool*.

MarkLogic Database Connections

Oxygen XML Editor Enterprise edition includes support for MarkLogic database connections. Once you *configure a MarkLogic connection*, you can use the ***Data Source Explorer*** *view* to display all the application servers that are configured on the MarkLogic server. You can expand each application server and view all of its configured modules, and the ***Data Source Explorer*** *view* allows you to open and edit these modules.

Note: To browse modules located in a database, directory properties must be associated with them. These directory properties are generated automatically if the *directory creation* property of the database is set to automatic. If this property is set to **manual** or **manual-enforced**, add the directory properties of the modules manually, using the XQuery function `xdmp:directory-create()`. For example, for two documents with the `/code/modules/main.xqy` and `/code/modules/imports/import.xqy` IDs, run the following query:

```
(xdmp:directory-create('/code/modules/'), xdmp:directory-create('/code/modules/
imports/'))
```

For more information about directory properties, go to: *http://blakeley.com/blogofile/2012/03/19/directory-assistance/*.

MarkLogic and XQuery

MarkLogic connections can be used in conjunction with XQuery scripts to debug and solve problems with XQuery transformations. XQuery modules can also be validated using a MarkLogic server to allow to you to spot possible issues without the need of actually executing the XQuery script.

When *debugging XQuery files with MarkLogic*, you can use the ***Data Source Explorer*** *view* to open the files from the application server that is involved in the debugging process. By using the ***Data Source Explorer*** *view*, any imported modules are better identified by the MarkLogic server. You can also *use step actions and breakpoints* in the modules to help identify problems.

Modules Container

For each Application server (for example: *Bill (HTTP port:8060)*), you have access to the XQuery modules that are visible to that server. When editing, executing, or debugging XQuery it is recommended to open the XQuery files from this **Modules** container.

Note: You can also manage resources for a MarkLogic database through a WebDAV connection, although it is not recommended if you work with XQuery files since imported modules may not be resolved correctly.

Requests Container

Each MarkLogic application server includes a **Requests** container. In this container, Oxygen XML Editor displays both queries that are stopped for debugging purposes and queries that are still running. To clean up the entire **Requests** container at the end of your session, right-click it and use the ***Cancel all requests*** *action*.

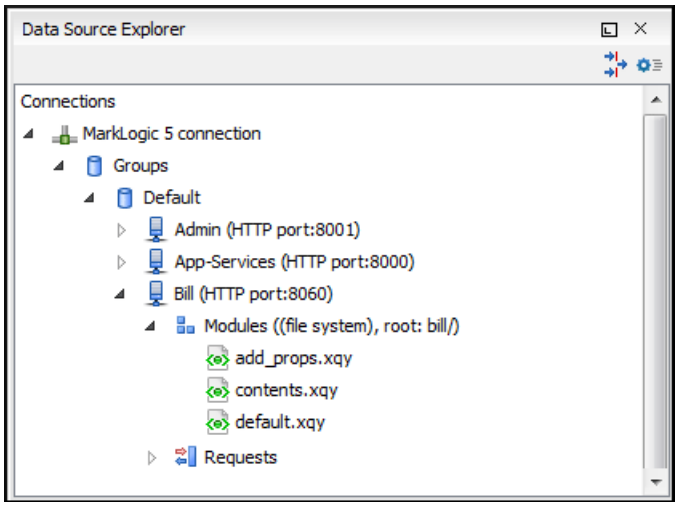


Figure 518: MarkLogic Connection in Data Source Explorer

Configuring a MarkLogic Database Connection

Note that this feature is available in Oxygen XML Editor Enterprise edition only.

Follow this procedure to configure the support for a MarkLogic database connection:

1. Download the MarkLogic driver from *MarkLogic Community site*.
2. *Configure MarkLogic Data Source drivers*.
3. *Configure a MarkLogic Connection*.
4. To view your connection, go to the ***Data Source Explorer** view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

Related Information:

MarkLogic Development in Oxygen XML Editor on page 1096

How to Configure MarkLogic Data Source Drivers

Available in the Enterprise edition only.

Note: Oxygen XML Editor supports MarkLogic version 4.0 or later.

To configure a data source for MarkLogic, follow this procedure:

1. Download the **XCC Java distribution** zip file from: *http://developer.marklogic.com/products/xcc*.
2. Unzip the downloaded archive.
3. *Open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Data Sources**.
4. Click the **New** button in the **Data Sources** panel.
5. Enter a unique name for the data source.
6. Select *MarkLogic* from the driver **Type** drop-down list.
7. Click the **Add Files** button and select the MarkLogic driver file from the `lib` folder of the archive that you downloaded and unzipped. The driver file name is `marklogic-xcc-{server_version}.jar`, where *{server_version}* is the MarkLogic server version.
8. Click the **OK** button to finish the data source configuration.
9. Continue on to *configure your MarkLogic Connection*.

How to Configure a MarkLogic Connection

Available in the Enterprise edition only.

Note: Oxygen XML Editor supports MarkLogic version 4.0 or later.

To configure a connection to a MarkLogic database, follow these steps:

1. *Open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Data Sources**.

2. Click the **New** button in the **Connections** panel.
3. Enter a unique name for the connection.
4. Select a previously configured MarkLogic data source from the **Data Source** drop-down menu.
5. Enter the connection details.
   a) The host name or IP address of the installed MarkLogic engine in the **XDBC Host** field.

      Oxygen XML Editor uses XCC connector to interact with MarkLogic XDBC server and requires the basic authentication schema to be set. Starting with version MarkLogic 4.0 the default authentication method when you create an HTTP or WebDAV Server is *digest*, so make sure to change it to *basic*.
   b) Set the port number of the MarkLogic engine in the **Port** field. A MarkLogic XDBC application server must be configured on the server on this port. This XDBC server will be used to process XQuery expressions against the server. Later, if you want to change the XDBC server, instead of editing the configuration just use the ***Use it to execute queries*** *action* from Data Source Explorer.
   c) Set the user name to access the MarkLogic engine in the **User** field.
   d) Set the password to access the MarkLogic engine in the **Password** field.
   e) Optionally, in the **WebDAV URL** field, set the URL used for browsing the MarkLogic database in the ***Data Source Explorer*** *view*.

      The **Database** field specifies the database for which the XQuery expressions are executed. If you set this option to default, the database associated to the application server of the configured port is used.
6. Click the **OK** button to finish the connection configuration.
7. To view your connection, go to the ***Data Source Explorer*** *view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

MarkLogic Development in Oxygen XML Editor

The Oxygen XML Editor support for MarkLogic includes features designed for developers, such as debugging XQuery transformations, remote and collaborative debugging, XQuery editing and validation, and an *XQuery builder* that helps to improve productivity.

Working with XQuery Files

MarkLogic supports working with XQuery files to create queries over stored XML content. You can open an XQuery file, configure a transformation scenario to match your MarkLogic connection, write the XQuery, and then execute it.

When editing XQuery modules stored on the MarkLogic server, the ***Outline*** *view* collects and displays all the functions from all imported modules. The *Content Completion Assistant* also presents all of these functions along with the latest built-in XQuery functions in accordance with the server version.

When developing queries for MarkLogic, it is best to open the resources from the ***Data Source Explorer*** *view*. When you execute or debug XQuery files opened from this view, imported modules can be resolved better by the MarkLogic server. Another advantage is that validation is automatically performed on the MarkLogic server, including any imported modules.

XQuery Debugging

Oxygen XML Editor allows you to use MarkLogic connections to debug real applications that use XQuery (for example, web applications that trigger XQuery executions). By setting the server in debug mode, you can intercept all the XQuery scripts that run on that server. Oxygen XML Editor connects to the MarkLogic server, shows you the running XQuery scripts, and allows you to debug them. The remote debugging support also allows you to debug collaboratively. Multiple users can participate in the same debugging session. You can start a debugging session and another user can continue it, and vice versa.

Working with Modules

MarkLogic has a concept of two types of XQuery modules, *library* and *main* modules. A *library* module is used to define functions. Library modules cannot be evaluated directly. They are imported, either from other library modules or from main modules. A *main* module is used as an entry point that can be executed as an XQuery program. For more information on these types of modules, see *XQuery Library Modules and Main Modules*.

When working with *library* modules, you need to create a validation scenario and associate it with the module. In the validation scenario you need to specify a main module as the entry point for validation. The modules need to be deployed on a MarkLogic server because Oxygen XML Editor will request the server to validate the modules.

To validate *library* modules stored on a MarkLogic server, follow these steps:

1. *Configure a MarkLogic database connection*.
2. Expand the MarkLogic connection in the ***Data Source Explorer*** *view* and open the *library* modules. The *main* module must also be opened from the ***Data Source Explorer*** *view*.
3. *Configure a validation scenario* for each *library* module. Specify the *main* module in the **URL of the file to validate** field.

   Result: Validation is done on the server that contains the *main* module. The *main* module and all other *library* modules involved in the validation must be saved. Otherwise, the server will validate what was saved on the server, without the uncommitted changes. Also, the *Content Completion Assistant* and the ***Outline*** *view* should now present the functions from all the modules.

Related Information:

Debugging with MarkLogic on page 1097

Configuring a MarkLogic Database Connection on page 1095

Debugging with MarkLogic

Oxygen XML Editor includes support for debugging XQuery transformations that are executed against a MarkLogic database.

To use a debugging session against the MarkLogic engine, follow these steps:

1. Configure a *MarkLogic data source* and a *MarkLogic connection*.
2. Make sure that the debugging support is enabled in the MarkLogic server that Oxygen XML Editor accesses. On the server side, debugging must be activated in the XDBC server and in the *Task Server* section of the server control console (the switch *debug allow*). If the debugging is not activated, the MarkLogic server reports a `DBG-TASKDEBUGALLOW` error.

   Note: An XDBC application server must be running to connect to the MarkLogic server and this XDBC server will be used to process XQuery expressions against the server. You can change the XDBC application server that Oxygen XML Editor uses to process XQuery expressions by selecting the ***Use it to execute queries*** *action* from the contextual menu in the ***Data Source Explorer*** *view*.
3. Open the XQuery file and start the debugging process.
   - If you want to debug an XQuery file stored on the MarkLogic server, we recommend you to use the ***Data Source Explorer*** *view* and open the file from the application server that is involved in the debugging process. This improves the resolving of any imported modules.
   - The MarkLogic XQuery debugger integrates seamlessly into the *XQuery Debugger perspective*. If you have a MarkLogic validation scenario configured for the XQuery file, you can choose to *debug the scenario* directly.
   - Otherwise, switch to the **XQuery Debugger** *perspective*, open the XQuery file in the editor, and select the MarkLogic connection in the XQuery engine selector from the *debug control toolbar*.

     For general information about how a debugging session is started and controlled, see the *Working with the Debugger* section.

In a MarkLogic debugging session, you can use step actions and *breakpoints* to help identify problems. When you *add a breakpoint* on a line where the debugger never stops, Oxygen XML Editor displays a warning message. These warnings are displayed for *breakpoints* you add either in the main XQuery (which you can open locally or from the server) or for *breakpoints* you add in any XQuery that is opened from the connection that participates in the debugging session. For more information, see *Using Breakpoints for Debugging Queries that Import Modules with MarkLogic* on page 1098.

Remote Debugging with MarkLogic

Oxygen XML Editor allows you to debug remote applications that use XQuery (for example, web applications that trigger XQuery executions). Oxygen XML Editor connects to a MarkLogic server, shows you the running XQuery scripts and allows you to debug them. You can even pause the scripts so that you can start the debugging

queries in the exact context of the application. You can also switch a server to debug mode to intercept all XQuery scripts.

Oxygen XML Editor also supports collaborative debugging. This feature allows multiple users to participate in the same debugging session. You can start a debugging session and at a certain point, another user can continue it.

Important: When using the remote debugging feature, the HTTP and the XDBC servers involved in the debugging session must have the same module configuration.

To watch our video demonstration about the XQuery debugger for MarkLogic, go to *https://www.oxygenxml.com/demo/XQueryDebuggerforMarkLogic.html*.

Related Information:

MarkLogic Development in Oxygen XML Editor on page 1096
Configuring a MarkLogic Database Connection on page 1095

Using Breakpoints for Debugging Queries that Import Modules with MarkLogic

When debugging queries that imports modules stored in the database, it is recommended to place *breakpoints* in the modules. When starting a new debugging session, make sure that the modules that you will debug are already opened in the editor. This is necessary so that the *breakpoints* in all the modules will be considered. Also, make sure that there are no other opened modules that are not involved in the current debugging session.

To place *breakpoints* in the modules, use the following procedure:

1. In the ***Data Source Explorer*** *view*, open all the modules from the **Modules** container of the *XDBC application server* that performs the debugging.
2. *Set breakpoints* in the module as needed.
3. *Continue debugging* the query.

If you get a warning that the *breakpoints* failed to initialize, try the following solutions:

- Check the ***Breakpoints*** *view* and make sure there are no older *breakpoints* (set on resources that are not part of the current debugging context).
- Make sure you open the modules from the context of the application server that does the debugging and place *breakpoints* there.

Related Information:

MarkLogic Database Connections on page 1094
MarkLogic Development in Oxygen XML Editor on page 1096

Peculiarities and Limitations of the MarkLogic Debugger

MarkLogic debugger has the following peculiarities and limitations:

- Debugging support is only available for MarkLogic server versions 4.0 or newer.
- For MarkLogic server versions 4.0 or newer, there are three XQuery syntaxes that are supported: *'0.9-ml' (inherited from MarkLogic 3.2)*, *'1.0-ml'*, and *'1.0'*.
- All declared variables are presented as strings. The **Value** column of the **Variables** view contains the expression from the variable declaration. It can be evaluated by copying the expression with the **Copy value** action from the contextual menu of *the* ***Variables*** *view* and pasting it in *the* ***XWatch*** *view*.
- There is no support for *output to source mapping*.
- There is no support for *showing the trace*.
- You can only set *breakpoints* in imported modules in one of the following cases:
  - When you open the module from the context of the application server involved in the debugging, using the ***Data Source Explorer*** *view*.
  - When the debugger automatically opens the modules in the Editor.
- No *breakpoints* are set in modules from the same server that are not involved in the current debugging session.
- No support for *profiling* when an XQuery transformation is executed in the debugger.

MarkLogic Contextual Menu Actions

While browsing MarkLogic connections in the ***Data Source Explorer*** *view*, the various nodes include the following contextual menu actions:

Connection Level Nodes

Configure Database Sources

Opens the ***Data Sources*** *preferences page* where you can configure both data sources and connections.

Disconnect

Stops the connection.

Refresh

Performs a refresh on the selected node.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Container Level Nodes

Enable Debug Mode

Switches the server to a debugging mode. For more information, see *MarkLogic debugging sessions*.

Use it to Execute Queries

The server will be used to process XQuery expressions against it.

Refresh

Performs a refresh on the selected node.

Module or Folder Level Nodes

Export

Allows you to export the folder on the remote connection to a local folder.

Refresh

Performs a refresh on the selected node.

Requests Level Nodes

Refresh

Performs a refresh on the selected node.

Cancel all requests

Cancels all queries that are either running or stopped on the application server. You can use this action to clean up the entire **Requests** container at the end of your sessions.

Resource Level Nodes

Open

Opens the selected resource in the editor.

Open in System Application

When you use this action, Oxygen XML Editor downloads the selected resource to a local temporary folder and opens the selected resource in the system application that is currently set as the default application associated with that type of resource. You can then edit the resource, save it, and when you switch the focus back to the **Data Source Explorer** view, Oxygen XML Editor will detect that there was a change and will ask if you want to upload the edited resource to the server.

Copy location

Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

Refresh

Performs a refresh on the selected node.

Compare

Compares two selected resources using the ***Compare Files*** *tool*.

Related Information:

Configuring a MarkLogic Database Connection on page 1095
MarkLogic Development in Oxygen XML Editor on page 1096
Debugging with MarkLogic on page 1097

Documentum xDB (X-Hive/DB) 10 Database Connections

Oxygen XML Editor includes support for Documentum xDB (X-Hive/DB) 10 database connections. Oxygen XML Editor allows you to browse the structure of a Documentum xDB (X-Hive/DB) 10 database in the ***Data Source Explorer*** *view* and perform various operations on the resources in the repository.

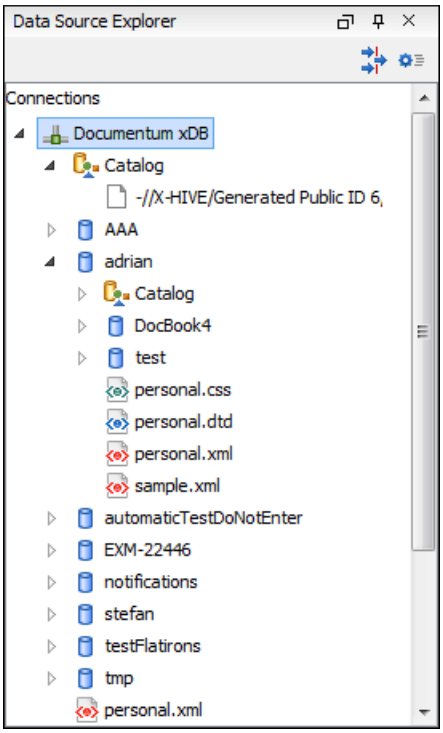


Figure 519: Documentum xDB (X-Hive/DB) 10 Connection

Configuring a Documentum xDB (X-Hive/DB) 10 Database Connection

Follow this procedure to configure the support for a Documentum xDB (X-Hive/DB) 10 database:

1. *Configure Documentum xDB Data Source drivers*.
2. *Configure a Documentum xDB Connection*.
3. To view your connection, go to the ***Data Source Explorer*** *view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

How to Configure Documentum xDB (X-Hive/DB) 10 Data Source Drivers

Available in the Enterprise edition only. For this procedure, you need to already have a Documentum xDB server installed.

To configure a data source for Documentum xDB (X-Hive/DB) 10, follow these steps:

1. *Open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Data Sources**.
2. Click the **New** button in the **Data Sources** panel.
3. Enter a unique name for the data source.

4. Select *Documentum xDB* from the driver **Type** drop-down menu.
5. Click the **Add** button to add the driver files.

   The driver files for the Documentum xDB (X-Hive/DB) 10 database are found in the Documentum xDB (X-Hive/DB) 10 lib directory from the server installation folder:

   - `antlr-runtime.jar`
   - `aspectjrt.jar`
   - `icu4j.jar`
   - `xhive.jar`
   - `google-collect.jar`
6. Click the **OK** button to finish the data source configuration.
7. Continue on to *configure your Documentum xDB (X-Hive/DB) 10 connection*.

How to Configure an Documentum xDB (X-Hive/DB) 10 Connection

To configure a connection to a Documentum xDB (X-Hive/DB) 10 database, follow these steps:

Note: The bootstrap type of X-Hive/DB connections is not supported in Oxygen XML Editor. The following procedure explains the `xhive://` protocol connection type.

1. *Open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Data Sources**.
2. Click the **New** button in the **Connections** panel.
3. Enter a unique name for the connection.
4. Select a previously configured data source from the **Data Source** drop-down menu.
5. Enter the connection details.
   a) Set the URL property of the connection in the **URL** field.

      If the property is a URL of the form *xhive://host:port*, the Documentum xDB (X-Hive/DB) 10 connection will attempt to connect to a Documentum xDB (X-Hive/DB) 10 server running behind the specified TCP/IP port.
   b) Set the user name to access the Documentum xDB (X-Hive/DB) 10 engine in the **User** field.
   c) Set the password to access the Documentum xDB (X-Hive/DB) 10 engine in the **Password** field.
   d) Set the name of the database to access from the Documentum xDB (X-Hive/DB) 10 engine in the **Database** field.
   e) Select the **Run XQuery in read / write session (with committing)** checkbox if you want to end the session with a commit. Otherwise, the session ends with a rollback.
6. Click the **OK** button to finish the connection configuration.
7. To view your connection, go to the ***Data Source Explorer** view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

Documentum xDB (X-Hive/DB) 10 Contextual Menu Actions

While browsing Documentum xDB (X-Hive/DB) 10 connections in the ***Data Source Explorer** view*, the various nodes include the following contextual menu actions:

Connection Level Nodes

Configure Database Sources

Opens the ***Data Sources** preferences page* where you can configure both data sources and connections.

Disconnect

Stops the connection.

Add Library

Allows you to add a new library.

Insert XML Instance

Allows you to add a new XML resource directly into the database root. See *Documentum xDB (X-Hive/DB) 10 Parser Configuration* for more details.

Insert non-XML Instance

Allows you to add a new non-XML resource directly in the database root.

Refresh

Performs a refresh on the selected node.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Catalog Level Nodes

Add AS Models

Allows you to add a new abstract schema model to the selected catalog.

Export

Allows you to export the folder on the remote connection to a local folder.

Set Default Schema

Allows you to set a default DTD to be used for parsing. It is not possible to set a default XML Schema.

Clear Default Schema

Allows you to clear the default DTD. The action is available only if there is a DTD set as default.

Refresh

Performs a refresh on the selected node.

Properties

Shows various properties of the current container.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Container (Library) Level Nodes

New File

Creates a new file on the connection, in the current folder.

Add Library

Allows you to add a new library.

Import Folders

Imports folders on the server.

Add Local Catalog

Adds a catalog to the selected library. By default, only the root-library has a catalog, and all models are stored there.

Insert XML Instance

Allows you to add a new XML resource directly into the database root. See *Documentum xDB (X-Hive/DB) 10 Parser Configuration* for more details.

Insert non-XML Instance

Allows you to add a new non-XML resource directly in the database root.

Export

Allows you to export the folder on the remote connection to a local folder.

Cut

Removes the current selection and places it in the clipboard.

Copy

Copies the current selection into the clipboard.

Paste

Pastes the copied selection.

Rename

Renames the current resource

Delete

Deletes the current container.

Refresh

Performs a refresh on the selected node.

Properties

Shows various properties of the current container.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Resource Level Nodes

Open

Opens the selected resource in the editor.

Open in System Application

When you use this action, Oxygen XML Editor downloads the selected resource to a local temporary folder and opens the selected resource in the system application that is currently set as the default application associated with that type of resource. You can then edit the resource, save it, and when you switch the focus back to the **Data Source Explorer** view, Oxygen XML Editor will detect that there was a change and will ask if you want to upload the edited resource to the server.

Save As

Allows you to save the selected resource as a file on disk.

Cut

Removes the current selection and places it in the clipboard.

Copy

Copies the current selection into the clipboard.

Copy location

Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

Add AS model

Allows you to add an XML schema to the selected XML resource.

Set AS model

Allows you to set an active AS model for the selected XML resource.

Clear AS model

Allows you to clear the active AS model of the selected XML resource.

Properties

Displays the resource properties. Available only for XML resources.

Set Default Schema

Allows you to set the selected DTD to be used as default for parsing. The action is available only for DTD.

Clear Default Schema

Allows you to unset the selected DTD. The action is available only if the selected DTD is the current default to be used for parsing.

Rename

Renames the current resource

Delete

Deletes the current container.

Refresh

Performs a refresh on the selected node.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Compare

Compares two selected resources using the ***Compare Files*** *tool*.

Documentum xDB (X-Hive/DB) 10 Parser Configuration for Adding XML Instances

When an XML instance document is added to a Documentum xDB (X-Hive/DB) 10 connection or library, it is parsed with an internal XML parser of the database server. The following options are available for configuring this parser:

- DOM Level 3 parser configuration parameters. More about each parameter can be found here: *DOM Level 3 Configuration*.
- Documentum xDB (X-Hive/DB) 10 specific parser parameters (for more information, consult the Documentum xDB (X-Hive/DB) 10 manual):
  - **xhive-store-schema** - If selected, the corresponding DTD or XML schemas are stored in the catalog during validated parsing.
  - **xhive-store-schema-only-internal-subset** - Stores only the internal sub-set of the document (not an external sub-set). This option modifies the **xhive-store-schema** (only has a function when that parameter is set to true, and when a DTD is involved). Select this option if you only want to store the internal sub-set of the document (not the external sub-set).
  - **xhive-ignore-catalog** - Ignores the corresponding DTD and XML schemas in the catalog during validated parsing.
  - **xhive-psvi** - Stores **psvi** information about elements and attributes. Documents parsed with this feature turned on give access to **psvi** information and enable support of data types by XQuery queries.
  - **xhive-sync-features** - With this convenience setting turned on, parameter settings of `XhiveDocumentIf` are synchronized with the parameter settings of `LSParser`. Note that parameter settings **xhive-psvi** and **schema-location** are always synchronized.

Troubleshooting Documentum xDB

Question:

I am able to access my XML Database in the ***Data Source Explorer*** and open files for reading but when I try to save changes to a file back into the database, I receive the following error: "**Cannot save the file. DTD factory class org.apache.xerces.impl.dv.dtd.DTDDVFactoryImpl does not extend from DTDDVFactory**". How can I fix this?

Answer:

`xhive.jar` contains a MANIFEST.MF with a classpath:

```
Class-Path: core/antlr-runtime.jar core/aspectjrt.jar core/fastutil-shrinked.jar
    core/google-collect.jar core/icu4j.jar core/lucene-regex.jar core/lucene.jar
    core/serializer.jar core/xalan.jar core/xercesImpl.jar
```

Since the driver was configured to use `xhive.jar` directly from the *xDB* installation (where many other *JARS* are located), `core/xercesImpl.jar` from the *xDB* installation directory is loaded even though it is not specified in the list of *JARS* from the data source driver configuration (it is in the classpath from `xhive.jar` - `MANIFEST.MF`). A simple workaround for this issue is to copy **ONLY** the *JAR* files used in the driver configuration to a separate folder and configure the data source driver to use them from there.

MySQL Database Connections

Oxygen XML Editor includes support for MySQL database connections. Oxygen XML Editor allows you to browse the structure of a SQL Server database in the ***Data Source Explorer*** *view*, open tables in the ***Table Explorer*** *view*, and perform various operations on the resources in the repository.

Configuring a MySQL Database Connection

To configure the support for a MySQL database, follow this procedure:

1. *Configure MySQL Data Source drivers*.
2. *Configure a MySQL Connection.*
3. To view your connection, go to the ***Data Source Explorer*** *view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

How to Configure MySQL Data Source Drivers

To connect to a MySQL server, you need to create a generic JDBC type data source based on *the MySQL JDBC driver available on the MySQL website*.

To configure this data source, follow these steps:

1. Go to *https://www.oxygenxml.com/database_drivers.html* and download the appropriate MySQL driver.
2. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Data Sources**.
3. Click the **+ New** button in the **Data Sources** panel.

   The dialog box for configuring a data source is opened.

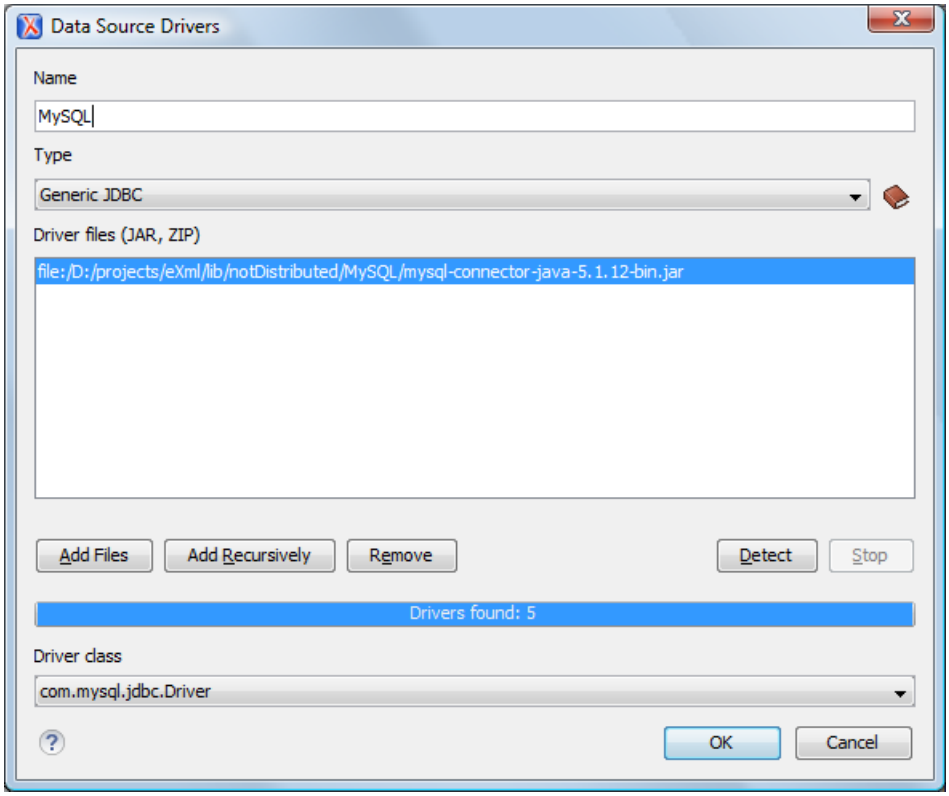


Figure 520: Data Source Drivers Configuration Dialog Box

4. Enter a unique name for the data source.
5. Select *Generic JDBC* in the driver **Type** drop-down list.
6. Click the **Add Files** button and select the MySQL driver file that you downloaded.

   The driver file for the MySQL server is called `mysql-com.jar`.
7. Select the most appropriate **Driver class**.
8. Click the **OK** button to finish the data source configuration.

9. Continue on to *configure your MySQL connection*.

How to Configure a MySQL Connection

To configure a connection to a MySQL server, follow these steps:

1. *Open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Data Sources**.
2. Click the **New** button in the **Connections** panel.

   The dialog box for configuring a database connection is displayed.

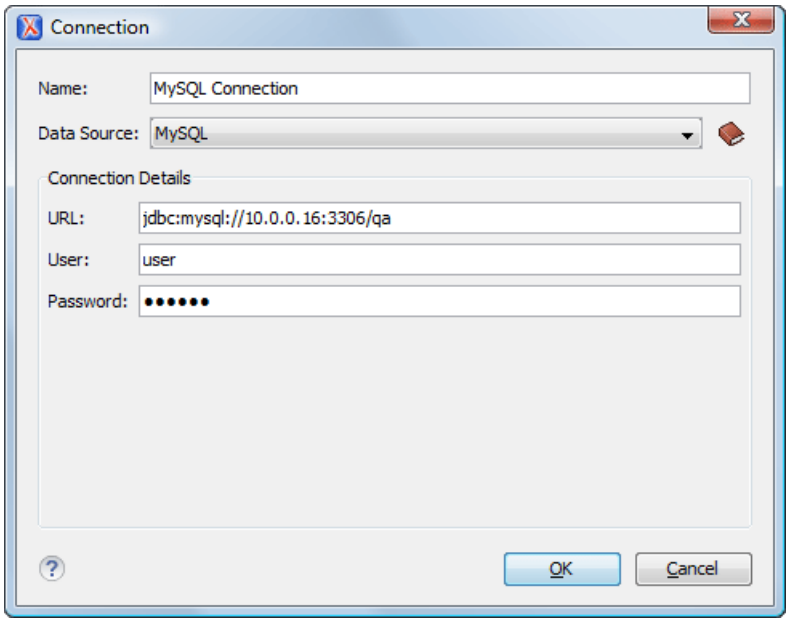


Figure 521: Connection Configuration Dialog Box

3. Enter a unique name for the connection.
4. Select the *MySQL* data source in the **Data Source** drop-down list.
5. Enter the connection details.
   a) Enter the URL of the MySQL server.
   b) Enter the user name for the connection to the MySQL server.
   c) Enter the password for the connection to the MySQL server.
6. Click the **OK** button to finish the connection configuration.
7. To view your connection, go to the ***Data Source Explorer** view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

Generic JDBC Database Connections

Oxygen XML Editor includes support for Generic JDBC database connections.

Configuring a Generic JDBC Database Connection

To configure the support for a generic JDBC database, follow this procedure:

1. *Configure Generic JDBC Data Source drivers*.
2. *Configure a Generic JDBC Connection*.
3. To view your connection, go to the ***Data Source Explorer** view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

How to Configure Generic JDBC Data Source Drivers

Starting with version 17, Oxygen XML Editor comes bundled with Java 8, which does not provide built-in access to JDBC-ODBC data sources. To access such sources, you need to find an alternative JDBC-ODBC bridge or use a platform-independent distribution of Oxygen XML Editor along with a Java VM version 7 or 6.

To configure a generic JDBC data source, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Data Sources**.
2. Click the **New** button in the **Data Sources** panel.
3. Enter a unique name for the data source.
4. Select *Generic JDBC* in the driver **Type** drop-down list.
5. Add the driver file(s) using the **Add Files** button.
6. Select the most appropriate **Driver class**.
7. Click the **OK** button to finish the data source configuration.
8. Continue on to *configure a generic JDBC connection*.

How to Configure a Generic JDBC Connection

To configure a connection to a generic JDBC database, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Data Sources**.
2. Click the **New** button in the **Connections** panel.
3. Enter a unique name for the connection.
4. Select the *Generic JDBC* data source in the **Data Source** drop-down menu.
5. Enter the connection details.
   a) Enter the URL of the generic JDBC database, with the following format:`jdbc: <subprotocol>: <subname>`.
   b) Enter the user name for the connection to the generic JDBC database.
   c) Enter the password for the connection to the generic JDBC database.
6. Click the **OK** button to finish the connection configuration.
7. To view your connection, go to the ***Data Source Explorer** view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

JDBC-ODBC Database Connections

Oxygen XML Editor includes support for JDBC-ODBC database connections.

How to Configure a JDBC-ODBC Connection

Starting with version 17, Oxygen XML Editor comes bundled with Java 8, which does not provide built-in access to JDBC-ODBC data sources. To access such sources, you need to find an alternative JDBC-ODBC bridge or use a platform-independent distribution of Oxygen XML Editor along with a Java VM version 7 or 6.

To configure a connection to an ODBC data source, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Data Sources**.
2. Click the **New** button in the **Connections** panel.

   The dialog box for configuring a database connection is displayed.

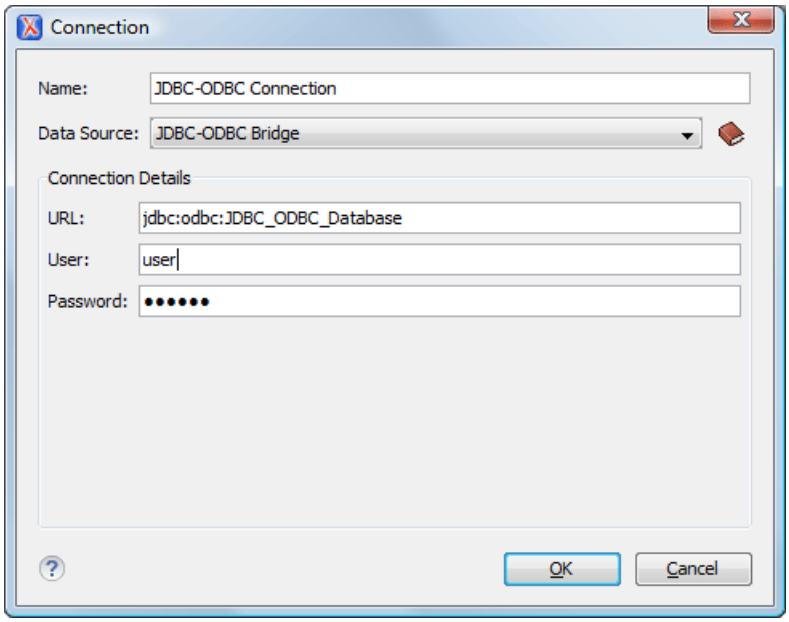


Figure 522: Connection Configuration Dialog Box

3. Enter a unique name for the connection.
4. Select *JDBC-ODBC Bridge* in the **Data Source** drop-down list.
5. Enter the connection details.
   a) Enter the URL of the ODBC source.
   b) Enter the user name of the ODBC source.
   c) Enter the password of the ODBC source.
6. Click the **OK** button to finish the connection configuration.
7. To view your connection, go to the ***Data Source Explorer*** *view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

BaseX Database Connections

Oxygen XML Editor includes support for BaseX database connections using a WebDAV connection. BaseX is a light-weight XML database engine and XQuery processor. Oxygen XML Editor allows you to browse the structure of a BaseX database in the ***Data Source Explorer*** *view* and perform XQuery executions.

How to Configure a BaseX Connection

To configure a BaseX connection, follow these steps:

1. First of all, make sure the BaseX HTTP Server is started. For details about starting the BaseX HTTP server, go to *http://docs.basex.org/wiki/Startup#BaseX_HTTP_Server*. The configuration file for the HTTP server is named `.basex` and is located in the BaseX installation directory. This file helps you to find out which port the HTTP server using. The default port for BaseX WebDAV is 8984.
2. To ensure that everything is functioning, open a WebDAV URL inside a browser and check to see if it works. For example, the following URL retrieves a document from a database named TEST: `http://localhost:8984/webdav/TEST/etc/factbook.xml`.
3. Once you are sure that the BaseX WebDAV service is working, you can configure the WebDAV connection in Oxygen XML Editor as described in *How to Configure a WebDAV Connection* on page 1111. The WebDAV URL should resemble this: `http://{hostname }:{port}/webdav/`. If the BaseX server is running on your own machine and it has the default configuration, the data required by the WebDAV connection is:
   - WebDAV URL: `http://localhost:8984/webdav`
   - User: `admin`
   - Password: `admin`

4. Once the WebDAV connection is created, to view your connection, go to the ***Data Source Explorer*** *view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

BaseX Contextual Menu Actions

While browsing BaseX connections in the ***Data Source Explorer*** *view*, the various nodes include the following contextual menu actions:

Connection Level Nodes

Configure Database Sources

Opens the ***Data Sources*** *preferences page* where you can configure both data sources and connections.

Disconnect

Stops the connection.

New Folder

Creates a new folder on the connection.

Import Files

Allows you to add a new file on the connection, in the current folder.

Refresh

Performs a refresh on the selected node.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Folder Level Nodes

New File

Creates a new file on the connection, in the current folder.

New Folder

Creates a new folder on the connection.

Import Folders

Imports folders on the server.

Import Files

Allows you to add a new file on the connection, in the current folder.

Export

Allows you to export the folder on the remote connection to a local folder.

Cut

Removes the current selection and places it in the clipboard.

Copy

Copies the current selection into the clipboard.

Paste

Pastes the copied selection.

Rename

Renames the current resource

Delete

Deletes the current container.

Refresh

Performs a refresh on the selected node.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Resource Level Nodes

Open

Opens the selected resource in the editor.

Open in System Application

When you use this action, Oxygen XML Editor downloads the selected resource to a local temporary folder and opens the selected resource in the system application that is currently set as the default application associated with that type of resource. You can then edit the resource, save it, and when you switch the focus back to the **Data Source Explorer** view, Oxygen XML Editor will detect that there was a change and will ask if you want to upload the edited resource to the server.

Cut

Removes the current selection and places it in the clipboard.

Copy

Copies the current selection into the clipboard.

Copy location

Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

Rename

Renames the current resource

Delete

Deletes the current container.

Refresh

Performs a refresh on the selected node.

Properties

Shows various properties of the current container.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Compare

Compares two selected resources using the ***Compare Files*** *tool*.

Base X XQJ Connection

XQuery execution is possible in a BaseX connection through an XQJ connection.

BaseX XQJ Data Source

First of all, create an XQJ data source as described in *How to Configure an XQJ Data Source* on page 1111.
The BaseX XQJ API-specific files that must be added in the configuration dialog box are `xqj-api-1.0.jar`, `xqj2-0.1.0.jar` and `basex-xqj-1.2.3.jar` (the version names of the *JAR* file may differ). These libraries can be downloaded from *`xqj.net/basex/basex-xqj-1.2.3.zip`*. As an alternative, you can also find the libraries in the BaseX installation directory, in the **lib** sub-directory.

BaseX XQJ Connection

The next step is to create an *XQJ connection*.

For a default BaseX configuration, the following connection details apply (you can modify them when necessary):

- **Port**: 1984

- **serverName**: localhost
- **user**: admin
- **password**: admin

XQuery Execution

Now that the XQJ connection is configured, open the XQuery file you want to execute in Oxygen XML Editor and create an *XQuery Transformation* on page 919. In the **Transformer** drop-down menu, select the name of the XQJ connection you created. Apply the transformation scenario and the XQuery will be executed.

How to Configure an XQJ Data Source

Any transformer that offers an XQJ API implementation can be used when validating XQuery or transforming XML documents. An example of an XQuery engine that implements the XQJ API is *Zorba*.

1. If your XQJ Implementation is native, make sure the directory containing the native libraries of the engine is added to your system environment variables: to PATH - on Windows, to LD_LIBRARY_PATH - on Linux, or to DYLD_LIBRARY_PATH - on OS X. Restart Oxygen XML Editor after configuring the environment variables.
2. *Open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Data Sources**.
3. Click the **New** button in the **Data Sources** panel.
4. Enter a unique name for the data source.
5. Select **XQuery API for Java (XQJ)** in the **Type** combo box.
6. Press the **Add** button to add XQJ API-specific files.

   You can manage the driver files using the **Add**, **Remove**, **Detect**, and **Stop** buttons.

   Oxygen XML Editor detects any implementation of `javax.xml.xquery.XQDataSource` and presents it in **Driver class** field.
7. Select the most suited driver in the **Driver class** combo box.
8. Click the **OK** button to finish the data source configuration.
9. Continue on to *configure the XQJ connection*.

How to Configure an XQJ Connection

The steps for configuring an XQJ connection are the following:

1. *Open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Data Sources**.
2. Click the **New** button in the **Connections** panel.
3. Enter a unique name for the connection.
4. Select one of the previously configured *XQJ data sources* in the **Data Source** combo box.
5. Fill-in the connection details.

   The properties presented in the connection details table are automatically detected depending on the selected data source.
6. Click the **OK** button to finish the connection configuration.

WebDAV Connections

Oxygen XML Editor includes support for WebDAV server connections. Oxygen XML Editor allows you to browse the structure of a WebDAV connection in the ***Data Source Explorer** view* and perform various operations on the resources in the repository.

How to Configure a WebDAV Connection

By default, Oxygen XML Editor contains built-in data source drivers for **WebDAV** connections. Based on this data source, you can create a WebDAV connection for browsing and editing data from a database that provides a WebDAV interface. The connection is available in the ***Data Source Explorer** view*.

To configure a WebDAV connection, follow these steps:

1. *Open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Data Sources**.

2. Click the **New** button in the **Connections** panel.
3. Enter a unique name for the connection.
4. Select one of the WebDAV data sources in the **Data Source** drop-down menu.
5. Enter the connection details:
   a) Set the URL to the WebDAV repository in the field **WebDAV URL**.
   b) Set the user name that is used to access the WebDAV repository in the **User** field.
   c) Set the password that is used to access the WebDAV repository in the **Password** field.
6. Click the **OK** button to finish the connection configuration.
7. To view your connection, go to the ***Data Source Explorer*** *view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

   To watch our video demonstration about the WebDAV support in Oxygen XML Editor, go to *https://www.oxygenxml.com/demo/WebDAV_Support.html*.

WebDAV Contextual Menu Actions

While browsing WebDAV connections in the ***Data Source Explorer*** *view*, the various nodes include the following contextual menu actions:

Connection Level Nodes

Configure Database Sources

Opens the ***Data Sources*** *preferences page* where you can configure both data sources and connections.

Disconnect

Stops the connection.

New Folder

Creates a new folder on the connection.

Import Files

Allows you to add a new file on the connection, in the current folder.

Refresh

Performs a refresh on the selected node.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Folder Level Nodes

New File

Creates a new file on the connection, in the current folder.

New Folder

Creates a new folder on the connection.

Import Folders

Imports folders on the server.

Import Files

Allows you to add a new file on the connection, in the current folder.

Export

Allows you to export the folder on the remote connection to a local folder.

Cut

Removes the current selection and places it in the clipboard.

Copy

Copies the current selection into the clipboard.

Paste

Pastes the copied selection.

Rename

Renames the current resource

Delete

Deletes the current container.

Refresh

Performs a refresh on the selected node.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Resource Level Nodes

Open

Opens the selected resource in the editor.

Open in System Application

When you use this action, Oxygen XML Editor downloads the selected resource to a local temporary folder and opens the selected resource in the system application that is currently set as the default application associated with that type of resource. You can then edit the resource, save it, and when you switch the focus back to the **Data Source Explorer** view, Oxygen XML Editor will detect that there was a change and will ask if you want to upload the edited resource to the server.

Cut

Removes the current selection and places it in the clipboard.

Copy

Copies the current selection into the clipboard.

Copy location

Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

Rename

Renames the current resource

Delete

Deletes the current container.

Refresh

Performs a refresh on the selected node.

Properties

Shows various properties of the current container.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Compare

Compares two selected resources using the ***Compare Files*** *tool*.

SQL Execution Support

The database support in Oxygen XML Editor includes support for writing SQL statements, syntax highlighting, *folding*, and dragging and dropping from the ***Data Source Explorer*** *view*. It also includes transformation scenarios for executing the statements, and the results are displayed in the ***Table Explorer*** *view*.

Drag and Drop from Data Source Explorer View

Drag and drop operations from the ***Data Source Explorer*** *view* to the SQL Editor allows you to create SQL statements quickly by inserting the names of tables and columns in the SQL statements.

1. Configure a database connection (see the specific procedure for your database server in the *Database Connection Support* section).
2. Browse to the table you will use in your statement.
3. Drag the table or a column of the table into the editor where a SQL file is open.

   Drag and drop actions are available both on the table and on its fields. A pop-up menu is displayed in the SQL editor.

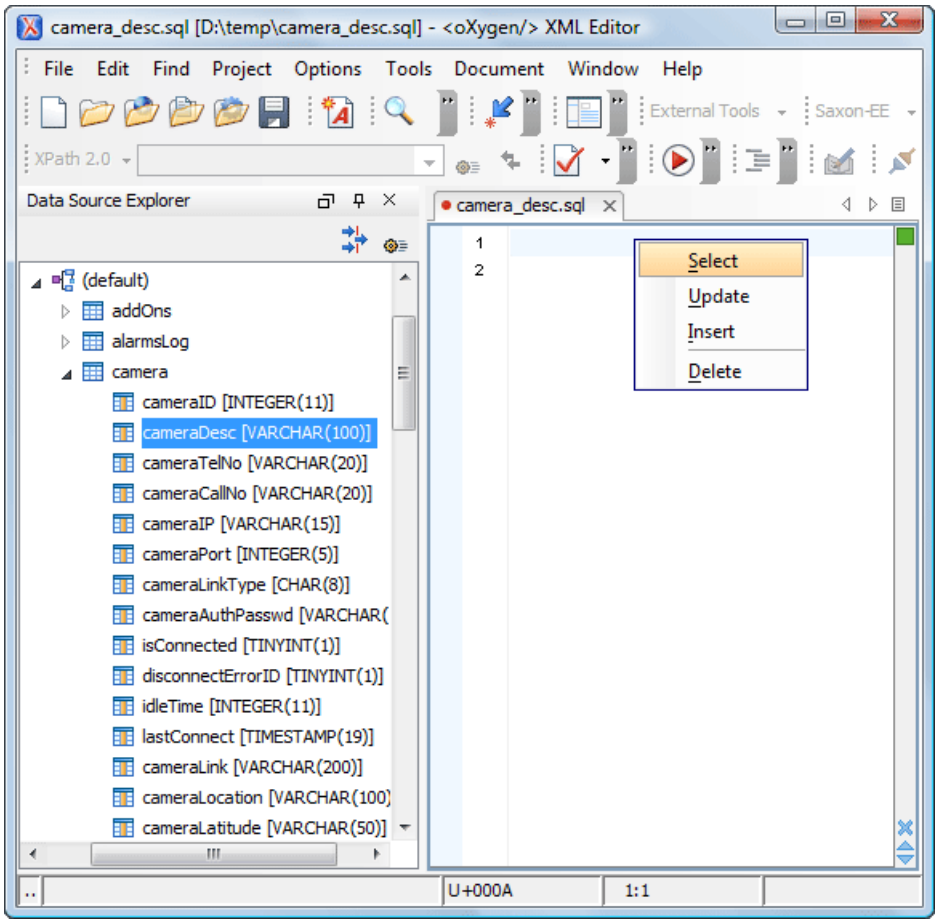


Figure 523: SQL Statement Editing with Drag and Drop

4. Select the type of statement from the pop-up menu.

   Depending on your choice, dragging a table results in one of the following statements being inserted into the document:

   - SELECT `*field1*`,`*field2*`,.... FROM `*catalog*`.`*table*` (for example: `SELECT `DEPT`,`DEPTNAME`,`LOCATION` FROM `camera`.`cameraDesc`)`
   - UPDATE `*catalog*`.`*table*` SET `*field1*`=,`*field2*`=,.... (for example: `UPDATE `camera`.`cameraDesc` SET `DEPT`=, `DEPTNAME`=, `LOCATION`=)`
   - INSERT INTO`*catalog*`.`*table*` (`*field1*`,`*field2*`,....) VALUES (, ,) (for example: `INSERT INTO `camera`.`cameraDesc` (`DEPT`,`DEPTNAME`,`LOCATION`) VALUES (, , ))`
   - DELETE FROM `*catalog*`.`*table*` (for example: `DELETE FROM `camera`.`cameraDesc`)`

   Depending on your choice, dragging a column results in one of the following statements being inserted into the document:

- SELECT `*field*` FROM `*catalog*`.`*table*` (for example: `SELECT `DEPT` FROM `camera`.`cameraDesc``)
- UPDATE `*catalog*`.`*table*` SET `*field*`= (for example: `UPDATE `camera`.`cameraDesc` SET `DEPT`=`)
- INSERT INTO`*catalog*`.`*table*` (`*field1*`) VALUES () (for example: `INSERT INTO `camera`.`cameraDesc` (`DEPT`) VALUES ()`)
- DELETE FROM `*catalog*`.`*table*` (for example: `DELETE FROM `camera`.`cameraDesc` WHERE `DEPT`=`)

SQL Validation

SQL validation support is offered for IBM DB2. Note that if you choose a connection that does not support SQL validation, you will receive a warning when trying to validate. The SQL document is validated using the connection from the associated transformation scenario.

Executing SQL Statements

The steps for executing an SQL statement on a relational database are as follows:

1. Configure a *transformation scenario* using the **Configure Transformation Scenario(s)** action from the toolbar or the **Document** > **Transformation** menu.

   A SQL transformation scenario needs a database connection. You can configure a connection using the **Preferences** button from the SQL transformation dialog box.

   The dialog box contains the list of existing scenarios that apply to SQL documents.
2. Set parameter values for SQL placeholders using the **Parameters** button from the SQL transformation dialog box.

   For example, in `SELECT * FROM `test`.`department` where DEPT = ? or DEPTNAME = ?` the two parameters can be configured for the place holders (?) in the transformation scenario.

   When the SQL statement is executed, the first placeholder is replaced with the value set for the first parameter in the scenario, the second placeholder is replaced by the second parameter value, and so on.

   Restriction: When a stored procedure is called in an SQL statement executed on an SQL Server database, mixing inline parameter values with values specified using the **Parameters** button of the scenario dialog box is not recommended. This is due to a limitation of the SQL Server driver for Java applications. An example of stored procedure that is not recommended: `call dbo.Test(22, ?)`.
3. Execute the SQL scenario by clicking the **OK** or **Apply associated** button.

   The result of a SQL transformation is *displayed in a view* at the bottom of the Oxygen XML Editor window.
4. View more complex return values of the SQL transformation in a separate editor panel.

   A more complex value returned by the SQL query (for example, an *XMLTYPE* or *CLOB* value) cannot be displayed entirely in the result table.

   a) Right-click the cell containing the complex value.
   b) Select the action **Copy cell** from the contextual menu.
      The action copies the value in the clipboard.
   c) Paste the value into an appropriate editor.

      For example, you can paste the value in an opened XQuery editor panel of Oxygen XML Editor.

XQuery and Databases

XQuery is a native XML query language that is useful for querying XML views of relational data to create XML results. It also provides the mechanism to efficiently and easily extract information from Native XML Databases (NXD) and relational data. The following database systems supported in Oxygen XML Editor offer XQuery support:

- *Native XML Databases:*
  - Berkeley DB XML
  - eXist
  - MarkLogic (validation support available starting with version 5)

- Documentum xDB (X-Hive/DB) 10
- *Relational Databases:*
  - IBM DB2
  - Microsoft SQL Server (validation support not available)
  - Oracle (validation support not available)

Related Information:

Editing XQuery Documents on page 618

Build Queries with Drag and Drop from the Data Source Explorer View

When a query is edited in the XQuery editor, the XPath expressions can be composed quickly by dragging them from the ***Data Source Explorer*** *view* and dropping them into the editor panel.

1. *Configure the data source drivers* for the particular relational database in the ***Data Sources*** *preferences page*.
2. *Configure the connection* for the particular relational database in the ***Data Sources*** *preferences page*.
3. Browse the connection in the ***Data Source Explorer*** *view*, expanded to the table or column that you want to insert in the query.
4. Drag the table or column name to the XQuery editor panel.
5. Drop the table or column name where the XPath expression is needed.

An XPath expression that selects the dragged name is inserted in the XQuery document at the cursor position.

XQuery Validation When Connected to a Database

With Oxygen XML Editor, you can validate your XQuery documents when connected to a database. When you open an XQuery document from a connection that supports validation (for example, MarkLogic, or eXist), by default Oxygen XML Editor uses this connection for validation. If you open an XQuery file using a MarkLogic connection, the validation resolves imports better.

Related Information:

XQuery Validation on page 618

XQuery Transformation for Databases

XQuery is designed to retrieve and interpret XML data from any source, whether it is a database or document. Data is stored in relational databases but it is often required that the data be extracted and transformed as XML when interfacing to other components and services. Also, it is an XPath-based querying language supported by most NXD vendors. To perform a query, you need an XQuery transformation scenario.

1. *Configure the data source drivers and the connection* for the particular database.
2. Configure an XQuery transformation scenario.
   a) Click the **Configure Transformation Scenario** toolbar button or go to menu **Document** > **Transformation** > **Configure Transformation Scenario**.

      The ***Configure Transformation Scenario*** *dialog box* is opened.
   b) Click the **New** button toward the bottom of the dialog box.
   c) Select ***XML Transformation with XQUERY***.

      The **New Scenario** dialog box for configuring an XQuery scenario is opened.

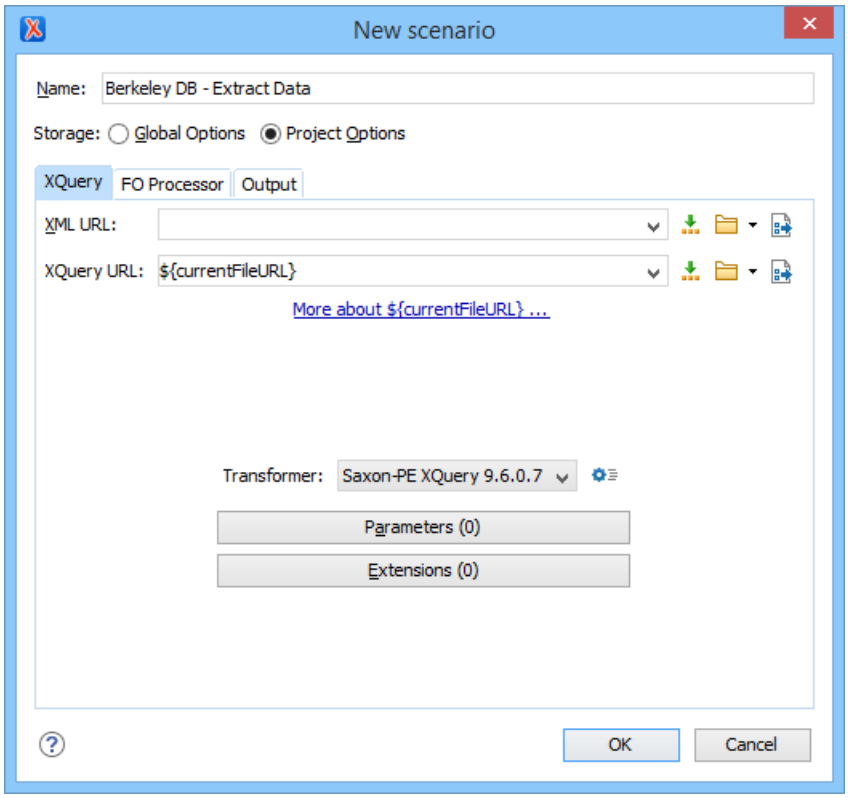


Figure 524: New Scenario Dialog Box

d) Insert the scenario name in the dialog box for editing the scenario.
e) Choose the database connection in the **Transformer** drop-down list.
f) Configure any other parameters as needed.

   For an XQuery transformation, the output tab has an option called **Sequence** that allows you to run an XQuery in lazy mode. The amount of data extracted from the database is controlled from the ***Size limit on Sequence view*** *option* in the **XQuery** preferences page. If you choose **Perform FO Processing** in the **FO Processor** tab, the **Sequence** option is ignored.
g) Click the **OK** button to finish editing the scenario.

Once the scenario is associated with the XQuery file, the query can include calls to specific XQuery functions that are implemented by that engine. The available functions depend on the target database engine selected in the scenario. For example, for eXist and Berkeley DB XML, the *Content Completion Assistant* lists the functions supported by that database engine. This is useful for only inserting calls to the supported functions (standard XQuery functions or extension ones) into the query .

Note: An XQuery transformation is executed against a Berkeley DB XML server as a transaction using the query transaction support of the server.

3. Run the transformation scenario.

To view a more complex value returned by the query that cannot be entirely displayed in the XQuery query result table at the bottom of the Oxygen XML Editor window (for example, an XMLTYPE or CLOB value), do the following:

- Right-click that table cell.
- Select the **Copy cell** action from the contextual menu to copy the value into the clipbord.
- Paste the value wherever you need it (for example, in an opened XQuery editor panel of Oxygen XML Editor).

Related Information:

XML Transformation with XQuery on page 893
This type of transformation specifies the transform parameters and location of an XQuery file that is applied to the edited XML document.

XQuery XQJ Transformation on page 1118

XQuery XQJ Transformation

XQuery API for Java (XQJ) refers to the common Java API for the W3C XQuery 1.0 specification. The XQJ API enables you to execute XQuery against an XML data source.

Oxygen XML Editor supports any transformer that offers an XQJ API implementation and it be used for validating XQuery or transforming XML documents.

To configure the support for XQJ, do the following:

1. *Configure an XQJ Data Source*.
2. *Configure an XQJ Connection*.
3. To view your connection, go to the ***Data Source Explorer** view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

How to Configure an XQJ Data Source

Any transformer that offers an XQJ API implementation can be used when validating XQuery or transforming XML documents. An example of an XQuery engine that implements the XQJ API is *Zorba*.

1. If your XQJ Implementation is native, make sure the directory containing the native libraries of the engine is added to your system environment variables: to PATH - on Windows, to LD_LIBRARY_PATH - on Linux, or to DYLD_LIBRARY_PATH - on OS X. Restart Oxygen XML Editor after configuring the environment variables.
2. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Data Sources**.
3. Click the **+ New** button in the **Data Sources** panel.
4. Enter a unique name for the data source.
5. Select **XQuery API for Java (XQJ)** in the **Type** combo box.
6. Press the **Add** button to add XQJ API-specific files.

   You can manage the driver files using the **Add**, **Remove**, **Detect**, and **Stop** buttons.

   Oxygen XML Editor detects any implementation of `javax.xml.xquery.XQDataSource` and presents it in **Driver class** field.
7. Select the most suited driver in the **Driver class** combo box.
8. Click the **OK** button to finish the data source configuration.
9. Continue on to *configure the XQJ connection*.

How to Configure an XQJ Connection

The steps for configuring an XQJ connection are the following:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Data Sources**.
2. Click the **+ New** button in the **Connections** panel.
3. Enter a unique name for the connection.
4. Select one of the previously configured *XQJ data sources* in the **Data Source** combo box.
5. Fill-in the connection details.

   The properties presented in the connection details table are automatically detected depending on the selected data source.
6. Click the **OK** button to finish the connection configuration.

XQuery Database Debugging

Oxygen XML Editor includes a debugging interface that helps you to detect and solve problems with XQuery transformations that are executed against MarkLogic and Berkeley DB XML databases.

For more information about the debugging support in Oxygen XML Editor, see *Debugging XSLT Stylesheets and XQuery Documents* on page 1142.

Debugging with MarkLogic

Oxygen XML Editor includes support for debugging XQuery transformations that are executed against a MarkLogic database.

To use a debugging session against the MarkLogic engine, follow these steps:

1. Configure a *MarkLogic data source* and a *MarkLogic connection*.
2. Make sure that the debugging support is enabled in the MarkLogic server that Oxygen XML Editor accesses. On the server side, debugging must be activated in the XDBC server and in the *Task Server* section of the server control console (the switch *debug allow*). If the debugging is not activated, the MarkLogic server reports a `DBG-TASKDEBUGALLOW` error.

   Note: An XDBC application server must be running to connect to the MarkLogic server and this XDBC server will be used to process XQuery expressions against the server. You can change the XDBC application server that Oxygen XML Editor uses to process XQuery expressions by selecting the ***Use it to execute queries*** *action* from the contextual menu in the ***Data Source Explorer*** *view*.
3. Open the XQuery file and start the debugging process.
   - If you want to debug an XQuery file stored on the MarkLogic server, we recommend you to use the ***Data Source Explorer*** *view* and open the file from the application server that is involved in the debugging process. This improves the resolving of any imported modules.
   - The MarkLogic XQuery debugger integrates seamlessly into the *XQuery Debugger perspective*. If you have a MarkLogic validation scenario configured for the XQuery file, you can choose to *debug the scenario* directly.
   - Otherwise, switch to the **XQuery Debugger** *perspective*, open the XQuery file in the editor, and select the MarkLogic connection in the XQuery engine selector from the *debug control toolbar*.

     For general information about how a debugging session is started and controlled, see the *Working with the Debugger* section.

In a MarkLogic debugging session, you can use step actions and *breakpoints* to help identify problems. When you *add a breakpoint* on a line where the debugger never stops, Oxygen XML Editor displays a warning message. These warnings are displayed for *breakpoints* you add either in the main XQuery (which you can open locally or from the server) or for *breakpoints* you add in any XQuery that is opened from the connection that participates in the debugging session. For more information, see *Using Breakpoints for Debugging Queries that Import Modules with MarkLogic* on page 1098.

Remote Debugging with MarkLogic

Oxygen XML Editor allows you to debug remote applications that use XQuery (for example, web applications that trigger XQuery executions). Oxygen XML Editor connects to a MarkLogic server, shows you the running XQuery scripts and allows you to debug them. You can even pause the scripts so that you can start the debugging queries in the exact context of the application. You can also switch a server to debug mode to intercept all XQuery scripts.

Oxygen XML Editor also supports collaborative debugging. This feature allows multiple users to participate in the same debugging session. You can start a debugging session and at a certain point, another user can continue it.

Important: When using the remote debugging feature, the HTTP and the XDBC servers involved in the debugging session must have the same module configuration.

To watch our video demonstration about the XQuery debugger for MarkLogic, go to *https://www.oxygenxml.com/demo/XQueryDebuggerforMarkLogic.html*.

Related Information:

MarkLogic Development in Oxygen XML Editor on page 1096

Configuring a MarkLogic Database Connection on page 1095

Using Breakpoints for Debugging Queries that Import Modules with MarkLogic

When debugging queries that imports modules stored in the database, it is recommended to place *breakpoints* in the modules. When starting a new debugging session, make sure that the modules that you will debug are already opened in the editor. This is necessary so that the *breakpoints* in all the modules will be considered. Also, make sure that there are no other opened modules that are not involved in the current debugging session.

To place *breakpoints* in the modules, use the following procedure:

1. In the ***Data Source Explorer*** *view*, open all the modules from the **Modules** container of the *XDBC application server* that performs the debugging.
2. *Set breakpoints* in the module as needed.
3. *Continue debugging* the query.

If you get a warning that the *breakpoints* failed to initialize, try the following solutions:

- Check the ***Breakpoints*** *view* and make sure there are no older *breakpoints* (set on resources that are not part of the current debugging context).
- Make sure you open the modules from the context of the application server that does the debugging and place *breakpoints* there.

Related Information:

MarkLogic Database Connections on page 1094

MarkLogic Development in Oxygen XML Editor on page 1096

Peculiarities and Limitations of the MarkLogic Debugger

MarkLogic debugger has the following peculiarities and limitations:

- Debugging support is only available for MarkLogic server versions 4.0 or newer.
- For MarkLogic server versions 4.0 or newer, there are three XQuery syntaxes that are supported: *'0.9-ml' (inherited from MarkLogic 3.2)*, *'1.0-ml'*, and *'1.0'*.
- All declared variables are presented as strings. The **Value** column of the **Variables** view contains the expression from the variable declaration. It can be evaluated by copying the expression with the **Copy value** action from the contextual menu of *the* ***Variables*** *view* and pasting it in *the* ***XWatch*** *view*.
- There is no support for *output to source mapping*.
- There is no support for *showing the trace*.
- You can only set *breakpoints* in imported modules in one of the following cases:
  - When you open the module from the context of the application server involved in the debugging, using the ***Data Source Explorer*** *view*.
  - When the debugger automatically opens the modules in the Editor.
- No *breakpoints* are set in modules from the same server that are not involved in the current debugging session.
- No support for *profiling* when an XQuery transformation is executed in the debugger.

Debugging with Berkeley DB XML

The Berkeley DB XML database added a debugging interface starting with version 2.5. The current version is supported in the Oxygen XML Editor XQuery Debugger. *The same restrictions and peculiarities* apply for the Berkeley debugger as for the MarkLogic debugger.

Content Management System (CMS) Integration

This chapter explains how you can use Oxygen XML Editor with Documentum CMS and Microsoft SharePoint. You can use the *plugin support* to develop your own integration with other content management systems..

All the major CMS vendors that are listed in the *CMS Solution Partners section of our website* also provide CMS integration with Oxygen XML Editor.

Related Information:

General Configuration of an Oxygen XML Editor Plugin on page 1336

Working with Databases on page 1068

Integration with Documentum (CMS) (deprecated)

Important: Starting with version 17.0, the support for Documentum (CMS) is deprecated and will no longer be actively maintained.

Oxygen XML Editor provides support for browsing and managing Documentum (CMS) connections in the ***Data Source Explorer*** *view*. You can easily create new resources on the repository, copy and move them using contextual actions or the drag and drop support, or edit and transform the documents in the editor.

Oxygen XML Editor supports Documentum (CMS) version 6.5 and 6.6 with **Documentum Foundation Services 6.5 or 6.6** installed.

Attention: It is recommended to use the latest 1.6.x Java version. It is possible that the Documentum (CMS) support will not work properly if you use other Java versions.

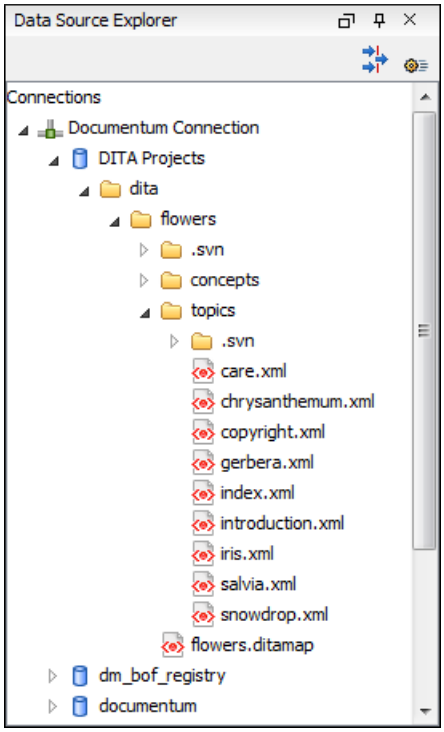


Figure 525: Documentum (CMS) Connection

Configuring a Documentum (CMS) Database Connection

Follow this procedure to configure the support for a Documentum (CMS) database:

1. *Configure Documentum xDB Data Source drivers*.
2. *Configure a Documentum xDB Connection*.
3. To view your connection, go to the ***Data Source Explorer*** *view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu) or switch to the **Database** *perspective*.

How to Configure Documentum (CMS) Data Source Drivers

Available in the Enterprise edition only.

To configure a Documentum (CMS) data source you need the Documentum Foundation Services Software Development Kit (DFS SDK) corresponding to your server version. The DFS SDK can be found in the Documentum (CMS) server installation kit or it can be downloaded from *EMC Community Network*.

Note: The DFS SDK can be found in the form of an archive named for Documentum (CMS) 6.5 (for example, *emc-dfs-sdk-6.5.zip*).

To configure a data source for Documentum (CMS), follow these steps:

1. *Open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Data Sources**.
2. In the **Data Sources** panel click the **New** button.
3. Enter a unique name for the data source.
4. Select **Documentum (CMS)** from the driver type combo box.
5. Press the **Choose DFS SDK Folder** button.

6. Select the folder where you have unpacked the DFS SDK archive file.

   If you have indicated the correct folder the following Java libraries (*JAR* files) will be added to the list (some variation of the library names is possible in future versions of the DFS SDK):

   - `lib/java/emc-bpm-services-remote.jar`
   - `lib/java/emc-ci-services-remote.jar`
   - `lib/java/emc-collaboration-services-remote.jar`
   - `lib/java/emc-dfs-rt-remote.jar`
   - `lib/java/emc-dfs-services-remote.jar`
   - `lib/java/emc-dfs-tools.jar`
   - `lib/java/emc-search-services-remote.jar`
   - `lib/java/ucf/client/ucf-installer.jar`
   - `lib/java/commons/*.jar` (multiple *JAR* files)
   - `lib/java/jaxws/*.jar` (multiple *JAR* files)
   - `lib/java/utils/*.jar` (multiple *JAR* files)

   Note: If for some reason the *JAR* files are not found, you can add them manually by using the **Add Files** and **Add Recursively** buttons and navigating to the `lib/java` folder from the DFS SDK.
7. Click the **OK** button to finish the data source configuration.
8. Continue on to *configure your Documentum connection*.

How to Configure a Documentum (CMS) Connection

Available in the Enterprise edition only.

To configure a connection for a Documentum (CMS) server, follow these steps:

1. *Open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Data Sources**.
2. In the **Connections** panel click the ➕ **New** button.
3. Enter a unique name for the connection.
4. Select one of the previously configured Documentum (CMS) data sources in the **Data Source** combo box.
5. Fill-in the connection details:

   - **URL** - The URL to the Documentum (CMS) server: `http://<hostname>:<port>`
   - **User** - The user name to access the Documentum (CMS) repository.
   - **Password** - The password to access the Documentum (CMS) repository.
   - **Repository** - The name of the repository to log into.
6. Click the **OK** button to finish the configuration of the connection.

Known Issues with Documentum (CMS)

The following are known issues with the Documentum (CMS):

- There is a known problem in the UCF Client implementation for Mac OS X from Documentum 6.5 that prevents you from viewing or editing XML documents from the repository on Mac OS X. The UCF Client is the component responsible for file transfer between the repository and the local machine. This component is deployed automatically from the server. Documentum 6.6 does not exhibit this problem.

  Note: This issue was reproduced with Documentum 6.5 SP1. In Documentum 6.6 this is no longer reproducing.
- For the Documentum driver to work faster on Linux, you need to specify to the JVM to use a weaker random generator, instead of the very slow native implementation. This can be done by modifying in the Oxygen XML Editor startup scripts (or in the `*.vmoptions` file) the system property:

  ```
  -Djava.security.egd=file:/dev/./urandom
  ```

Documentum (CMS) Contextual Menu Actions

While browsing Documentum (CMS) connections in the ***Data Source Explorer** view*, the various nodes include the following contextual menu actions:

Connection Level Nodes

Configure Database Sources

Opens the ***Data Sources*** *preferences page* where you can configure both data sources and connections.

Disconnect

Stops the connection.

New Cabinet

Creates a new cabinet in the repository. The cabinet properties are:

- **Type** - The type of the new cabinet (default is **dm_cabinet**).
- **Name** - The name of the new cabinet.
- **Title** - The title property of the cabinet.
- **Subject** - The subject property of the cabinet.

Refresh

Performs a refresh on the selected node.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Container (Cabinet) Level Nodes

New Folder

Creates a new folder in the current cabinet / folder. The folder properties are the following:

- **Path** - Shows the path where the new folder will be created.
- **Type** - The type of the new folder (default is **dm_folder**).
- **Name** - The name of the new folder.
- **Title** - The title property of the folder.
- **Subject** - The subject property of the folder.

New Document

Creates a new document in the current cabinet / folder. The document properties are the following:

- **Path** - Shows the path where the new document will be created.
- **Name** - The name of the new document.
- **Type** - The type of the new document (default is **dm_document**).
- **Format** - The document content type format.

Import

Imports local files / folders in the selected cabinet / folder of the repository. Actions available when performing an import:

- **Add Files** - Opens a file browse dialog box and allows you to select files to add to the list.
- **Add Folders** - Opens a folder browse dialog box that allows you to select folders to add to the list. The subfolders will be added recursively.
- **Edit** - Opens a dialog box where you can change the properties of the selected file / folder from the list.
- **Remove** - Removes the selected files / folders from the list.

Export

Allows you to export the folder on the remote connection to a local folder.

Rename

Renames the current resource

Delete

Deletes the current container.

Refresh

Performs a refresh on the selected node.

Properties

Shows various properties of the current container.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Resource Level Nodes

Edit

Checks out and opens the selected resource in the editor (if it is not already checked out).

Edit with

Checks out and opens the selected resource in the specified editor or tool (if it is not already checked out).

Open (Read-only)

Opens the selected resource in the editor. The resources are marked as read-only in the editor using a lock icon on the file tab. If you want to edit those resources, select the ***Can edit read only files*** *option* in the **Editor** preferences page.

Open with

Opens the selected resource in the specified editor or tool.

Check Out

Checks out the selected resource from the repository. The action is not available if the resource is already checked out.

Check In

Opens the **Check In** dialog box that allows you to check in the selected resource (commits changes) into the repository and configure some properties for the resource. The action is only available if the resource is checked out.

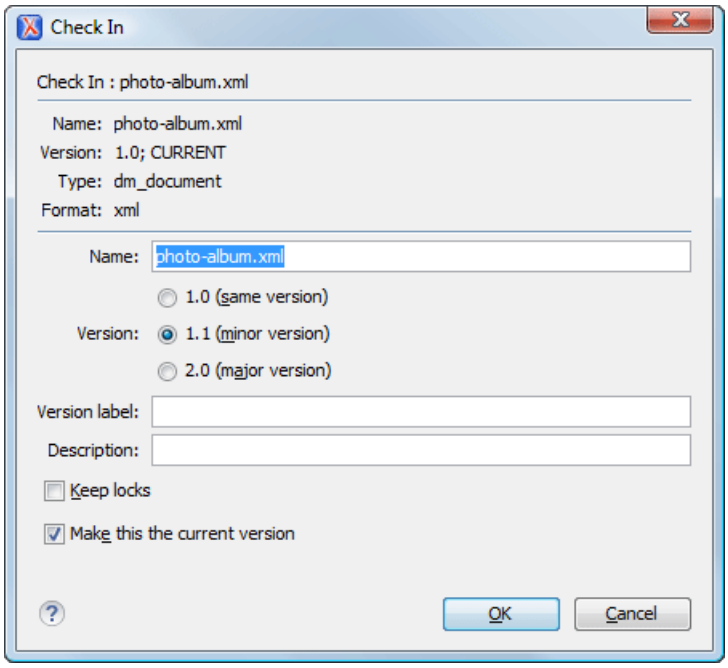


Figure 526: Check In Dialog Box

The following resource properties can be configured in this dialog box:

- **Name** - The resource name in the repository.

- **Version** - Allows you to choose what version the resource will have after being checked in.
- **Version label** - The label of the updated version.
- **Description** - An optional description of the resource.
- **Keep Locks** - When this option is selected, the updated resource is checked in to the repository but it also keeps it locked.
- **Make this the current version** - Makes the updated resource the current version (will have the *CURRENT* version label).

Cancel Checkout

Cancels the checkout process and loses all modifications since the checkout. Action is only available if the resource is checked out.

Export

Allows you to export the folder on the remote connection to a local folder.

Copy

Copies the selected resource to another location in the tree. This action is not available on virtual document descendants. This action can also be performed with drag and drop while holding the **Ctrl (Meta on OS X)** key pressed.

Move

Moves the selected resource to another location in the tree. Action is not available on virtual document descendants and on checked out resources. This action can also be performed with drag and drop.

Copy location

Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

Rename

Renames the current resource

Delete

Deletes the current container.

Add Relationship

Adds a new relationship for the selected resource. This action can also be performed with drag and drop between resources.

Convert to Virtual Document

Allows you to convert a simple document to a virtual document. Action is available only if the resource is a simple document.

Convert to Simple Document

Allows you to convert a virtual document to a simple document. Action is available only if the resource is a virtual document with no descendants.

Refresh

Performs a refresh on the selected node.

Properties

Shows various properties of the current container.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Compare

Compares two selected resources using the ***Compare Files*** *tool*.

Integration with Microsoft SharePoint

Oxygen XML Editor provides support for browsing and managing SharePoint connections in the ***Data Source Explorer*** *view*. You can easily create new resources on the repository, copy and move them using contextual actions or the drag and drop support, or edit and transform the documents in the editor.

Note: You can access documents stored on SharePoint Online for Office 365 sites that use either **Cloud identity (default)** or **Federated identity (ADFS)** as the authentication method.

Restriction: The SharePoint connection is only available in the Enterprise edition of Oxygen XML Editor.

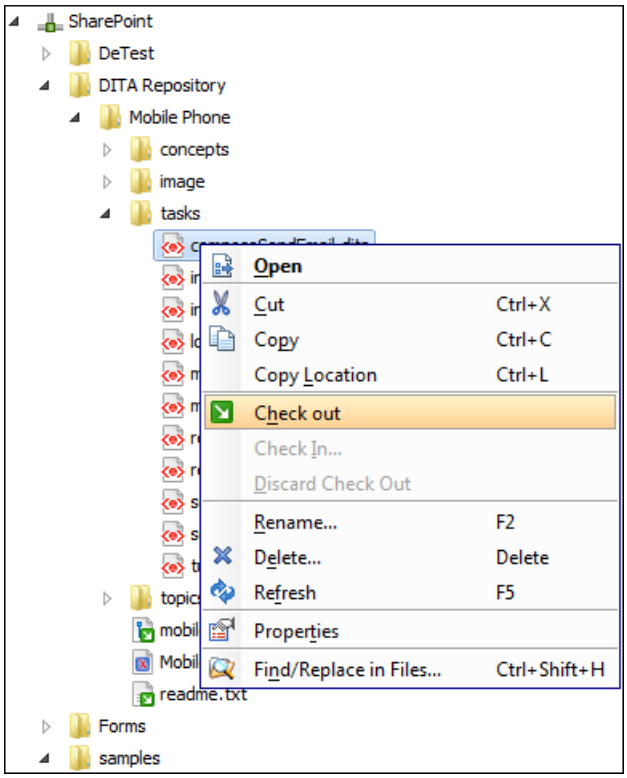


Figure 527: SharePoint Connection

Related Information:

Working with Databases on page 1068

How to Configure a SharePoint Connection

By default, Oxygen XML Editor contains built-in data source drivers for **SharePoint**. Use this data source to create a connection to a SharePoint server that will be available in the ***Data Source Explorer*** *view*.

To configure a SharePoint connection, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Data Sources**.
2. In the **Connections** panel click the **New** button.
3. Enter a unique name for the connection.
4. Select SharePoint in the **Data Source** combo box.
5. Fill-in the connection details:
   a) Set the URL to the SharePoint repository in the field **SharePoint URL**.
   b) Set the server domain in the **Domain** field. If you are using a SharePoint 365 account, leave this field empty.
   c) Set the user name to access the SharePoint repository in the **User** field.
   d) Set the password to access the SharePoint repository in the **Password** field.
6. To view your connection, go to the ***SharePoint Browser*** *view* (if the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu).

To watch our video demonstration about connecting to repository located on a SharePoint server, go to *https://www.oxygenxml.com/demo/SharePoint_Support.html*.

SharePoint Browser View

The **SharePoint Browser** view allows you to connect to a SharePoint repository and perform SharePoint-specific actions on the available resources. To display this view, go to **Window** > **Show View** > **SharePoint Browser**.

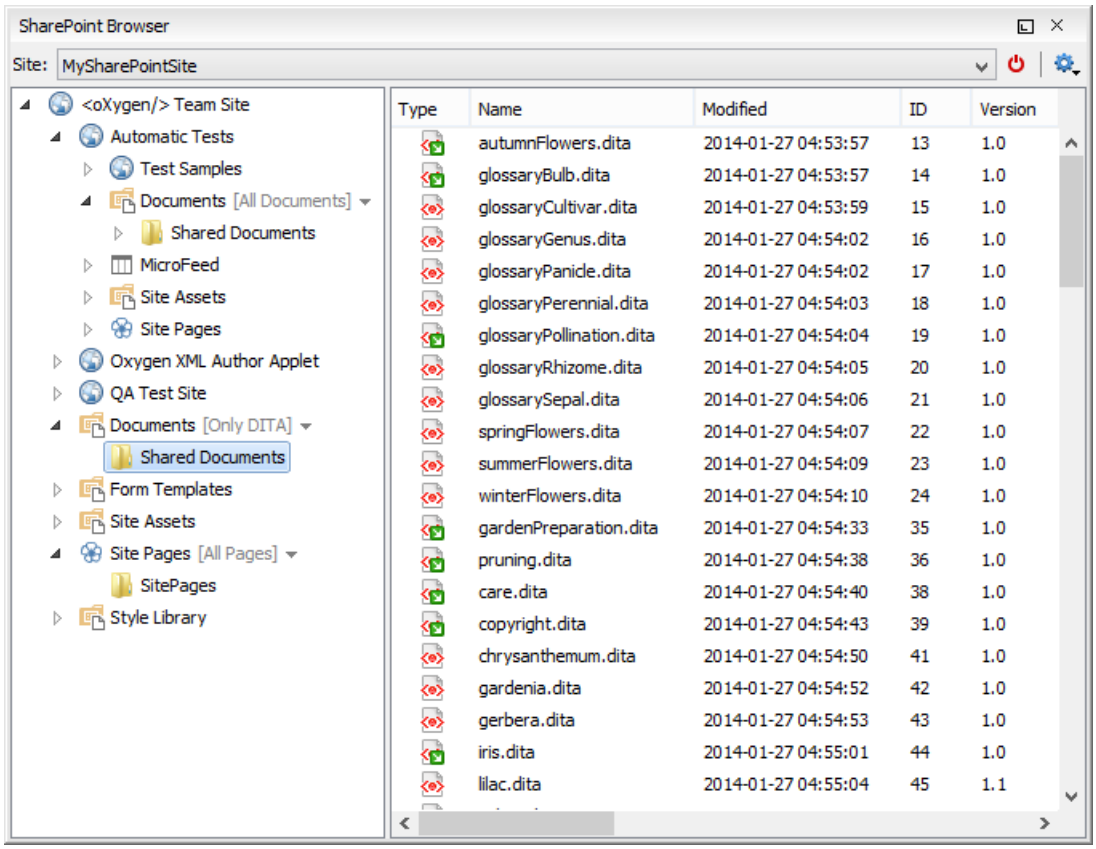


Figure 528: SharePoint Browser View

The view is split in several functional areas:

Connection Area

The following controls are available:

- The **Site** combo box allows you to select and connect to an already *defined SharePoint connection*.
- The **Disconnect** action terminates the current connection.
- The **Settings** drop-down menu contains actions that help you to quickly define a new connection or manage the existing ones from the **Data Source** options page: **New SharePoint Connection** and **Configure Database Sources**. Also, here you can choose one of the predefined view layouts.

SharePoint Site Navigation Area

If there is no connection selected in the **Site** combo box, this area is left blank and promotes the actions that allow you to quickly add SharePoint connections. Otherwise, the navigation area presents the SharePoint site structure in a tree-like fashion with various node types (such as *sites*, *libraries*, and *folders*).

Depending on the type of node, a contextual menu offers customized actions that can be performed on that node. The contextual menu of a folder allows you to create new folders and documents, import folders and files, and to rename and delete the folder.

Note: The rename and delete actions are not available for library root folders (folders located at first level in a SharePoint library).

Each library node displays a drop-down menu next to its name where you can select what you want to display for the current library node. This functionality is also available on the contextual menu of the node.

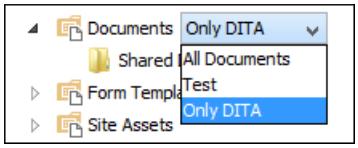


Figure 529: Drop-Down Menu to Select Which Items to Display

Folder Content Area

The content of a folder is displayed in a tabular form, where each row represents the properties of a folder or document. The list of columns and the way the documents and folders are organized depends on the currently selected view of the parent library.

Table 39: Contextual Menu Actions for the Folder Area

Action	Description	Available for	
		folders	documents
Open	Displays the content of the currently selected folder. Opens the current document for editing.	✓	✓
Rename	Renames the current node on server.	✓	✓
Import	Import files or folders into the currently selected folder.	✓	✓
Delete	Deletes the current node from the server.	✓	✓
Copy Location	Copies to clipboard the URL of the current node.	✓	✓
Check Out	Reserves the current document for your use so that other users cannot change it while you are editing it.		✓
Check In	Commits on the server the changes you made to the document, so that other users can see them. It also makes the document available for editing to other users.		✓
Discard Check Out	Discards the previous checkout operation, making the file available for editing to other users.		✓
Refresh	Queries the server to refresh the available properties of the current node.	✓	✓
Drag and Drop	You can drag documents from the **SharePoint Browser** view and drop them in the main editor area to open them with ease.		✓

You can filter and sort the displayed items. To display the available filters of a column, click the filter widget located on the column header. You can apply multiple filters at the same time.

Note: A column can be filtered or sorted only if it was configured this way on the server side.

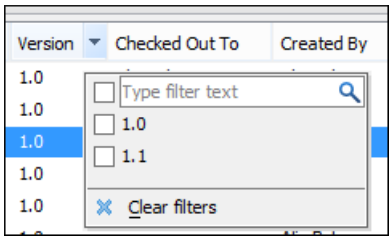


Figure 530: Column Filter

Related Information:

How to Configure a SharePoint Connection on page 1126

SharePoint Contextual Menu Actions

While browsing SharePoint connections in the ***Data Source Explorer*** *view*, the various nodes include the following contextual menu actions:

Connection Level Nodes

Configure Database Sources

Opens the ***Data Sources*** *preferences page* where you can configure both data sources and connections.

Disconnect

Stops the connection.

New Folder

Creates a new folder on the connection.

Import Files

Allows you to add a new file on the connection, in the current folder.

Refresh

Performs a refresh on the selected node.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Folder Level Nodes

New File

Creates a new file on the connection, in the current folder.

New Folder

Creates a new folder on the connection.

Import Folders

Imports folders on the server.

Import Files

Allows you to add a new file on the connection, in the current folder.

Export

Allows you to export the folder on the remote connection to a local folder.

Cut

Removes the current selection and places it in the clipboard.

Copy

Copies the current selection into the clipboard.

Paste

Pastes the copied selection.

Rename

Renames the current resource

Delete

Deletes the current container.

Refresh

Performs a refresh on the selected node.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Resource Level Nodes

Open

Opens the selected resource in the editor.

Open in System Application

When you use this action, Oxygen XML Editor downloads the selected resource to a local temporary folder and opens the selected resource in the system application that is currently set as the default application associated with that type of resource. You can then edit the resource, save it, and when you switch the focus back to the **Data Source Explorer** view, Oxygen XML Editor will detect that there was a change and will ask if you want to upload the edited resource to the server.

Cut

Removes the current selection and places it in the clipboard.

Copy

Copies the current selection into the clipboard.

Copy location

Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

Check Out

Checks out the selected document on the server.

Check In

Checks in the selected document on the server. This action opens the **Check In** dialog box. In this dialog box, the following options are available:

- **Minor Version** - Increments the minor version of the file on the server.
- **Major Version** - Increments the major version of the file on the server.
- **Overwrite** - Overwrites the latest version of the file on the server.
- **Comment** - Allows you to comment on a file that you check in.

Discard Check Out

Discards the previous checkout operation, making the file available for editing to other users.

Rename

Renames the current resource

Delete

Deletes the current container.

Refresh

Performs a refresh on the selected node.

Properties

Shows various properties of the current container.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace text in multiple files from the connection.

Compare

Compares two selected resources using the ***Compare Files*** *tool*.

Importing Data

13

Topics:

- *Import from Text Files*
- *Import from MS Excel Files*
- *Import Database Data as an XML Document*
- *Import from HTML Files*
- *Import Content Dynamically*

Computer systems and databases contain data in incompatible formats and exchanging data between these systems can be very time consuming. Converting the data to XML can greatly reduce the complexity and create data that can be read by various types of applications.

Oxygen XML Editor offers support for importing text files, MS Excel files, Database Data, and HTML files into XML documents. The XML documents can be further converted into other formats using the *Transform features*.

Import from Text Files

To import a text file into an XML file, follow these steps:

1. Go to **File** > **Import** > **Text File**.
   A **Select text file** dialog box is displayed.
2. Select the URL of the text file.
3. Select the encoding of the text file.
4. Click the **Next** button.
   The **Import Criteria** dialog box is displayed.

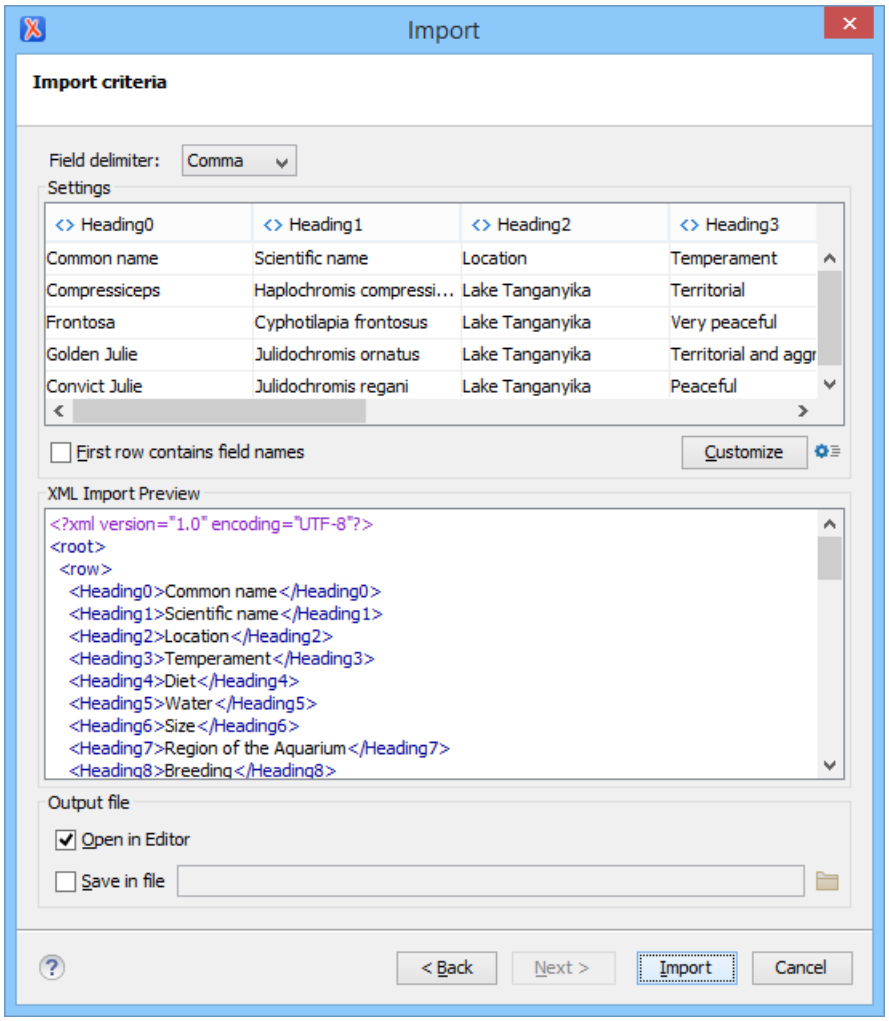


Figure 531: Import Criteria Dialog Box

5. Configure the settings for the conversion.
   a) Select the **Field delimiter** for the import settings. You can choose between the following: `Comma`, `Semicolon`, `Tab`, `Space`, or `Pipe`.
   b) The **Import settings** section presents the input data in a tabular form. By default, all data items are converted to element content (<> symbol), but this can be overridden by clicking the individual column headers. Clicking a column header once causes the data from this column to be converted to attribute values (= symbol). Clicking a second time causes the column data to be ignored (× symbol) when generating the XML file. You can cycle through these three options by continuing to click the column header.
   c) **First row contains field names** - If this option is selected, the default column headers are replaced (where such information is available) by the content of the first row. In other words, the first row is interpreted as containing the field names. The changes are also visible in the preview panel.
   d) **Customize** - This button opens a **Presentation Names** dialog box that allows you to edit the name, XML name, and conversion criterion for the root and row elements. The XML names can be edited by double-clicking the desired item and entering the label. The conversion criteria can also be modified by selecting one of the following option in the drop-down menu: `ELEMENT`, `ATTRIBUTE`, or `SKIPPED`.
   e) **Import Settings** - Clicking this button opens the ***Import*** *preferences page* that allows you to configure more import options.
   f) The **XML Import Preview** panel contains an example of what the generated XML document looks like.
   g) **Open in editor** - If selected, the new XML document created from the imported text file is opened in the editor.
   h) **Save in file** - If selected, the new XML document is saved in the specified path.

6. Click **Import** to generate the XML document.

Import from MS Excel Files

Oxygen XML Editor provides two methods for importing MS Excel files into an XML file. The first method is to simply copy data from Excel and paste it into a document in **Author** mode, but this is only supported in DITA, DocBook, TEI, JATS, and XHTML documents. Oxygen XML Editor also offers a configurable import wizard that works with any type of XML document.

Smart Paste Method

If you are importing data into DITA, DocBook, TEI, JATS, or XHTML documents, you can open the Excel spreadsheet in your office application, copy its content, and paste it into your document in **Author** mode.

The Oxygen XML Editor **Smart Paste** mechanism will convert the pasted content to the equivalent XML markup and considers various pasting solutions to keep the resulting document valid, while preserving the original text styling (such as bold, italics, underline) and formatting (such as lists, tables, paragraphs).

Import Wizard Method

By default, this method supports importing Excel 97/2000/XP/2003 formats out-of-the-box. To import spreadsheet data from Excel 2007 or newer, additional libraries are needed before using this procedure. See *Import Data from MS Excel 2007 or Newer* on page 1136 for instructions on adding more libraries.

To use the **Import** wizard to import an Excel file into an XML file, follow these steps:

1. Go to **File** > **Import** > **MS Excel file**.
2. Select the URL of the Excel file.

   The sheets of the document you are importing are presented in the **Available Sheets** section of this dialog box.
3. Click the **Next** button to proceed to the next stage of the wizard.

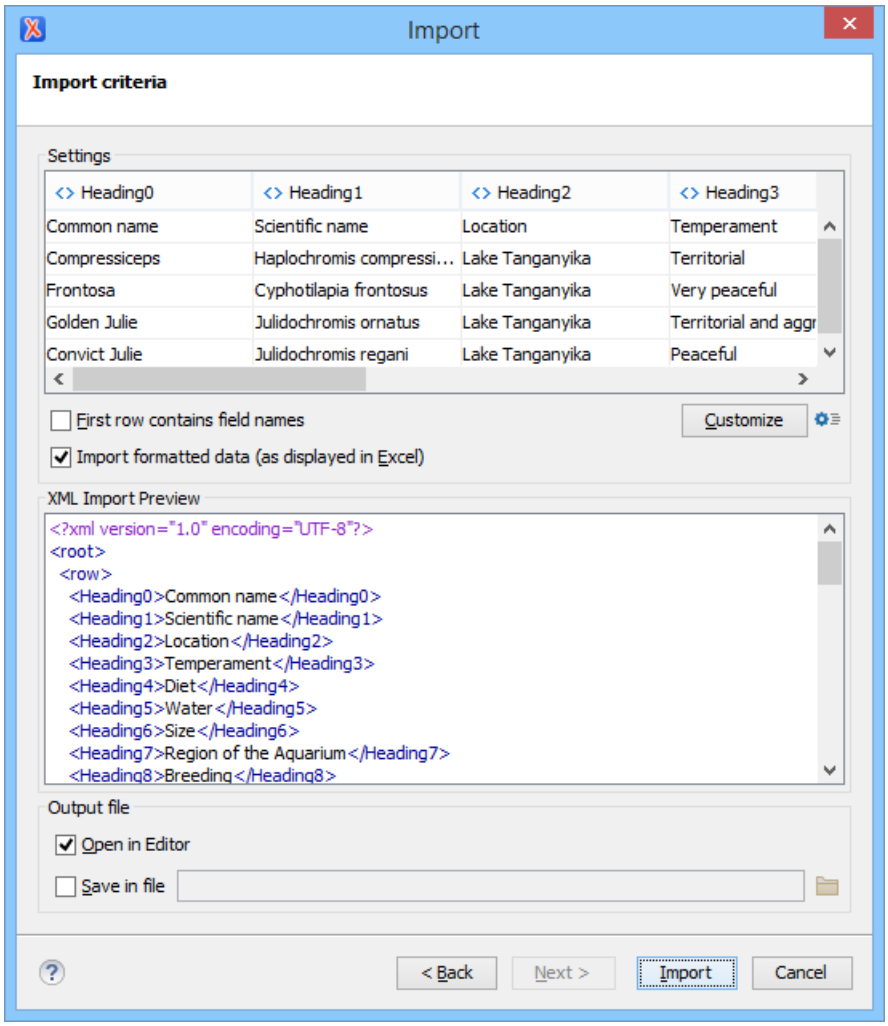


Figure 532: Import Wizard

4. Configure the settings for the conversion. This stage of the wizard offers the following options:

Import settings section

Presents the input data in a tabular form. By default, all data items are converted to element content (<> symbol), but this can be overridden by clicking the individual column headers. Clicking a column header once causes the data from this column to be converted to attribute values (= symbol). Clicking a second time causes the column data to be ignored (× symbol) when generating the XML file. You can cycle through these three options by continuing to click the column header.

First row contains field names

If this option is selected, the default column headers are replaced (where such information is available) by the content of the first row. In other words, the first row is interpreted as containing the field names. The changes are also visible in the preview panel.

Customize

This button opens a **Presentation Names** dialog box that allows you to edit the name, XML name, and conversion criterion for the root and row elements. The XML names can be edited by double-clicking the desired item and entering the label. The conversion criteria can also be modified by selecting one of the following option in the drop-down menu: `ELEMENT`, `ATTRIBUTE`, or `SKIPPED`.

Import Settings

Clicking this button opens the ***Import*** *preferences page* that allows you to configure more import options.

Import formatted data (as displayed in Excel)

If this option is selected, the imported data retains the Excel data formatting (such as the representation of numeric values or dates). If deselected, the data formatting is not imported.

XML Import Preview panel

Contains an example of what the generated XML document will look like.

Open in editor

If selected, the new XML document created from the imported file is opened in the editor.

Save in file

If selected, the new XML document is saved in the specified path.

5. Click **Import** to generate the XML document.

Import Data from MS Excel 2007 or Newer

To import spreadsheet data from Excel 2007 or newer (`.xlsx`), Oxygen XML Editor needs additional libraries from the release 3.10 of the Apache POI project.

To add the libraries, follow these steps:

1. Download version 3.10 of the Apache POI project from *http://archive.apache.org/dist/poi/release/bin/*. The specific ZIP file that you need is: `poi-bin-3.10-FINAL-20140208.zip`.
2. Unpack `poi-bin-3.10-FINAL-20140208.zip`.
3. Copy the following `.jar` files in the `lib` directory of the installation folder of Oxygen XML Editor :
   - `dom4j-1.6.1.jar`
   - `poi-ooxml-3.10-FINAL-20140208.jar`
   - `poi-ooxml-schemas-3.10-FINAL-20140208.jar`
   - `xmlbeans-2.3.0.jar`

Result: You can now use the ***Import** wizard* to import data from Excel 2007 or newer.

Import Database Data as an XML Document

To import the data from a relational database table as an XML document, follow these steps:

1. Go to **File** > **Import** > **Database Data** to start the **Import** wizard.

   This opens a **Select database table** dialog box that lists all the defined database connections:

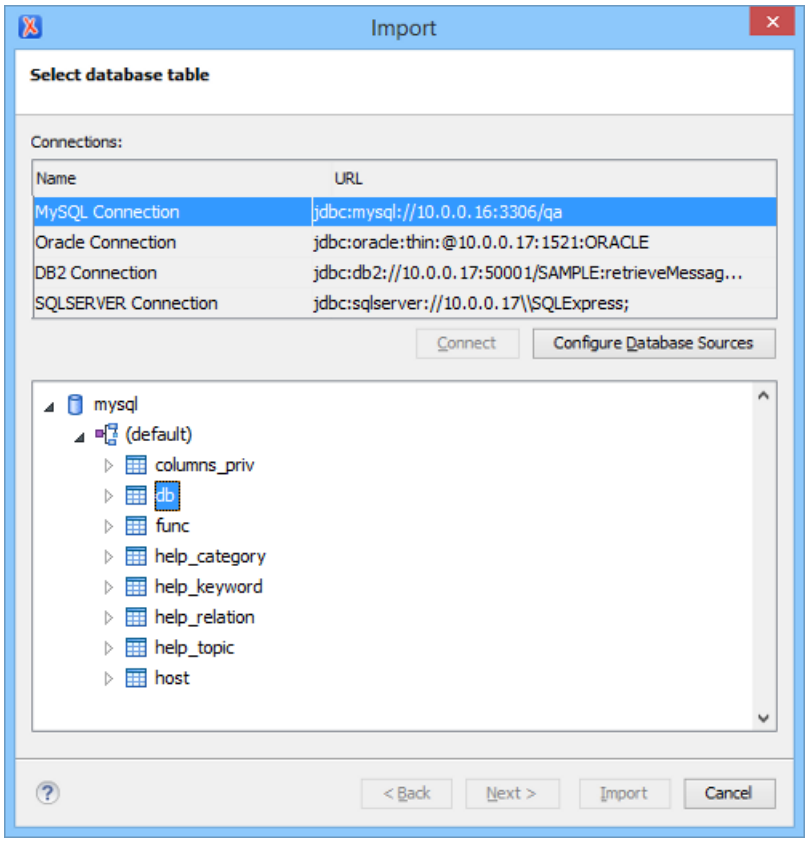


Figure 533: Select Database Table Dialog Box

2. Select the connection to the database that contains the appropriate data.

   Only connections configured in relational data sources can be used to import data.
3. If you want to edit, delete, or add a data source or connection, click the **Configure Database Sources** button. The **Preferences**/**Data Sources** option page is opened.
4. Click **Connect**.
5. In the list of sources, expand a schema and choose the required table.
6. Click the **Next** button.

   The **Import Criteria** dialog box is opened with a default query string in the **SQL Query** pane.

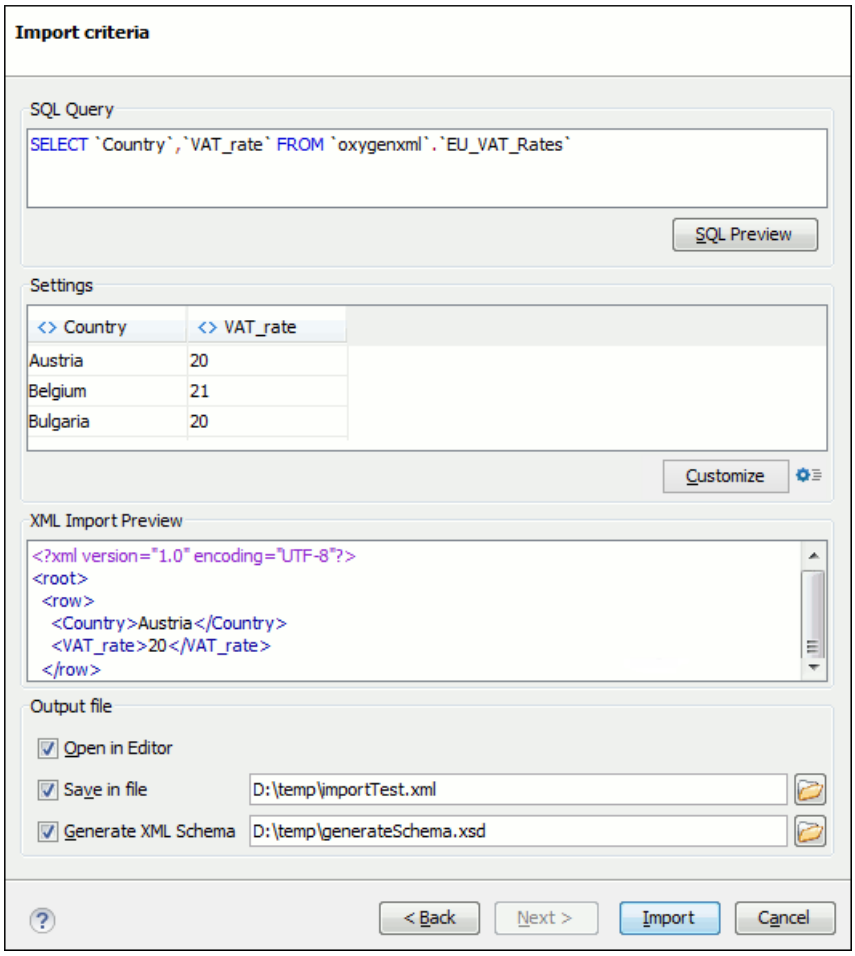


Figure 534: Import from Database Criteria Dialog Box

7. Configure the settings for the conversion.
   a) **SQL Preview** - If this button is pressed, the **Settings** pane displays the labels that are used in the XML document and the first five lines from the database. By default, all data items are converted to element content (<> symbol), but this can be overridden by clicking the individual column headers. Clicking a column header once causes the data from this column to be converted to attribute values (= symbol). Clicking a second time causes the column data to be ignored (× symbol) when generating the XML file. You can cycle through these three options by continuing to click the column header.
   b) **Customize** - This button opens a **Presentation Names** dialog box that allows you to edit the name, XML name, and conversion criterion for the root and row elements. The XML names can be edited by double-clicking the desired item and entering the label. The conversion criteria can also be modified by selecting one of the following option in the drop-down menu: `ELEMENT`, `ATTRIBUTE`, or `SKIPPED`.
   c) **Import Settings** - Clicking this button opens the ***Import*** *preferences page* that allows you to configure more import options.
   d) The **XML Import Preview** panel contains an example of what the generated XML document looks like.
   e) **Open in editor** - If selected, the new XML document created from the imported file is opened in the editor.
   f) **Save in file** - If selected, the new XML document is saved in the specified path.
   g) **Generate XML Schema** - Allows you to specify the path of the generated XML Schema file.
8. Click **Import** to generate the XML document.

Import from HTML Files

Oxygen XML Editor offers two methods for importing HTML files into an XML document. The first method is to simply copy data from an HTML document and paste it into a document in **Author** mode, but this is only

supported in DITA, DocBook, TEI, JATS, and XHTML documents. Oxygen XML Editor also offers a configurable import wizard that works with any type of XML document.

Smart Paste Method

If you are importing data into DITA, DocBook, TEI, JATS, or XHTML documents, you can open the HTML document in your web browser, copy its content, and paste it into your document in **Author** mode.

The Oxygen XML Editor **Smart Paste** mechanism will convert the pasted content to the equivalent XML markup and considers various pasting solutions to keep the resulting document valid, while preserving the original text styling (such as bold, italics, underline) and formatting (such as lists, tables, paragraphs).

Import Wizard Method

To use the **Import** wizard to import from HTML files, follow these steps:

1. Go to **File** > **Import** > **HTML File**. The **Import HTML** wizard is displayed.
2. Enter the URL of the HTML document.
3. Select the type of the resulting XHTML document:
   - XHTML5
   - XHTML 1.0 Transitional
   - XHTML 1.0 Strict
4. Click the **OK** button.

Result: The resulting document is an XHTML file containing a DOCTYPE declaration that references the XHTML DTD definition on the Web. The parsed content of the imported file is transformed to XHTML5, XHTML Transitional, or XHTML Strict depending on the option you chose.

Import Content Dynamically

Along with the built-in support for various useful URL protocols (such as HTTP or FTP), Oxygen XML Editor also provides special support for a *convert* protocol that can be used to chain predefined processors to dynamically import content from various sources.

A *dynamic conversion URL* chains various processors that can be applied, in sequence, on a target resource and has the following general syntax:

```
convert:/processor=xslt;ss=urn:processors:excel2d.xsl/processor=excel!/
urn:files:sample.xls
```

The previous example first applies a processor (`excel`) on a target identified by the identifier (`urn:files:sample.xls`) and converts the Excel™ resource to XML. The second applied processor (`xslt`) applies an XSLT stylesheet identified using the identifier (`urn:processors:excel2d.xsl`) over the resulting content from the first applied processor. These identifiers are all mapped to real resources on disk via an *XML catalog* that is configured in the application, as in the following example:

```
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
  <rewriteURI uriStartString="urn:files:" rewritePrefix="./resources/"/>
  <rewriteURI uriStartString="urn:processors:" rewritePrefix="./processors/"/>
</catalog>
```

The target resource part of the conversion URL must always follow the `!/` pattern. It can be any of the following:

- An absolute URL that points to a resource.
- An identifier that will be resolved to an actual resource via the *XML Catalog* support in the application. In the example above, the **urn:files:sample.xls** target resource is resolved via the *XML catalog*.
- A relative location. This location can only be resolved to an actual resource URL when the application has enough information about the location where the URL is referenced.

  For example, for a *DITA map* with a `topicref` such as:

```
<topicref href="convert:/.../processor=excel!/resources/sample.xls"/>
```

the `resources/sample.xls` path will be resolved relative to the *DITA map* location.

This type of URL can be opened in the application by using the **Open URL** action from the **File** menu. It can also be referenced from existing XML resources via `xi:include` or as a topic reference from a *DITA map*.

A *GitHub* project that contains various dynamic conversion samples for producing DITA content from various sources (and then publishing it) can be found here: *https://github.com/oxygenxml/dita-glass*.

Conversion Processors

A set of predefined conversion processors is provided in Oxygen XML Editor. Each processor has its own parameters that can be set to control the behavior of the conversion process. All parameters that are resolved to resources are passed through the *XML catalog* mapping.

The following predefined conversion processors are included:

- **`xslt` Processor** - Converts an XML input using the Saxon EE XSLT processor. The `ss` parameter indicates the stylesheet resource to be loaded. All other specified parameters will be set as parameters to the XSLT transformation.

  ```
  convert:/processor=xslt;ss=urn:processors:convert.xsl;p1=v1!/urn:files:sample.xml
  ```
- **`xquery` Processor** - Converts an XML input using the Saxon EE XQuery processor. The `ss` parameter indicates the XQuery script to be loaded. All other specified parameters will be set as parameters to the XSLT transformation.

  ```
  convert:/processor=xquery;ss=urn:processors:convert.xquery;p1=v1!/
  urn:files:sample.xml
  ```
- **`excel` Processor** - Converts an Excel™ input to an XML format that can later be converted by other piped processors. It has a single parameter `sn`, which indicates the name of the sheet that needs to be converted. If this parameter is missing, the XML will contain the combined content of all sheets included in the Excel™ document.

  ```
  convert:/processor=excel;sn=test!/urn:files:sample.xls
  ```
- **`java` Processor** - Converts an input to another format by applying a specific Java method. The `jars` parameter is a comma-separated list of *JAR* libraries, or folders that libraries will be loaded from. The `ccn` parameter is the fully qualified name of the conversion class that will be instantiated. The conversion class needs to have a method with the following signature:

  ```
   public void convert(String systemID, String originalSourceSystemID,
  InputStream is, OutputStream os, LinkedHashMap<String, String> properties)
  throws IOException
  ```

  ```
  convert:/processor=java;jars=libs;ccn=test.JavaToXML!/
  urn:files:java/WSEditorBase.java
  ```
- **`js` Processor** - Converts an input to another format by applying a JavaScript method. The `js` parameter indicates the script that will be used. The `fn` parameter is the name of the method that will be called from the script. The method must take a string as an argument and return a string. If any of the parameters are missing, an error is thrown and the conversion stops.

  ```
  convert:/processor=js;js=urn:processors:md.js;fn=convertExternal!/
  urn:files:sample.md
  ```
- **`json` Processor** - Converts a JSON input to XML. It has no parameters.

  ```
  convert:/processor=json!/urn:files:personal.json
  ```
- **`xhtml` Processor** - Converts HTML content to well-formed XHTML. It has no parameters.

  ```
  convert:/processor=xhtml!/urn:files:test.html
  ```
- **`wrap` Processor** - Wraps content in a tag name making it well-formed XML. The `rn` parameter indicates the name of the root tag to use. By default, it is `wrapper`. The `encoding` parameter specifies the encoding that

should be used to read the content. By default, it is UTF8. As an example, this processor can be used if you want to process a comma-separated values file with an XSLT stylesheet to produce XML content. The CSV file is first wrapped as well-formed XML, which is then processed with an `xslt` processor.

```
convert:/processor=wrap!/urn:files:test.csv
```

- **cache Processor** - Caches the converted content obtained from the original document to a temporary file. The cache will be used on subsequent uses of the same URL, thus increasing the speed for the application returning the converted content. If the original URL points to the local disk, the cache will be automatically invalidated when the original file content gets modified. Otherwise, if the original URL points to a remote resource, the cache will need to be invalidated by reloading (**File** > **Reload (F5)**) the URL content that is opened in the editor.

  ```
  convert:/processor=cache/processor=xslt;…..!/urn:files:test.csv
  ```

Reverse Conversion Processors

All processors defined above can also be used for saving content back to the target resource if they are defined in the URL as reverse processors. Reverse processors are evaluated right to left. These reverse processors allow *round-tripping* content to and from the target resource.

As an example, the following URL converts HTML to DITA when the URL is opened using the `h2d.xsl` stylesheet and converts DITA to HTML when the content is saved in the application using the `d2h.xsl` stylesheet.

```
convert:/processor=xslt;ss=h2d.xsl/rprocessor=xslt;ss=d2h.xsl!/
urn:files:sample.html
```

Important: If you are publishing a *DITA map* that has such conversion URL references inside, you need to edit the transformation scenario and set the value of the parameter *fix.external.refs.com.oxygenxml* to *true*. This will instruct Oxygen XML Editor to resolve such references during a special pre-processing stage. Depending on the conversion, you may also require additional libraries to be added using the **Libaries** button in the **Advanced** tab of the transformation scenario.

Related Information:

Debugging XSLT Stylesheets and XQuery Documents 14

Topics:

- *Debugger Layout*
- *Working with the XSLT / XQuery Debugger*
- *Debugging Java Extensions*
- *Supported Processors for XSLT / XQuery Debugging*
- *Performance Profiling of XSLT Stylesheets and XQuery Documents*

Oxygen XML Editor includes a powerful debugging interface that helps you to detect and solve problems with XSLT and XQuery transformations.

XSLT Debugger Perspective

The **XSLT Debugger** *perspective* allows you to detect problems in an XSLT transformation by executing the process step by step. To switch the focus to this *perspective*, select the **XSLT Debugger** button in the top-right corner of the interface or **Window** > **Open perspective** > **XSLT Debugger**.

XQuery Debugger Perspective

The **XQuery Debugger** *perspective* allows you to detect problems in an XQuery transformation process by executing the process step by step in a controlled environment and inspecting the information provided in the special views. To switch the focus to this *perspective*, select the **XQuery Debugger** button in the top-right corner of the interface or **Window** > **Open perspective** > **XQuery Debugger**.

XSLT/XQuery Debugging Overview

The **XSLT Debugger** and **XQuery Debugger** *perspectives* allows you to test and debug XSLT 1.0 / 2.0 / 3.0 stylesheets and XQuery 1.0 / 3.0 documents including complex XPath 2.0 / 3.0 expressions. The interface presents simultaneous views of the source XML document, the XSLT/XQuery document and the result document. As you go step by step through the XSLT/XQuery document the corresponding output is generated step by step, and the corresponding position in the XML file is highlighted. At the same time, special views provide various types of debugging information and events useful to understand the transformation process.

The following set of features allow you to test and solve XSLT/XQuery problems:

- Support for XSLT 1.0 stylesheets (using Saxon 6.5.5 and Xalan XSLT engines), XSLT 2.0 / 3.0 stylesheets and XPath 2.0 / 3.0 expressions that are included in the stylesheets (using Saxon 9.7.0.15 XSLT engine) and XQuery 1.0 / 3.0 (using Saxon 9.7.0.15 XQuery engine).
- Stepping capabilities: step in, step over, step out, run, run to cursor, run to end, pause, stop.
- Output to source mapping between every line of output and the instruction element / source context that generated it.
- *Breakpoints* on both source and XSLT/XQuery documents.
- Call stack on both source and XSLT/XQuery documents.
- Trace history on both source and XSLT/XQuery documents.
- Support for XPath expression evaluation during debugging.
- Step into imported/included stylesheets as well as included source entities.
- Available templates and hits count.
- Variables view.
- Dynamic output generation.

Debugger Layout

The XML and XSL files are displayed in ***Text*** *mode*. The other modes (***Author*** *mode*, ***Grid*** *mode*) are available only in *the* ***Editor*** *perspective*.

The **XSLT/XQuery Debugger** *perspective* contains the following components:

- **Source Document View (XML)** - Displays and allows the editing of XML files (documents).
- **XSLT/XQuery Document View (XSLT/XQuery)** - Displays and allows the editing of XSL files (stylesheets) or XQuery documents.
- **Output View** - Displays the output that results from inputting a document (XML) and a stylesheet (XSL) or XQuery document in the transformer. The transformation result is written dynamically while the transformation is processed. Several actions are available in the contextual menu for this view, including **Find/Replace**, **Copy**, and **Format and Indent**. There are two types of output views: a **Text** view (with XML syntax highlights) and **XHTML** view. For large outputs, the XHTML view can be disabled (see *Debugger Settings*).
- ***Control Toolbar*** - Contains a variety of actions to help you configure and control the debugging process.
- ***Information Views*** - The information views at the bottom of the editor display various types of information to help you understand the transformation process.

Tip: The **Output** view and the various other information views are *dockable* so that you can configure the workspace according to your preferences.

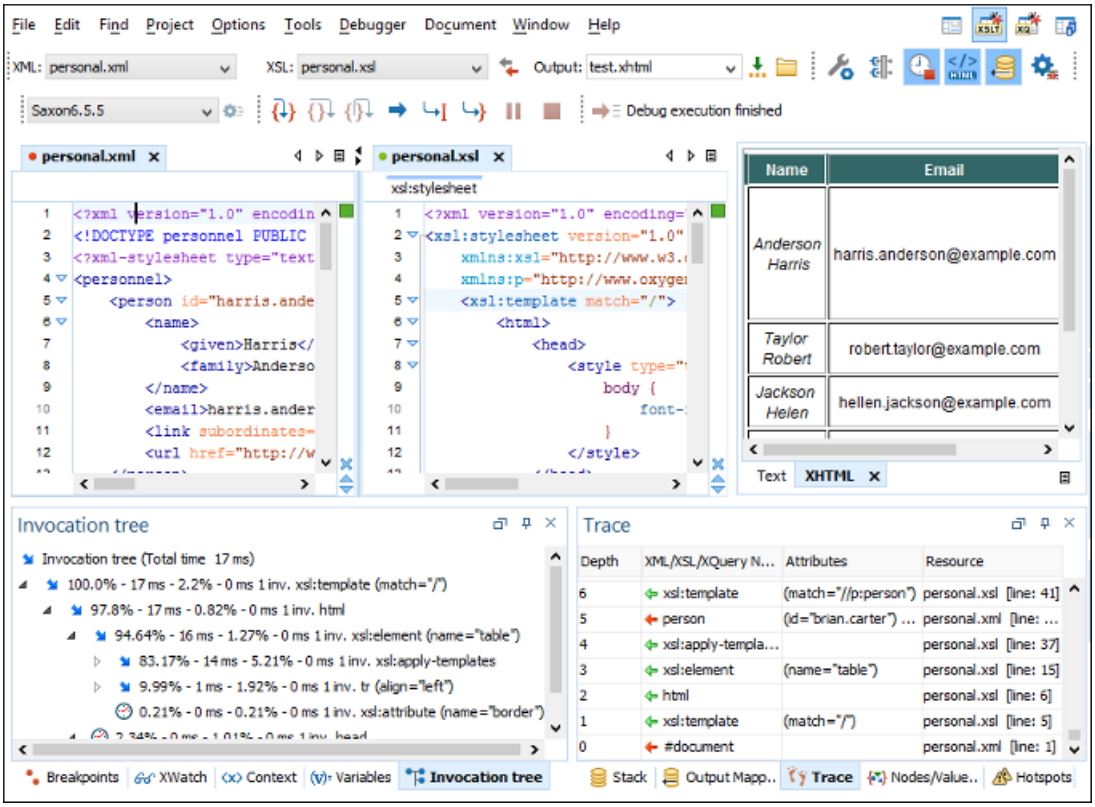


Figure 535: Debugger Interface

XML documents and XSL stylesheets or XQuery documents that were opened in the **Editor** *perspective* are automatically sorted into the first two panes. When multiple files of each type are opened, the individual documents and stylesheets are separated using the familiar tab management system of the **Editor** *perspective*. Selecting a tab brings the document or stylesheet into focus and enables editing without the need to go back to the **Editor** *perspective*.

In Debugger mode, the normal editor toolbar is not available. However, the functions are still accessible from the **Document** menu and the contextual menus.

Bookmarks are replaced in the **Debugger** *perspective* by *breakpoints*.

During debugging, the current execution node is highlighted in both document (XML) and XSLT/XQuery views.

Control Toolbar

The **Control** toolbar contains all the actions that you need to configure and control the debugging process. The following actions are described as they appear in the toolbar from left to right.

Figure 536: Control Toolbar

XML source selector

The current selection represents the source document used as input by the transformation engine. The selection list contains all opened files (XML files being emphasized). This option allows you to use other file types also as source documents. In an XQuery debugging session this selection field can be set to the default value `NONE`, because usually XQuery documents do not require an input source.

XSL / XQuery selector

The current selection represents the stylesheet or XQuery document to be used by the transformation engine. The selection list contains all opened files (XSLT / XQuery files being emphasized).

Link with editor

When selected, the XML and XSLT/XQuery selectors display the names of the files opened in the central editor panels. This button is toggled off by default.

Output selector

The selection represents the output file specified in the associated transformation scenario. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Configure parameters

Opens a dialog box that allows you to configure the XSLT / XQuery parameters to be used by the transformation.

Libraries

Allows you to add and remove the Java classes and *JARS* used as XSLT extensions.

/Turn on/off profiling

Enables / Disables current transformation profiling.

Enable XHTML output

Enables the rendering of the output in the ***XHTML*** *output view* during the transformation process. For performance issues, disable XHTML output when working with very large files. Note that only XHTML conformant documents can be rendered by this view. To view the output result of other formats, such as HTML, save the **Text output** area to a file and use an external browser for viewing.

When starting a debug session from the **Editor** *perspective* by using the **Debug Scenario** action, the state of this toolbar button reflects the state of the **Show as XHTML** output option from the scenario.

Turn on/off output to source mapping

Enables or disables the output to source mapping between every line of output and the instruction element / source context that generated it.

Debugger preferences

Quick link to *Debugger preferences page*.

XSLT / XQuery engine selector

Lists the *processors available for debugging XSLT and XQuery transformations*.

XSLT / XQuery engine advanced options

Advanced options available for Saxon 9.7.0.15.

Step into

Starts the debugging process and runs until the next instruction is encountered.

Step over

Run until the current instruction and its sub-instructions are over. Usually this will advance to the next sibling instruction.

```
12  <xsl:template match="CCC" priority="4">
13    <h3 style="color:blue">
14      <xsl:value-of select="name()"/>
15      <xsl:text> (id=</xsl:text>
16      <xsl:value-of select="@id"/>
17      <xsl:text>)</xsl:text>
18    </h3>
19    <xsl:message>Step over goes here</xsl:message>
20  </xsl:template>
```

Figure 537: Step over

Step out

Run until the parent of the current instruction is over. Usually this will advance to the next sibling of the parent instruction.

```
12  <xsl:template match="CCC" priority="4">
13    <h3 style="color:blue">
14      <xsl:value-of select="name()"/>
15      <xsl:text> (id=</xsl:text>
16      <xsl:value-of select="@id"/>
17      <xsl:text>)</xsl:text>
18    </h3>
19    <xsl:message>Step out goes here</xsl:message>
20  </xsl:template>
```

Figure 538: Step out

Run

Starts the debugging process. The execution of the process is paused when a *breakpoint* is encountered or the transformation ends.

Run to cursor

Starts the debugging process and runs until one of the following conditions occur: the line of cursor is reached, a valid *breakpoint* is reached or the execution ends.

Run to end

Runs the transformation until the end, without taking into account enabled *breakpoints*, if any.

Pause

Request to pause the current transformation as soon as possible.

Stop

Request to stop the current transformation without completing its execution.

Show current execution nodes

Reveals the current debugger context showing both the current instruction and the current node in the XML source. Possible displayed states:

- Entering ( ) or leaving ( ) an XML execution node.

- Entering ( ) or leaving ( ) an XSL execution node.
- Entering ( ) or leaving ( ) an XPath execution node.

  Note: When you set a MarkLogic server as a processor, the **Show current execution nodes** button is named **Refresh current session context from server**. Click this button to refresh the information in all the views.

Note: For some of the XSLT processors (Saxon-HE/PE/EE) the debugger could be configured to step into the XPath expressions affecting the behavior of the following debugger actions: **Step into**, **Step over** or **Step Out**.

Related Information:

Advanced Saxon HE/PE/EE XQuery Transformation Options on page 895

Debugging Information Views

The information views at the bottom of the editor is comprised of two panes that are used to display various types of information used to understand the transformation process. For each information type there is a corresponding tab. While running a transformation, relevant events are displayed in the various information views. This enables the developer to obtain a clear view of the transformation progress. By using the debug controls, developers can easily isolate parts of stylesheet. Therefore, they may be more easily understood and modified.

The information types include the following:

Left side information views

- ***Context*** *node view*
- ***XWatch*** *view*
- ***Breakpoints*** *view*
- ***Messages*** *view* (XSLT only)
- ***Variables*** *view*
- ***Invocation Tree*** *view*

Right side information views

- ***Stack*** *view*
- ***Output Mapping Stack*** *view*
- ***Trace*** *view*
- ***Templates*** *view* (XSLT only)
- ***Nodes/Values Set*** *view*
- ***Hotspots*** *view*

Tip: The information views are *dockable* so that you can configure the workspace according to your preferences.

Context Node View

The context node is valid only for XSLT debugging sessions and is a source node corresponding to the XSL expression that is evaluated. It is also called the context of execution. The context node implicitly changes as the processor hits various steps (at the point where XPath expressions are evaluated). This node has the same value as evaluating '.' (dot) XPath expression in ***XWatch*** *view*. The value of the context node is presented as a tree in the **Context Node** view. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

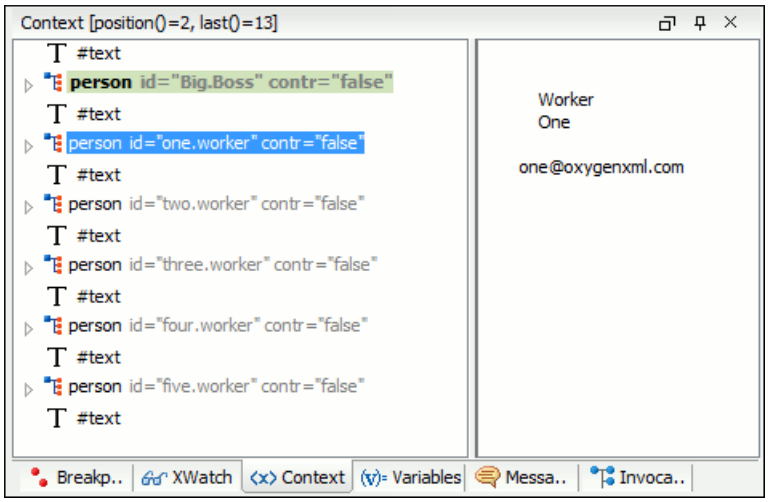


Figure 539: Context node view

The context node is presented in a tree-like fashion. Nodes from a defined namespace bound to a prefix are displayed using the qualified name. If the namespace is not bound to a prefix, the namespace URI is presented before the node name. The value of the selected attribute or node is displayed in the right side panel. The **Context** view also presents the current mode of the XSLT processor if this mode differs from the default one.

The title bar displays the current element index and the number of elements that compose the current context (this information is not available if you choose Xalan or Saxon 6 as processing engine).

XPath Watch (XWatch) View

The **XWatch** view shows XPath expressions evaluated during the debugging process. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Expressions are evaluated dynamically as the processor changes its source context. When you type an XPath expression in the **Expression** column, Oxygen XML Editor supports you with syntax highlight and *content completion assistance*.

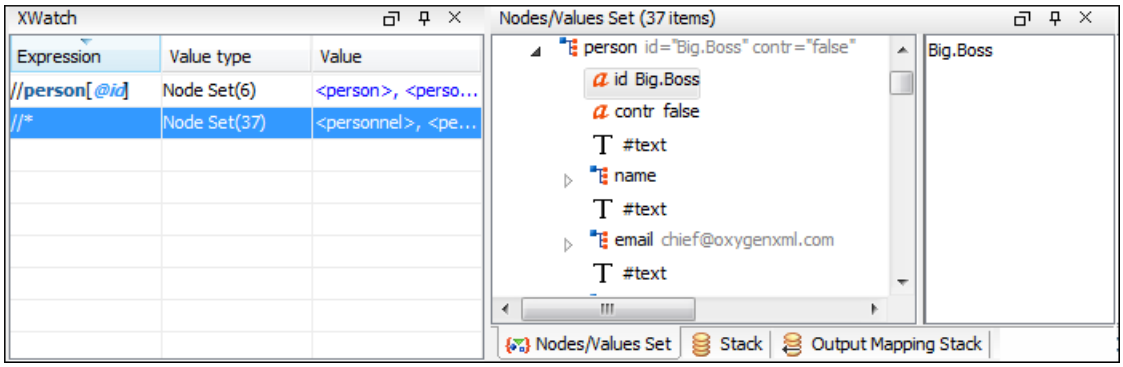


Figure 540: XPath Watch View

Table 40: XWatch columns

Column	Description
Expression	XPath expression to be evaluated (XPath 1.0 or 2.0 / 3.0 compliant).
Value	Result of XPath expression evaluation. Value has a type (see *the possible values* in the section *Variables View*). For *Node Set* results, the number of nodes in the set is shown in parenthesis.

Important: Remarks about working with the **XWatch** view:

- Expressions that reference variable names are not evaluated.
- The expression list is not deleted at the end of the transformation (it is preserved between debugging sessions).
- To insert a new expression, click the first empty line of the **Expression** column and start typing. As an alternative, right-click and select the **Add** action. Press **(Enter)** on the cell to add and evaluate.
- To delete an expression, click its **Expression** column and delete its content. As an alternative, right-click and select the **Remove** action. Press **(Enter)** on the cell to commit changes.
- If the expression result type is a *Node Set*, click it (**Value** column) and its value is displayed in the ***Nodes/ Values Set*** *view*.
- The **Copy**, **Add**, **Remove** and **Remove All** actions are available in every row's contextual menu.

Breakpoints View

The **Breakpoints** view lists all *breakpoints* that are set on opened documents. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu. *Breakpoints can be inserted* in XSLT/XQuery documents and XML documents in XSLT/XQuery debugging sessions.

Once you insert a *breakpoint*, it is automatically added to the list in the **Breakpoints** view and you can edit its associated *condition*. A *breakpoint* can have an associated break condition that represents an XPath expression evaluated in the current debugger context. In order to be processed, their evaluation result should be a boolean value. A *breakpoint* with an associated condition only stops the execution of the Debugger if the *breakpoint condition* is evaluated as **true**.

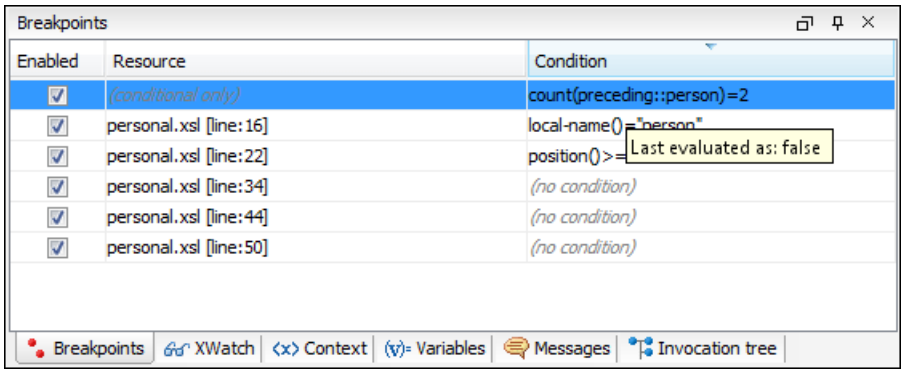


Figure 541: Breakpoints View

The **Breakpoints** view contains the following columns:

- **Enabled** - If selected, the current condition is evaluated and taken into account.
- **Resource** - Resource file and number of the line where the *breakpoint* is set. The Entire path of resource file is available as tooltip.
- **Condition** - XSLT/XQuery expression to be evaluated during debugging. The expression will be evaluated at every debug step.

Clicking a record highlights the *breakpoint* line in the document.

Note: The *breakpoints* list is not deleted at the end of a transformation (it is preserved between debugging sessions).

The following actions are available in the contextual menu of the table:

Go to

Moves the cursor to the source of the *breakpoint*.

Run to Breakpoint

Runs the debugger up to the point of this particular *breakpoint* and ignores the others (regardless of whether they were previously enabled or disabled).

Enable

Enables the *breakpoint*.

Disable

Disables the *breakpoint*. A disabled *breakpoint* will not be evaluated by the Debugger.

Add

Allows you to add a new *breakpoint* and *breakpoint condition*.

Edit

Allows you to edit an existing *breakpoint*.

Remove

Deletes the selected *breakpoint*.

Enable all

Enables all *breakpoints*.

Disable all

Disables all *breakpoints*.

Remove all

Removes all *breakpoints*.

Related Information:

Inserting Breakpoints on page 1156

Messages View

xsl:message instructions are one way to signal special situations encountered during transformation as well as a raw way of doing the debugging. The **Messages** view is available only for XSLT debugging sessions and shows all `xsl:message` calls executed by the XSLT processor during transformation. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

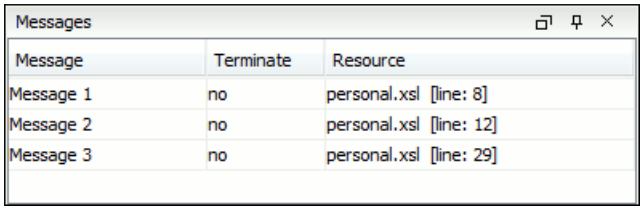


Figure 542: Messages View

Table 41: Messages columns

Column	Description
Message	Message content.
Terminate	Signals if processor terminates the transformation or not once it encounters the message (yes/no respectively).
Resource	Resource file where *xsl:message* instruction is defined and the message line number. The complete path of the resource is available as tooltip.

The following actions are available in the contextual menu:

Go to

Highlight the XSL fragment that generated the message.

Copy

Copies to clipboard message details (system ID, severity info, description, start location, terminate state).

Important:

- Clicking a record from the table highlights the `xsl:message` declaration line.
- Message table values can be sorted by clicking the corresponding column header. Clicking the column header switches the sorting order between: ascending, descending, no sort.

Stack View

The **Stack** view shows the current execution stack of both source and XSLT/XQuery nodes. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

During transformation two stacks are managed: one of source nodes being processed and the other for XSLT/XQuery nodes being processed. Oxygen XML Editor shows both node types into one common stack. The source (XML) nodes are preceded by a red color icon while XSLT/XQuery nodes are preceded by a green color icon. The advantage of this approach is that you can always see the source scope on which an XSLT/XQuery instruction is executed (the last red color node on the stack). The stack is oriented upside down.

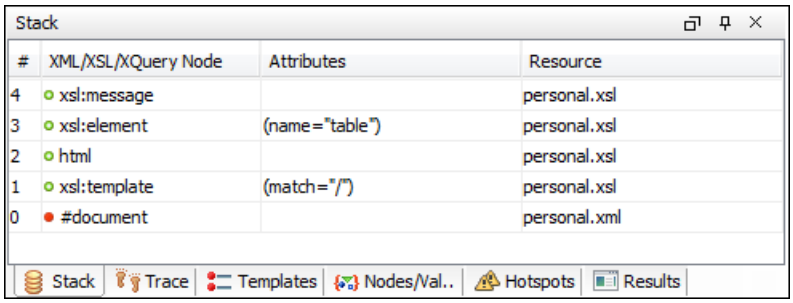


Figure 543: Stack View

The contextual menu contains one action: **Go to**, which moves the selection in the editor panel to the line containing the XSLT element that is displayed on the selected line from the view.

Table 42: Stack columns

Column	Description
#	Order number, represents the depth of the node (0 is the stack base).
XML/XSLT/XQuery Node	Node from source or stylesheet document currently being processed. One particular stack node is the document root, noted as **#document**.
Attributes	Attributes of the node (a list of *id="value"* pairs).
Resource	Resource file where the node is located. The entire path is available as tooltip.

Important: Remarks:

- Clicking a record from the stack highlights that node's location inside resource.
- Using Saxon, the stylesheet elements are qualified with XSL proxy, while using Xalan you only see their names. (example: `xsl:template` using Saxon and `template` using Xalan).
- Only the Saxon processor shows element attributes.
- The Xalan processor shows also the built-in rules.

Output Mapping Stack View

The **Output Mapping Stack** view displays *context data* and presents the XSLT templates/XQuery elements that generated specific areas of the output. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

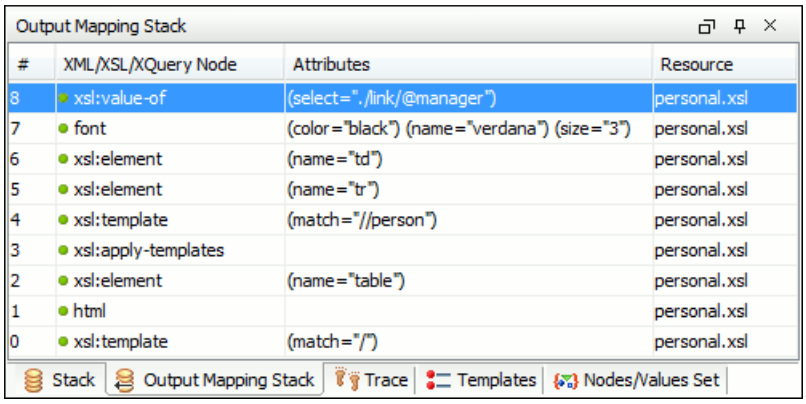


Figure 544: Output Mapping Stack view

The **Go to** action of the contextual menu takes you in the editor panel at the line containing the XSLT element displayed in the **Output Mapping Stack** view.

Table 43: Output Mapping Stack columns

Column	Description
#	The order number in the stack of XSLT templates/XQuery elements. Number 0 corresponds to the bottom of the stack in the status of the XSLT/XQuery processor. The highest number corresponds to the top of the stack.
XSL/XQuery Node	The name of an XSLT template/XQuery element that participated in the generation of the selected output area.
Attributes	The attributes of the XSLT template/XQuery node.
Resource	The name of the file containing the XSLT template/XQuery element.

Important: Remarks:

- Clicking a record highlights that XSLT template definition/XQuery element inside the resource (XSLT stylesheet file/XQuery file).
- Saxon only shows the applied XSLT templates having at least one hit from the processor. Xalan shows all defined XSLT templates, with or without hits.
- The table can be sorted by clicking the corresponding column header. When clicking a column header the sorting order switches between: ascending, descending, no sort.
- Xalan shows also the built-in XSLT rules.

Related Information:

Identify the XSLT / XQuery Expression that Generated Particular Output on page 1157
Stack View on page 1150
Trace History View on page 1151
Templates View on page 1153

Trace History View

Usually the XSLT/XQuery processors signal the following events during transformation:

- - Entering a source (XML) node.
- - Leaving a source (XML) node.
- - Entering an XSLT/XQuery node.

- ⇦ - Leaving an XSLT/XQuery node.

The **Trace History** view catches all these events, so you can see how the process evolved. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

The red icon lines denote source nodes while the green icon lines denote XSLT/XQuery nodes. It is possible to save the element trace in a structured XML document. The action is available on the contextual menu of the view. Thus, you have the possibility of comparing the trace results from multiple debug sessions.

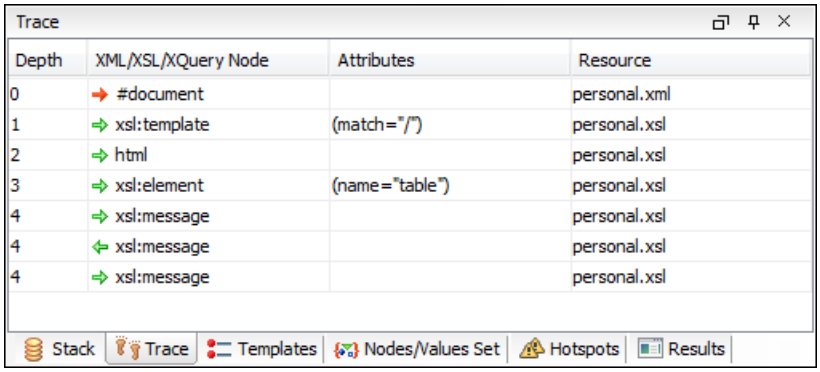


Figure 545: Trace History View

The contextual menu contains the following actions:

Go to

Moves the selection in the editor panel to the line containing the XSLT element or XML element that is displayed on the selected line from the view;

Export to XML

Saves the entire trace list into XML format.

Table 44: Trace History columns

Column	Description
Depth	Shows you how deep the node is nested in the XML or stylesheet structure. The bigger the number, the more nested the node is. A depth 0 node is the document root.
XML/XSLT/XQuery Node	Represents the node from the processed source or stylesheet document. One particular node is the document root, noted as *#document*. Every node is preceded by an arrow that represents what action was performed on it (entering or leaving the node).
Attributes	Attributes of the node (a list of *id="value"* pairs).
Resource	Resource file where the node is located. The complete path of the resource file is provided as tooltip.

Important: Remarks:

- Clicking a record highlights that node's location inside the resource.
- Only the Saxon processor shows the element attributes.
- The Xalan processor shows also the built-in rules.

Templates View

The **xsl:template** is the basic element for stylesheets transformation. The **Templates** view is only available during XSLT debugging sessions and shows all *xsl:template* instructions used by the transformation. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Being able to see the number of *hits* for each of the templates allows you to get an idea of the stylesheet coverage by template rules with respect to the input source.

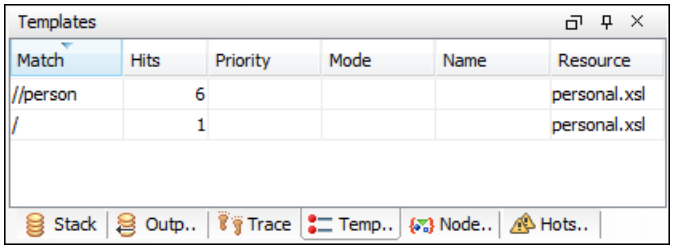


Figure 546: Templates view

The contextual menu contains one action: **Go to**, which moves the selection in the editor panel to the line containing the XSLT template that is displayed on the selected line from the view.

Table 45: Templates columns

Column	Description
Match	The *match* attribute of the *xsl:template*.
Hits	The number of hits for the *xsl:template*. Shows how many times the XSLT processor used this particular template.
Priority	The template priority as established by XSLT processor.
Mode	The *mode* attribute of the *xsl:template*.
Name	The *name* attribute of the *xsl:template*.
Resource	The resource file where the template is located. The complete path of the resource file is available as tooltip.

Important: Remarks:

- Clicking a record highlights that template definition inside the resource.
- Saxon only shows the applied templates having at least one hit from the processor. Xalan shows all defined templates, with or without hits.
- Template table values can be sorted by clicking the corresponding column header. When clicking a column header the sorting order switches between: ascending, descending, no sort.
- Xalan shows also the built-in rules.

Nodes/Values Set View

The **Nodes/Values Set** view is always used in relation with *The **Variables** view* and *the **XWatch** view*. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu. It shows an XSLT node set value in a tree form. The node set view is updated as response to the following events:

- You click a variable having a node set value in one of the above 2 views.
- You click a tree fragment in one of the above 2 views.
- You click an XPath expression evaluated to a node set in one of the above 2 views.

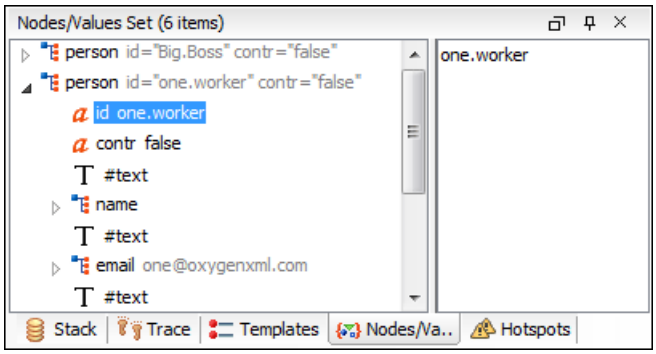


Figure 547: Node Set view

The nodes / values set is presented in a tree-like fashion. The total number of items is presented in the title bar. Nodes from a defined namespace bound to a prefix are displayed using the qualified name. If the namespace is not bound to a prefix the namespace URI is presented before the node name. The value of the selected attribute or node is displayed in the right side panel.

Important: Remarks:

- For longer values in the right side panel, the interface displays it with an ellipsis (...) at the end. A more detailed value is available as a tooltip when hovering over it.
- Clicking a record highlights the location of that node in the source or stylesheet view.

Variables View

The **Variables** view displays variables and parameters (local and global), along with their values. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Variables and parameters play an important role during an XSLT/XQuery transformation. Oxygen XML Editor uses the following icons to differentiate variables and parameters:

- V - Global variable.
- {V} - Local variable.
- P - Global parameter.
- {P} - Local parameter.

The following value types are available:

- **Boolean**
- **String**
- **Date** - XSLT 2.0 / 3.0 only.
- **Number**
- **Set**
- **Object**
- **Fragment** - Tree fragment.
- **Any**
- **Undefined** - The value was not yet set, or it is not accessible.

  Note:

  When Saxon 6.5 is used, if the value is unavailable, then the following message is displayed in the **Value** field: "The variable value is unavailable".

  When Saxon 9 is used:

  - If the variable is not used, the **Value** field displays "The variable is declared but never used".
  - If the variable value cannot be evaluated, the **Value** field displays "The variable value is unavailable".
- **Document**
- **Element**
- **Attribute**

- **ProcessingInstruction**
- **Comment**
- **Text**
- **Namespace**
- **Evaluating** - Value under evaluation.
- **Not Known** - Unknown types.

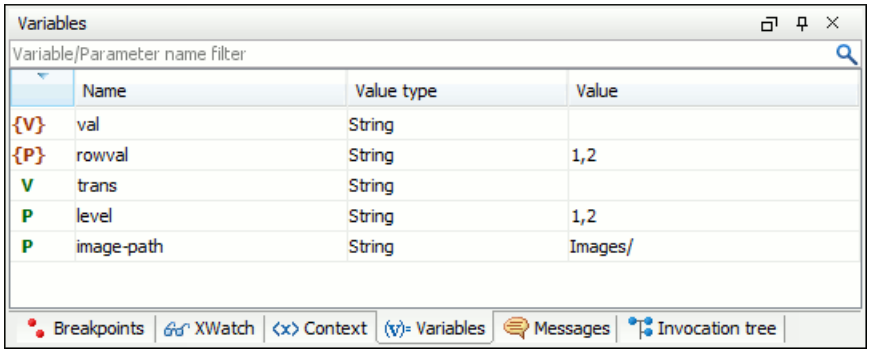


Figure 548: Variables View

Table 46: Variables Columns

Column	Description
Name	Name of variable / parameter.
Value Type	Type of variable/parameter.
Value	Current value of variable / parameter.

The value of a variable (the **Value** column) can be copied to the clipboard for pasting it to other editor area with the action **Copy value** from the contextual menu of the table from the view. This is useful in case of long and complex values that are not easy to remember by looking at them once.

Important: Remarks:

- Local variables and parameters are the first entries presented in the table.
- Clicking a record highlights the variable definition line.
- Variable values could differ depending on the transformation engine used or stylesheet version set.
- If the value of the variable is a node set or a tree fragment, clicking it causes the ***Node Set*** *view* to be shown with the corresponding set of values.
- Variable table values can be sorted by clicking the corresponding column header. Clicking the column header switches between the orders: ascending, descending, no sort.

Multiple Output Documents in XSLT 2.0 and XSLT 3.0

For XSLT 2.0 and XSLT 3.0 stylesheets that store the output in multiple files by using the `xsl:result-document` instruction, the content of the file created in this way is displayed dynamically while the transformation is running in an output view. There is one view for each `xsl:result-document` instruction so that the output is not mixed while still being presented in multiple views.

Working with the XSLT / XQuery Debugger

The topics in this section describe how to work with the debugger in some of the most common use cases. The information includes the steps typically involved in a debugging process, how to use *breakpoints*, and how to identify an expression that generates a particular output.

For even more information, watch our ***XSLT Debugger*** *video demonstration*.

Steps in a Typical Debugging Process

To debug a stylesheet or XQuery document, follow this procedure:

1. *Open the source XML document* and *the XSLT/XQuery document.*
2. If you are in the **Editor** *perspective*, switch to the **XSLT Debugger** or **XQuery Debugger** *perspective* with one of the following actions:
   - Select **Window** > **Open perspective** > **XSLT Debugger/XQuery Debugger** or the **XSLT Debugger/XQuery Debugger** button in the top-right corner of the interface.
   - Select **Document** > **XML Document** > **Debug scenario** or use the **Debug scenario** action on the toolbar.. This action initializes the Debugger *perspective* with the parameters of the transformation scenario. Any modification applied to the scenario parameters (the transformer engine, XSLT parameters, transformer extensions, etc.) will be saved back in the scenario when exiting from the Debugger *perspective*.
3. Select the source XML document in the XML source selector of *the Control toolbar*. In the case of XQuery debugging, if your XQuery document has no implicit source, set the source selector value to **NONE**.
4. Select the XSLT/XQuery document in the XSLT/XQuery selector of *the Control toolbar*.
5. Set XSLT/XQuery parameters from the button available on *the Control toolbar*.
6. *Set one or more breakpoints*.
7. Step through the stylesheet using the following buttons available on *the Control toolbar*:
   - **Step into**
   - **Step over**
   - **Step out**
   - **Run**
   - **Run to cursor**
   - **Run to end**
   - **Pause**
   - **Stop**
8. Examine the information in the information views to find the bug in the transformation process.

   You may find *the procedure for determining the XSLT template/XQuery element that generated an output section* useful for fixing bugs in the transformation.

Related Information:

Identify the XSLT / XQuery Expression that Generated Particular Output on page 1157

Using Breakpoints

The Oxygen XML Editor XSLT/XQuery Debugger allows you to interrupt XSLT/XQuery processing to gather information about variables and processor execution at particular points. To ensure *breakpoints* are persistent between work sessions, they are saved at project level. You can set a maximum of 100 *breakpoints* per project.

Related Information:

Breakpoints View on page 1148

Inserting Breakpoints

Breakpoints can be set in XSLT/XQuery documents and XML documents in XSLT/XQuery debugging sessions.

To insert a *breakpoint*, follow these steps:

1. Click the line where you want to insert the *breakpoint* in the XML source document or the XSLT/XQuery document.

   You can only set *breakpoints* on the XML source in XSLT or XQuery debugging sessions.

   Breakpoints are automatically created on the ending line of a start tag, even if you click a different line.

2. Click the vertical stripe on the left side of the editor panel or select **Create Breakpoint (Shift+F7)** from the **Edit** > **Breakpoints** menu.

   Once you insert a *breakpoint*, it is automatically added to the list in the **Breakpoints** view and you can edit its associated *condition*. A *breakpoint* can have an associated break condition that represents an XPath expression evaluated in the current debugger context. In order to be processed, their evaluation result should be a boolean value. A *breakpoint* with an associated condition only stops the execution of the Debugger if the *breakpoint condition* is evaluated as **true**.

```
4   <title></title>
5   <body>
6     <p></p>
7     <table frame="all" rowsep="1"
8       <title>test</title>
9       <tgroup cols="2">
```

Figure 549: Example: Breakpoints

Related Information:

Breakpoints View on page 1148

Removing Breakpoints

Only one action is required to remove a *breakpoint*:

Click the *breakpoint* icon (●) in the vertical stripe on the left side of the editor panel or right-click the *breakpoint* and select **Renove** or **Remove all**.

Identify the XSLT / XQuery Expression that Generated Particular Output

To quickly spot the XSLT templates or XQuery expressions with problems it is important to know what XSLT template in the XSLT stylesheet (or XQuery expression in the XQuery document) and what element in the source XML document generated a specified area in the output.

Some of the debugging capabilities (for example, *Step in*) can be used for this purpose. Using *Step in* you can see how output is generated and link it with the XSLT/XQuery element being executed in the current source context. However, this can become difficult on complex XSLT stylesheets or XQuery documents that generate a large output.

You can click the text from the **Text** output view or **XHTML** output view and the editor will select the XML source context and the XSLT template/XQuery element that generated the text. Also, inspecting the whole stack of XSLT templates/XQuery elements that determined the state of the XSLT/XQuery processor at the moment of generating the specified output area speeds up the debugging process.

1. Switch to the **XSLT Debugger** or **XQuery Debugger** *perspective* with one of the following actions:
   - Select **Window** > **Open perspective** > **XSLT Debugger/XQuery Debugger** or the **XSLT Debugger/XQuery Debugger** button in the top-right corner of the interface.
   - Select **Document** > **XML Document** > **Debug scenario** or use the **Debug scenario** action on the toolbar.. This action initializes the Debugger *perspective* with the parameters of the transformation scenario. Any modification applied to the scenario parameters (the transformer engine, XSLT parameters, transformer extensions, etc.) will be saved back in the scenario when exiting from the Debugger *perspective*.
2. Select the source XML document in the XML source selector of *the Control toolbar*. In the case of XQuery debugging without an implicit source choose the **NONE** value.
3. Select the XSLT/XQuery document in the XSLT/XQuery selector of *the Control toolbar*.
4. Select the XSLT/XQuery engine in the XSLT/XQuery engine selector of *the Control toolbar*.
5. Set XSLT/XQuery parameters from the button available on *the Control toolbar*.
6. Apply the XSLT stylesheet or XQuery transformation using the **Run to end** button that is available on *the Control toolbar*.
7. Inspect the mapping by clicking a section of the output from the **Text** view tab or from the **XHTML** view tab of the *Output document view* .

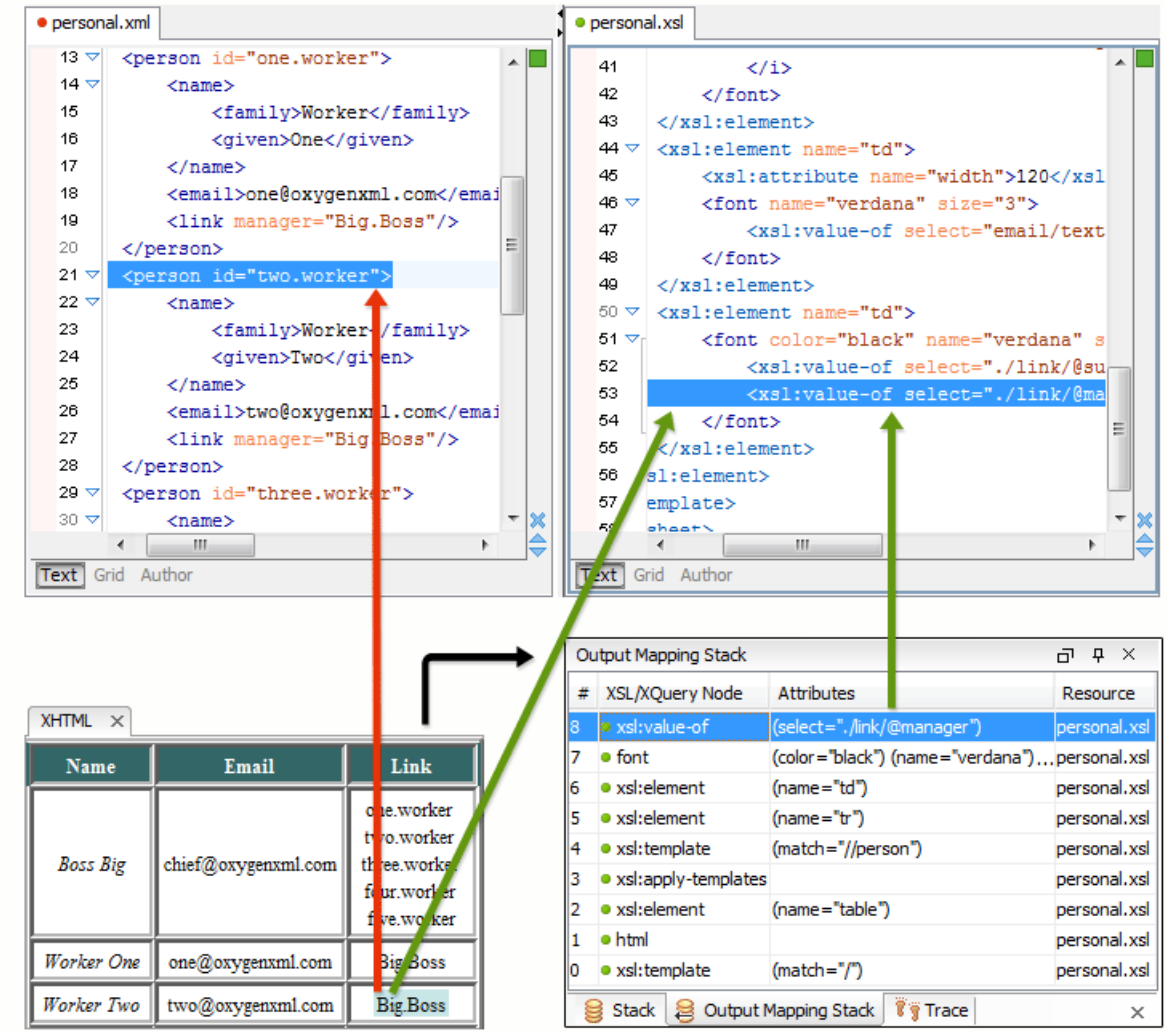


Figure 550: XHTML Output to Source Mapping

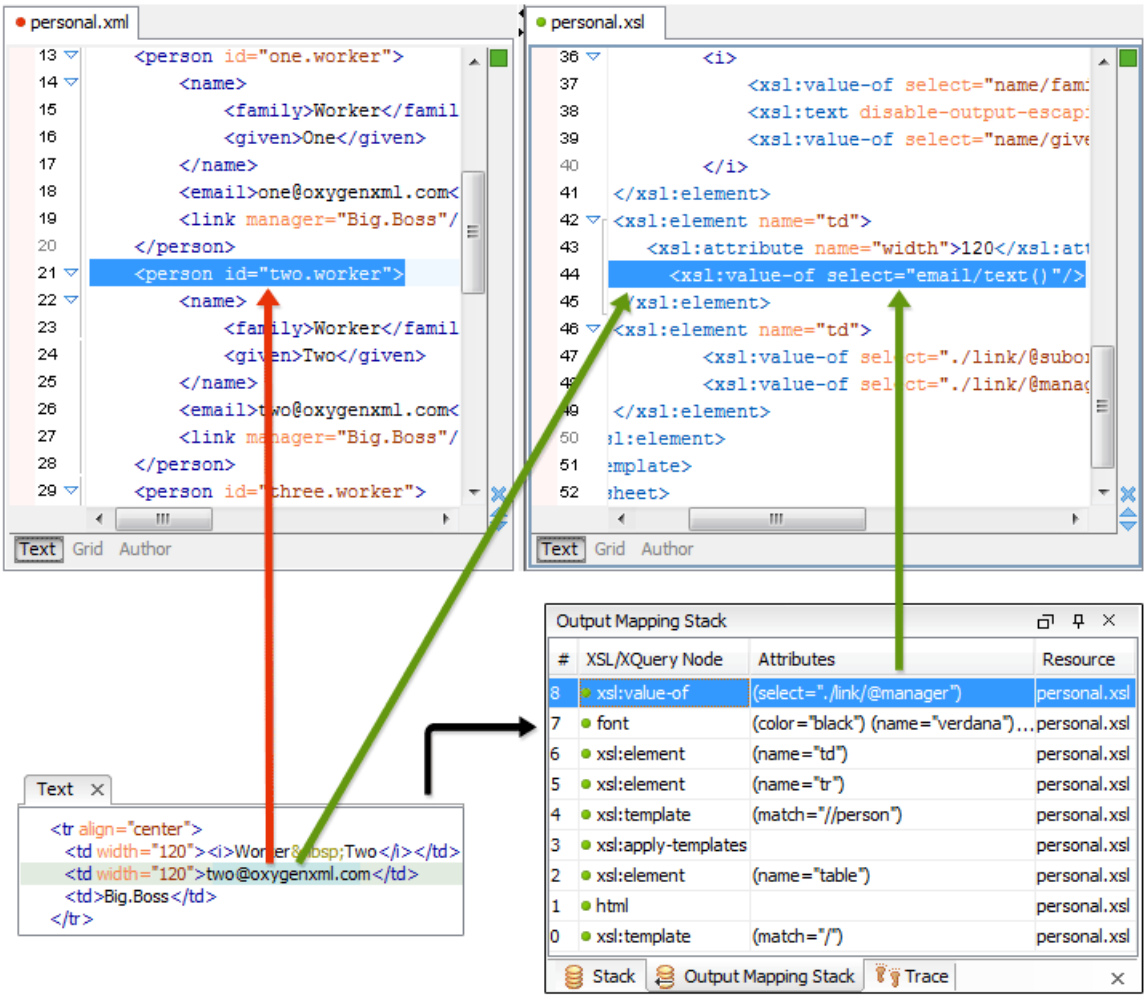


Figure 551: Text Output to Source Mapping

This action will highlight the XSLT / XQuery element and the XML source context. This XSLT template/XQuery element that is highlighted in the XSLT/XQuery editor represents only the top of the stack of XSLT templates/ XQuery elements that determined the state of the XSLT/XQuery processor at the moment of generating the clicked output section. In the case of complex transformations inspecting the whole stack of XSLT templates/ XQuery elements speeds up the debugging process. This stack is available in *the **Output Mapping Stack** view*.

Related Information:

Output Mapping Stack View on page 1150
Trace History View on page 1151
Templates View on page 1153

Debugging Java Extensions

The XSLT/XQuery debugger does not step into Java classes that are configured as XSLT/XQuery extensions of the transformation. To step into Java classes, inspect variable values, and set *breakpoints* in Java methods, you can set up a Java debug configuration in an IDE (such as the Eclipse SDK) as described in the following steps:

1. Create a debug configuration.
   a) Set at least 256 MB as heap memory for the Java virtual machine (recommended 512 MB) by setting the `-Xmx` parameter in the debug configuration (for example, “-Xmx512m”).

b) Make sure the `[OXYGEN_INSTALL_DIR]/lib/oxygen.jar` file and your Java extension classes are on the Java classpath.

   The Java extension classes should be the same classes that were *set as an extension* of the XSLT/XQuery transformation in the debugging *perspective*.

c) Set the class `ro.sync.exml.Oxygen` as the main Java class of the configuration.

   The main Java class `ro.sync.exml.Oxygen` is located in the `oxygen.jar` file.

2. Start the debug configuration.

Now you can set *breakpoints* and inspect Java variables as in any Java debugging process executed in the selected IDE (Eclipse SDK, and so on.).

Supported Processors for XSLT / XQuery Debugging

The following built-in XSLT processors are integrated in the debugger and can be selected in the *Control Toolbar*:

- Saxon 9.7.0.15 HE (Home Edition) - a limited version of the Saxon 9 processor, capable of running XSLT 1.0, XSLT 2.0 / 3.0 basic and XQuery 1.0 transformations, available in both the XSLT debugger and the XQuery one,
- Saxon 9.7.0.15 PE (Professional Edition) - capable of running XSLT 1.0 transformations, XSLT 2.0 basic ones and XQuery 1.0 ones, available in both the XSLT debugger and the XQuery one,
- Saxon 9.7.0.15 EE (Enterprise Edition) - a schema aware processor, capable of running XSLT 1.0 transformations, XSLT 2.0 /3.0 basic ones, XSLT 2.0 / 3.0 schema aware ones and XQuery 1.0 / 3.0 ones, available in both the XSLT debugger and the XQuery debugger,
- Saxon 6.5.5 - capable of running only XSLT 1.0 transformations, available only in the XSLT debugger,
- Xalan 2.7.1 - capable of running only XSLT 1.0 transformations, available only in the XSLT debugger.

Performance Profiling of XSLT Stylesheets and XQuery Documents

This chapter explains the user interface and how to use the profiler for finding performance problems in XSLT transformations and XQuery ones.

XSLT/XQuery Performance Profiling Overview

Whether you are trying to identify a performance issue that is causing your production XSLT/XQuery transformation to not meet customer expectations or you are trying to proactively identify issues prior to deploying your XSLT/XQuery transformation, using the XSLT/XQuery profiler feature is essential to helping you save time and ultimately ensure a better performing, more scalable XSLT/XQuery transformation.

The XSLT/XQuery profiling feature can use any available XSLT/XQuery processors that could be used for debugging and it is available from the debugging *perspective*.

Enabling and disabling the profiler is controlled by the ***Profiler*** *button* from the *debugger control toolbar*. The XSLT/XQuery profiler is off by default. This option is not available during a debugger session so you should set it before starting the transformation.

Working with XSLT/XQuery Profiler

Profiling activity is linked with debugging activity, so the first step in profiling is to switch to the debugging *perspective* and follow the corresponding procedure for debugging (see *Steps in a Typical Debugging Process* on page 1156).

Immediately after turning the profiler on two new information views are added to the current debugger *information views*:

- ***Invocation tree*** *view* on left side
- ***Hotspots*** *view* on right side

Profiling data is available only after the transformation ends successfully.

Looking to the right side (***Hotspots** view*), you can immediately spot the time the processor spent in each instruction. As an instruction usually calls other instructions the used time of the called instruction is extracted from the duration time of the caller (the hotspot only presents the inherent time of the instruction).

Looking to the left side (***Invocation tree** view*), you can examine how style instructions are processed. This result view is also named call-tree, as it represents the order of style processing. The profiling result shows the duration time for each of the style-instruction including the time needed for its called children.

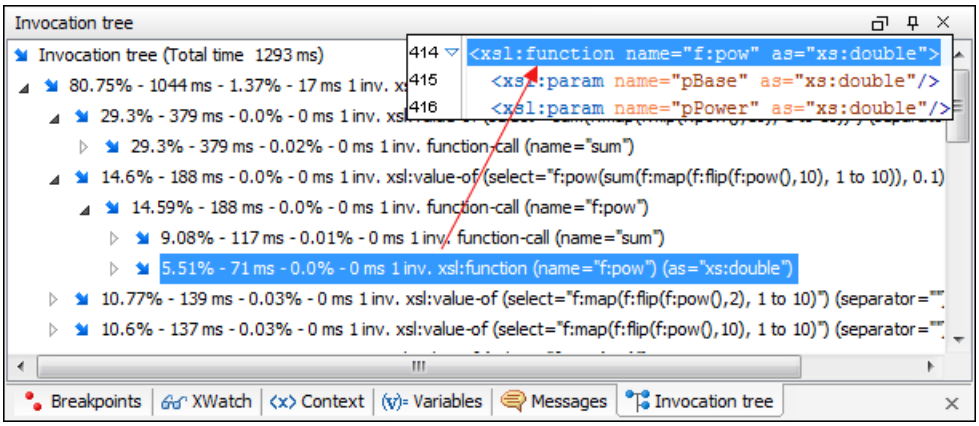


Figure 552: Source backmapping

In any of the above views you can use the backmapping feature to find the XSLT stylesheet or XQuery expression definition. Clicking the selected item cause Oxygen XML Editor to highlight the XSLT stylesheet or XQuery expression source line where the instruction is defined.

When navigating through the trees by opening instruction calls, Oxygen XML Editor automatically expands instructions that are only called by one other instruction themselves.

The profiling data can be saved into XML and HTML format. On any of the above views, use the contextual menu and select the corresponding choice. Basically saving HTML means saving XML and applying an XSLT stylesheet to render the report as XML. These stylesheets are included in the Oxygen XML Editor distribution (see the subfolder `[OXYGEN_INSTALL_DIR]/frameworks/profiler/`) so you can make your own report based on the profiling raw data.

If you want to change the *XSLT/XQuery profiler settings*, use the contextual menu and choose the corresponding **View settings** entry.

CAUTION: Profiling exhaustive transformation may run into an OutOfMemory error due to the large amount of information being collected. If this is the case you can close unused projects when running the profiling or use high values for Java VM options `-Xms` and `-Xmx`. If this does not help you can shorten your source XML file and try again.

To watch our video demonstration about the XSLT/XQuery Profiler, go to *https://www.oxygenxml.com/demo/XSLT_Profiling.html*.

Invocation Tree View

The **Invocation Tree** view shows a top-down call tree that represents how XSLT instructions or XQuery expressions are processed. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

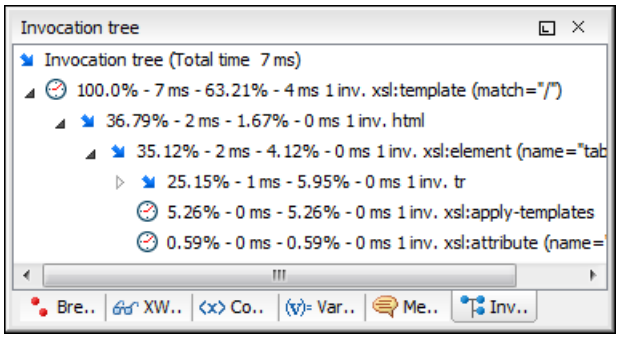


Figure 553: Invocation Tree View

The entries in the invocation tree include a few possible icons that indicate the following:

- - Points to a call whose inherent time is insignificant compared to its total time.
- - Points to a call whose inherent time is significant compared to its total time (greater than 1/3rd of its total time).

Every entry in the invocation tree includes textual information that depends on the ***XSLT/XQuery profiler settings***:

- A percentage number of the total time that is calculated with respect to either the root of the tree or the calling instruction.
- A total time measurement in milliseconds or microseconds. This is the total execution time that includes calls into other instructions.
- A percentage number of the inherent time that is calculated with respect to either the root of the tree or the calling instruction.
- An inherent time measurement in milliseconds or microseconds. This is the inherent execution time of the instruction.
- An invocation count that shows how often the instruction has been invoked on this call-path.
- An instruction name that contains also the attributes description.

Hotspots View

The **Hotspots** view displays a list of all instruction calls that lie above the threshold defined in the ***XSLT/XQuery profiler settings***. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

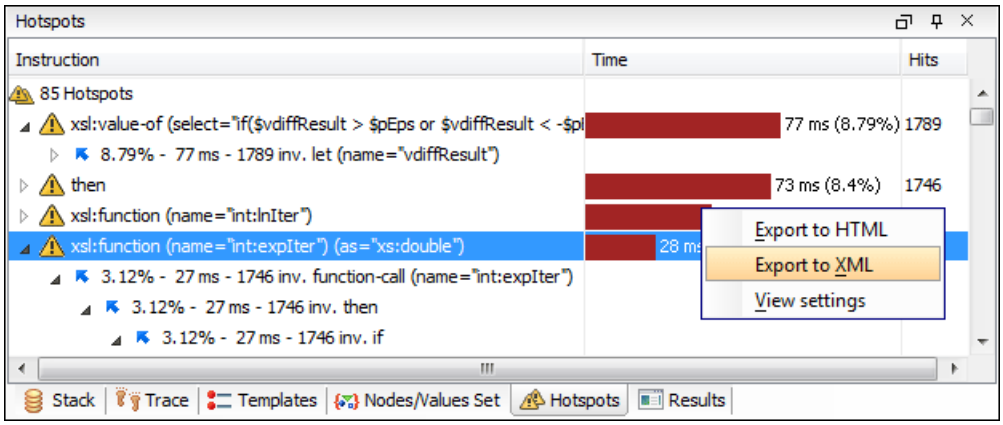


Figure 554: Hotspots View

By opening a hotspot instruction entry, the tree of back-traces leading to that instruction call are calculated and shown.

Every hotspot is described by the values from the following columns:

- **Instruction** - The name of the instruction.

- **Time** - The inherent time in milliseconds or microseconds of how much time has been spent in the hotspot, along with a bar whose length is proportional to this value. All calls into this instruction are summed up regardless of the particular call sequence.
- **Hits** - The invocation count of the hotspot entry.

If you click the ⚠ handle on the left side of a hotspot, a tree of back-traces will be shown.

Every entry in the backtrace tree has textual information attached to it that depends on the ***XSLT/XQuery profiler settings***:

- A percentage number that is calculated with respect to either the total time or the called instruction.
- A time measured in milliseconds or microseconds of how much time has been contributed to the parent hotspot on this call-path.
- An invocation count that shows how often the hotspot has been invoked on this call-path.

  Note: This is not the number of invocations of this instruction.
- An instruction name that also contains its attributes.

Author Mode Customization

15

Topics:

- *Author Mode Customization Guide*
- *CSS Support in Author Mode*
- *Creating and Running Automated Tests*
- *API Frequently Asked Questions (API FAQ)*

This section contains an *Author Mode Customization Guide* on page 1164, *CSS Support in Author Mode* on page 1256, a collection of *Frequently Asked Questions regarding the Oxygen XML Editor API*, and other topics in regards to customizing the **Author** mode.

Author Mode Customization Guide

The **Author** mode editor of Oxygen XML Editor was designed to provide a user-friendly interface for editing XML documents. **Author** combines the power of source editing with the intuitive interface of a word processor. You can customize the **Author** mode editor to support new custom XML formats or to change how standard XML formats are edited.

Although Oxygen XML Editor includes built-in, configured *frameworks* for DocBook, DITA, TEI, and XHTML you might need to create a customization of the editor to handle other types of documents. A common use case is when your organization holds a collection of XML document types used to define the structure of internal documents and they need to be visually edited by people with no experience working with XML files.

There are several ways to customize the editor:

1. Create a CSS file defining styles for the XML elements you will work with, and create XML files that reference the CSS through an `xml-stylesheet` processing instruction.
2. Fully configure a *framework*. This involves putting together the CSS stylesheets, XML schemas, actions, menus, bundling them, and distributing an archive. The CSS and GUI elements are settings for the Oxygen XML Editor **Author** mode. The other settings such as the templates, catalogs, and transformation scenarios are general settings and are enabled whenever the association is active, regardless of the editing mode (**Text**, **Grid**, or **Author**).

Simple Customization Tutorial

The most important elements of a document type customization are represented by an XML Schema to define the XML structure, the CSS to render the information and the XML instance template that links the first two together.

XML Schema

To provide as-you-type validation and to compute valid insertion proposals, Oxygen XML Editor needs an XML grammar (XML Schema, DTD, or RelaxNG) associated to the XML. The grammar specifies how the internal structure of the XML is defined. For information about associating a schema and how Oxygen XML Editor detects the schema, see *Associating a Schema to XML Documents* on page 464.

Consider a use-case in which several users are testing a system and must send report results to a content management system. The customization should provide a visual editor for these kind of documents. The following XML Schema, `test_report.xsd` defines a report with results of a testing session. The report

consists of a title, few lines describing the test suite that was run, and a list of test results (each with a name and a boolean value indicating if the test passed or failed).

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="report">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="title"/>
                <xs:element ref="description"/>
                <xs:element ref="results"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="title" type="xs:string"/>
    <xs:element name="description">
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded">
                <xs:element name="line">
                    <xs:complexType mixed="true">
                        <xs:sequence minOccurs="0"
                            maxOccurs="unbounded">
                            <xs:element name="important"
                              type="xs:string"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="results">
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded">
                <xs:element name="entry">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="test_name"
                              type="xs:string"/>
                            <xs:element name="passed"
                               type="xs:boolean"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

CSS Stylesheet

A set of rules must be defined for describing how the XML document is to be rendered in **Author** mode. This is done using Cascading Style Sheets (CSS). CSS is a language used to describe how an HTML or XML document should be formatted by a browser. CSS is widely used in the majority of websites.

The elements from an XML document are displayed in the layout as a series of boxes. Some of the boxes contain text and may flow one after the other, from left to right. These are called inline boxes. There are also other type of boxes that flow one below the other (such as paragraphs). These are called block boxes.

For example, consider the way a traditional text editor arranges the text. A paragraph is a block, because it contains a vertical list of lines. The lines are also blocks. However, blocks that contains inline boxes arrange its children in a horizontal flow. That is why the paragraph lines are also blocks, while the traditional "bold" and "italic" sections are represented as inline boxes.

The CSS allows us to specify that some elements are displayed as tables. In CSS, a table is a complex structure and consists of rows and cells. The `table` element must have children that have a *table-row* style. Similarly, the `row` elements must contain elements with a *table-cell* style.

To make it easy to understand, the following section describes how each element from a schema is formatted using a CSS file. Note that this is just one of infinite possibilities for formatting the content.

`report`

This element is the root element of a report document. It should be rendered as a box that contains all other elements. To achieve this the display type is set to **block**. Additionally, some margins are set for it. The CSS rule that matches this element is:

```
report{
    display:block;
    margin:1em;
```

```
}
```

`title`

The title of the report. Usually titles have a large font. The **block** display is used so that the subsequent elements will be placed below it, and its font is changed to double the size of the normal text.

```
title {
    display:block;
    font-size:2em;
}
```

`description`

This element contains several lines of text describing the report. The lines of text are displayed one below the other, so the description has the **block** display. Also, the background color is changed to make it standout.

```
description {
    display:block;
    background-color:#EEEEFF;
    color:black;
}
```

`line`

A line of text in the description. A specific aspect is not defined and it just indicates that the display should be **block** style.

```
line {
    display:block;
}
```

`important`

The `important` element defines important text from the description. Since it can be mixed with text, its display property must be set to **inline**. Also, the text is emphasized with **bold**to make it easier to spot.

```
important {
    display:inline;
    font-weight:bold;
}
```

`results`

The `results` element displays the list of *test_names* and the results for each one. To make it easier to read, it is displayed as a **table**, with a green border and margins.

```
results{
    display:table;
    margin:2em;
    border:1px solid green;
}
```

`entry`

An item in the results element. The results are displayed as a table so the entry is a row in the table. Thus, the display is **table-row**.

```
entry {
    display:table-row;
}
```

`test_name`, `passed`

The name of the individual test, and its result. They are cells in the results table with the display set to **table-cell**. Padding and a border are added to emphasize the table grid.

```
test_name, passed{
    display:table-cell;
    border:1px solid green;
    padding:20px;
}

passed{
    font-weight:bold;
```

```
}
```

The full content of the CSS file `test_report.css` is:

```
report {
    display:block;
    margin:1em;
}

description {
    display:block;
    background-color:#EEEEFF;
    color:black;
}

line {
    display:block;
}

important {
    display:inline;
    font-weight:bold;
}

title {
    display:block;
    font-size:2em;
}

results{
    display:table;
    margin:2em;
    border:1px solid green;
}

entry {
    display:table-row;
}

test_name, passed{
    display:table-cell;
    border:1px solid green;
    padding:20px;
}

passed{
    font-weight:bold;
}
```

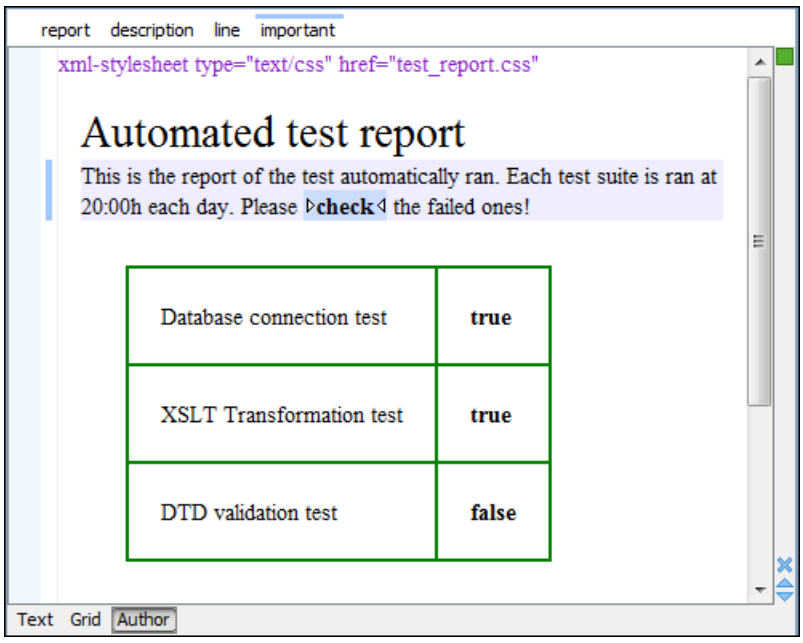


Figure 555: Report Rendered in Author Mode

Note: You can edit attributes in-place in the **Author** mode using *form-based controls*.

Associating a Stylesheet with an XML Document

The rendering of an XML document in the **Author** mode is driven by a CSS stylesheet that conforms to the *version 2.1 of the CSS specification* from the W3C consortium. Some CSS 3 features, such as namespaces and custom extensions, of the CSS specification are also supported. Oxygen XML Editor also supports stylesheets coded with the LESS dynamic stylesheet language.

There are several methods for associating a stylesheet (CSS or LESS) with an XML document:

1. Insert the `xml-stylesheet` processing instruction with the `type` attribute at the beginning of the XML document. If you do not want to alter your XML documents, *you should create a new document type (framework)*.

   CSS example:

   ```
   <?xml-stylesheet type="text/css" href="test.css"?>
   ```

   LESS example:

   ```
   <?xml-stylesheet type="text/css" href="test.less"?>
   ```

   Note: XHTML documents need a `link` element, with the `href` and `type` attributes in the `head` child element, as specified in the *W3C CSS specification*. XHTML example:

   ```
   <link href="/style/screen.css" rel="stylesheet" type="text/css"/>
   ```

   Tip: You can also insert the `xml-stylesheet` processing instruction by using the **Associate XSLT/CSS Stylesheet** action that is available on the toolbar or in the **Document** > **XML Document** menu.
2. Add a new CSS or LESS file to a *framework* (document type). To do so, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Document Type Association**. Edit the appropriate *framework*, open the **Author** tab, then the **CSS** subtab. Press the **New** button to add a new CSS or LESS file.

   Note: The predefined *frameworks* are read-only, so you need to ***Extend*** or ***Duplicate*** them to configure them as custom *frameworks*.

You can read more about associating a CSS to a document type in *Customizing the Main CSS of a Framework*.

If a document has no CSS association or the referenced stylesheet files cannot be loaded, a default one is used. A warning message is also displayed at the beginning of the document, presenting the reason why the CSS cannot be loaded.

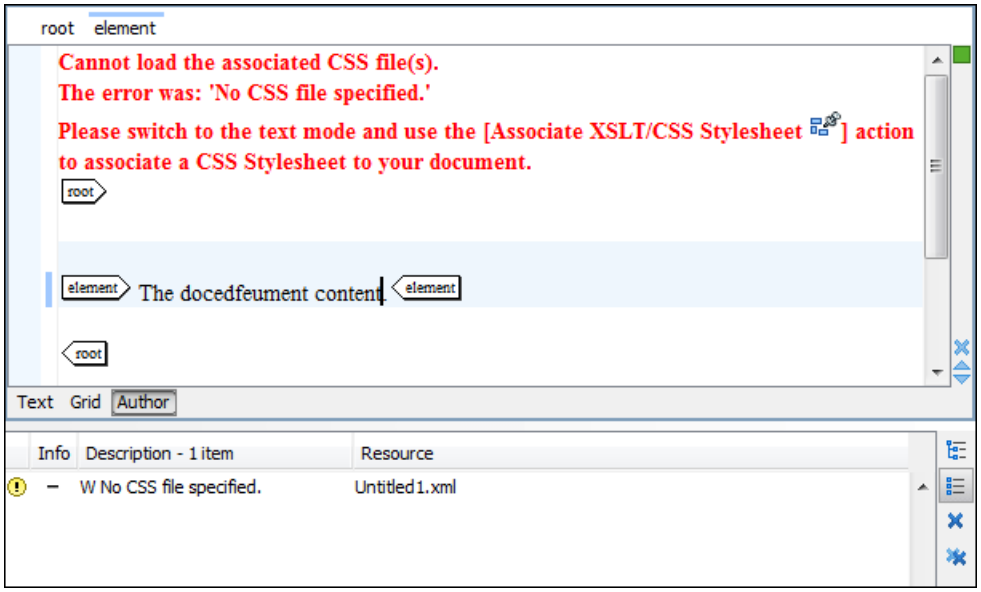


Figure 556: Document with no CSS association default rendering

XML Instance Template

Based on the XML Schema and CSS file Oxygen XML Editor can help the content author in loading, editing, and validating the test reports. An XML file template must be created as a kind of skeleton that the users can use as a starting point for creating new test reports. The template must be generic enough and reference the XML Schema file and the CSS stylesheet.

This is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="test_report.css"?>
<report xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="test_report.xsd">
  <title>Automated test report</title>
  <description>
    <line>This is the report of the test automatically ran.
      Each test suite is ran at 20:00h each day.
      Please <important>check</important> the failed ones!</line>
  </description>
  <results>
    <entry>
      <test_name>Database connection test</test_name>
      <passed>true</passed>
    </entry>
    <entry>
      <test_name>XSLT Transformation test</test_name>
      <passed>true</passed>
    </entry>
    <entry>
      <test_name>DTD validation test</test_name>
      <passed>false</passed>
    </entry>
  </results>
</report>
```

The processing instruction `xml-stylesheet` associates the CSS stylesheet to the XML file. The `href` pseudo attribute contains the URI reference to the stylesheet file. In our case the CSS is in the same directory as the XML file.

The next step is to place the XSD file and the CSS file on a web server and modify the template to use the HTTP URLs, like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css"
        href="http://www.mysite.com/reports/test_report.css"?>
<report xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation=
        "http://www.mysite.com/reports/test_report.xsd">
    <title>Test report title</title>
    <description>
.......
```

If you want to share the files with other team members, you could create an archive containing the *`test_report.xml`*, *`test_report.css`*, and *`test_report.xsd`* and send it to the other users.

Advanced Framework Customization

Oxygen XML Editor supports individual document types and classes of document types through *frameworks*. A *framework* associates a document type or a class of documents with CSS stylesheets, validation schemas, catalog files, new files templates, transformation scenarios and custom actions.

In this tutorial, we create a *framework* for a set of documents. As an example, we create a light documentation *framework* (similar to DocBook), then we set up a complete customization of the **Author** mode.

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

Related Information:

Document Types and Frameworks on page 763
Predefined Document Types (Frameworks) on page 763

Creating the Basic Association

Let us go through an example of creating a custom document type (*framework*) and editing an XML document of this type.

First Step - XML Schema

To illustrate an example of creating an XML Schema a custom DocBook *framework*, suppose the documents are either `articles` or `books`, both composed of `sections`. The `sections` may contain `titles`, `paragraphs`, `figures`, `tables`, and other `sections`. To complete the picture, each section includes a `def` element from another namespace.

The first schema file:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.oxygenxml.com/sample/documentation"
    xmlns:doc="http://www.oxygenxml.com/sample/documentation"
    xmlns:abs="http://www.oxygenxml.com/sample/documentation/abstracts"
    elementFormDefault="qualified">

    <xs:import namespace=
    "http://www.oxygenxml.com/sample/documentation/abstracts"
     schemaLocation=
    "abs.xsd"/>
```

The namespace of the documents will be `http://www.oxygenxml.com/sample/documentation`. The namespace of the `def` element is `http://www.oxygenxml.com/sample/documentation/abstracts`.

Next, we define the structure of the sections. They all start with a title, then have the optional `def` element then either a sequence of other sections, or a mixture of paragraphs, images and tables.

```
<xs:element name="book" type="doc:sectionType"/>
<xs:element name="article" type="doc:sectionType"/>
<xs:element name="section" type="doc:sectionType"/>

<xs:complexType name="sectionType">
    <xs:sequence>
        <xs:element name="title" type="xs:string"/>
        <xs:element ref="abs:def" minOccurs="0"/>
        <xs:choice>
            <xs:sequence>
                <xs:element ref="doc:section" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:choice maxOccurs="unbounded">
                <xs:element ref="doc:para"/>
                <xs:element ref="doc:image"/>
                <xs:element ref="doc:table"/>
            </xs:choice>
        </xs:choice>
    </xs:sequence>
</xs:complexType>
```

The paragraph contains text and other styling markup, such as bold (b) and italic (`i`) elements.

```
<xs:element name="para" type="doc:paragraphType"/>

<xs:complexType name="paragraphType" mixed="true">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="b"/>
        <xs:element name="i"/>
    </xs:choice>
</xs:complexType>
```

The `image` element has an attribute with a reference to the file containing image data.

```
<xs:element name="image">
    <xs:complexType>
        <xs:attribute name="href" type="xs:anyURI" use="required"/>
    </xs:complexType>
</xs:element>
```

The `table` contains a header row and then a sequence of rows (`tr` elements) each of them containing the cells. Each cell has the same content as the paragraphs.

```
 <xs:element name="table">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="header">
            <xs:complexType>
               <xs:sequence>
                   <xs:element name="td" maxOccurs="unbounded"
                        type="doc:paragraphType"/>
               </xs:sequence>
            </xs:complexType>
```

```
        </xs:element>
        <xs:element name="tr" maxOccurs="unbounded">
            <xs:complexType>
               <xs:sequence>
                   <xs:element name="td" type="doc:tdType"
                         maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:complexType name="tdType">
    <xs:complexContent>
        <xs:extension base="doc:paragraphType">
            <xs:attribute name="row_span" type="xs:integer"/>
            <xs:attribute name="column_span" type="xs:integer"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The `def` element is defined as a text only element in the imported schema `abs.xsd`:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace=
     "http://www.oxygenxml.com/sample/documentation/abstracts">
    <xs:element name="def" type="xs:string"/>
</xs:schema>
```

Now the XML data structure will be styled.

Schema Settings

In the bottom section of the ***Document Type*** *configuration dialog box*, there are a series of tabs. The first one refers to the schema that is used for validation of the documents that match the defined **Association Rules**.

Important: If the document references a schema using a `DOCTYPE` declaration or a `xsi:schemaLocation` attribute, the schema from the document type configuration will not be used when validating.

Schema Type

Select from the combo box the value **XML Schema**.

Schema URI

Enter the value of the schema location (for example, `${frameworks}/sdf/schema/sdf.xsd`). Use the *${frameworks} editor variable* in the schema URI path instead of a full path to be valid for multiple Oxygen XML Editor installations.

Important: The *${frameworks}* variable is expanded at the time of validation into the absolute location of the directory containing the *framework*.

Second Step - CSS

If you read the *Simple Customization Tutorial* then you already have some basic notions about creating simple styles. The example document contains elements from various namespaces, so you need to use CSS Level 3 extensions (supported by the **Author** mode layout engine) to associate specific properties with that element.

Defining the General Layout

Now the basic layout of the rendered documents is created.

Elements that are stacked one on top of the other are: `book`, `article`, `section`, `title`, `figure`, `table`, `image`. These elements are marked as having `block` style for display. Elements that are placed one after the other in a flowing sequence are: `b`, `i`. These will have `inline` display.

```
/* Vertical flow */
book,
section,
para,
title,
image,
ref {
    display:block;
}

/* Horizontal flow */
b,i {
```

```
    display:inline;
}
```

Important: Having `block` display children in an `inline` display parent results in Oxygen XML Editor changing the style of the parent to `block` display.

Styling the `section` Element

The title of any section must be bold and smaller than the title of the parent section. To create this effect, a sequence of CSS rules must be created. The `*` operator matches any element, it can be used to match titles having progressive depths in the document.

```
title{
    font-size: 2.4em;
    font-weight:bold;
}
* * title{
    font-size: 2.0em;
}
* * * title{
    font-size: 1.6em;
}
* * * * title{
    font-size: 1.2em;
}
```

It's useful to have before the title a constant text, indicating that it refers to a section. This text can include also the current section number. The `:before` and `:after` pseudo elements will be used, plus the CSS counters.

First declare a counter named `sect` for each `book` or `article`. The counter is set to zero at the beginning of each such element:

```
book,
article{
    counter-reset:sect;
}
```

The `sect` counter is incremented with each `section`, that is a direct child of a `book` or an `article` element.

```
book > section,
article > section{
    counter-increment:sect;
}
```

The "static" text that will prefix the section title is composed of the constant "Section ", followed by the decimal value of the `sect` counter and a dot.

```
book > section > title:before,
article > section > title:before{
    content: "Section " counter(sect) ". ";
}
```

To make the documents easy to read, you add a margin to the sections. In this way the higher nesting level, the larger the left side indent. The margin is expressed relatively to the parent bounds:

```
section{
    margin-left:1em;
    margin-top:1em;
}
```

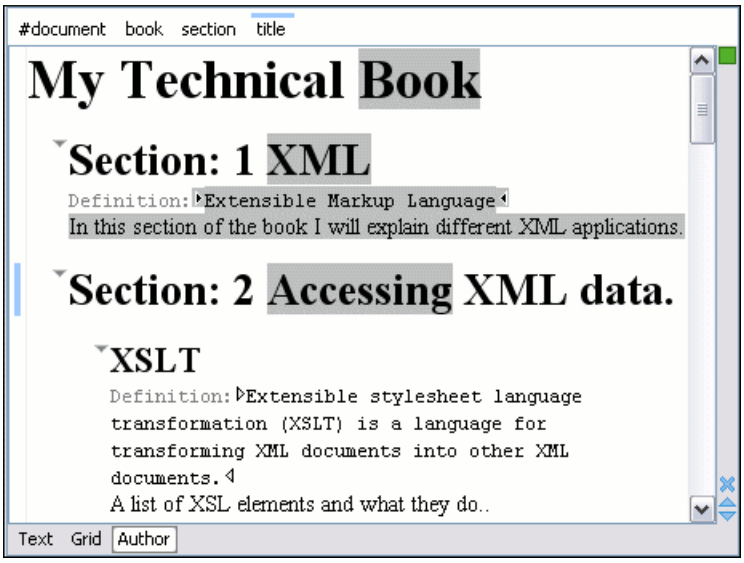


Figure 557: A sample of nested sections and their titles.

In the above screenshot you can see a sample XML document rendered by the CSS stylesheet. The selection "avoids" the text that is generated by the CSS "content" property. This happens because the CSS generated text is not present in the XML document and is just a visual aid.

Styling the Inline Elements

The "bold" style is obtained by using the `font-weight` CSS property with the value `bold`, while the "italic" style is specified by the `font-style` property:

```
b {
    font-weight:bold;
}

i {
    font-style:italic;
}
```

Styling Images

The CSS 2.1 does not specify how an element can be rendered as an image. To overpass this limitation, Oxygen XML Editor supports a CSS Level 3 extension allowing to load image data from a URL. The URL of the image must be specified by one of the element attributes and it is resolved through the catalogs specified in Oxygen XML Editor.

```
image{
    display:block;
    content: attr(href, url);
    margin-left:2em;
}
```

Our `image` element has the required attribute `href` of type `xs:anyURI`. The `href` attribute contains an image location so the rendered content is obtained by using the function:

```
attr(href, url)
```

The first argument is the name of the attribute pointing to the image file. The second argument of the `attr` function specifies the type of the content. If the type has the `url` value, then Oxygen XML Editor identifies the content as being an image. If the type is missing, then the content will be the text representing the attribute value.

Oxygen XML Editor handles both absolute and relative specified URLs. If the image has an *absolute* URL location (for example: "http://www.oasis-open.org/images/standards/oasis_standard.jpg") then it is loaded directly from this location. If the image URL is *relative* specified to the XML document (for example: "images/my_screenshot.jpg") then the location is obtained by adding this value to the location of the edited XML document.

An image can also be referenced by the name of a DTD entity that specifies the location of the image file. For example, if the document declares an entity **graphic** that points to a JPEG image file:

```
<!ENTITY graphic SYSTEM "depo/keyboard_shortcut.jpg" NDATA JPEG>
```

and the image is referenced in the XML document by specifying the name of the entity as the value of an attribute:

```
<mediaobject>
    <imageobject>
        <imagedata entityref="graphic" scale="50"/>
    </imageobject>
</mediaobject>
```

The CSS should use the functions `url`, `attr` and `unparsed-entity-uri` for displaying the image in the **Author** mode:

```
imagedata[entityref]{
    content: url(unparsed-entity-uri(attr(entityref)));
}
```

To take into account the value of the `width` attribute of the `imagedata` and use it for resizing the image, the CSS can define the following rule:

```
imagedata[width]{
  width:attr(width, length);
}
```

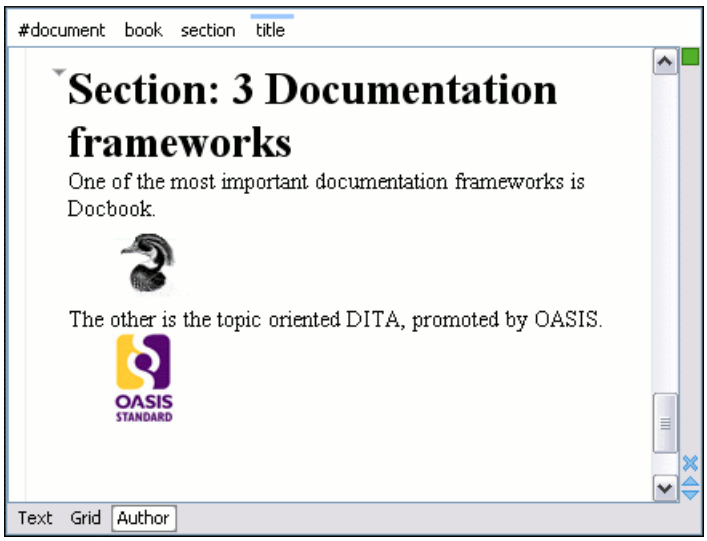


Figure 558: Samples of images in Author

Testing the Framework Customization

To test the new *framework* customization, create an XML instance that conforms with the association rules that you specified in your *framework* customization. You will not specify an XML Schema location directly in the document, using an `xsi:schemaLocation` attribute. Instead, Oxygen XML Editor will detect its associated document type and use the specified schema.

```
<book xmlns="http://www.oxygenxml.com/sample/documentation"
      xmlns:abs="http://www.oxygenxml.com/sample/documentation/abstracts">

    <title>My Technical Book</title>
    <section>
        <title>XML</title>
        <abs:def>Extensible Markup Language</abs:def>
        <para>In this section of the book I will
           explain different XML applications.</para>
    </section>
</book>
```

When trying to validate the document there should be no errors. Now modify the `title` to `title2`. Validate again. This time there should be as error.

```
cvc-complex-type.2.4.a: Invalid content was found starting with element
'title2'. One of '{"http://www.oxygenxml.com/sample/documentation":title}'
is expected.
```

Undo the tag name change, go to **Author** mode, and Oxygen XML Editor should load the CSS from the *document type association* and create a layout similar to this:

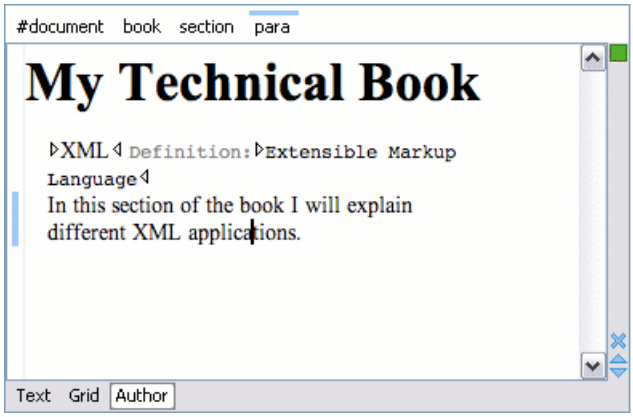


Figure 559: Example: Testing a Framework Customization

Organizing the Framework Files

First, create a new folder for your customized *framework* *[OXYGEN_INSTALL_DIR]*`/frameworks`. This folder will be used to store all files related to the documentation *framework*. The folder structure will look something like this:

```
oxygen
  frameworks
     sdf
       schema
       css
```

The `frameworks` directory is the container where all the Oxygen XML Editor *framework* customizations are located. Each subdirectory contains files related to a specific type of XML documents (schemas, catalogs, stylesheets, CSS stylesheets, etc.) Distributing a *framework* means delivering a *framework* directory.

It is assumed that you have the right to create files and folders inside the Oxygen XML Editor installation directory. If you do not have this right, you will have to install another copy of the program in a folder you have access to, the home directory for instance, or your desktop.

To test your *framework* distribution, copy it in the `frameworks` directory of the newly installed application and start Oxygen XML Editor by running the provided start-up script files.

You should copy the created schema files `abs.xsd` and `sdf.xsd`, `sdf.xsd` being the master schema, to the `schema` directory and the CSS file `sdf.css` to the `css` directory.

Packaging and Deploying

Using a file explorer, go to the Oxygen XML Editor *[OXYGEN_INSTALL_DIR]*`/frameworks` directory. Select the `sdf` directory and make an archive from it. Move it to another Oxygen XML Editor installation (eventually on another computer). Extract it in the *[OXYGEN_INSTALL_DIR]*`/frameworks` directory. Start Oxygen XML Editor and test the association as explained above.

When deploying your customized `sdf` directory, make sure that your `sdf` directory contains the `sdf.framework` file (that is the file defined as External Storage in the ***Document Type Association*** *preferences page* shall always be stored inside the `sdf` directory). If your external storage points somewhere else Oxygen XML Editor will not be able to update the *Document Type Association* options automatically on the deployed computers.

Adding/Editing a Framework (Document Type)

To add or edit a *framework* (document type), *open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Document Type Association**. From this ***Document Type Association** preferences page* you can use the **New**, **Edit**, **Duplicate**, or **Extend** buttons to open a ***Document Type** configuration dialog box* that allows you to customize a new or existing document type (*framework*).

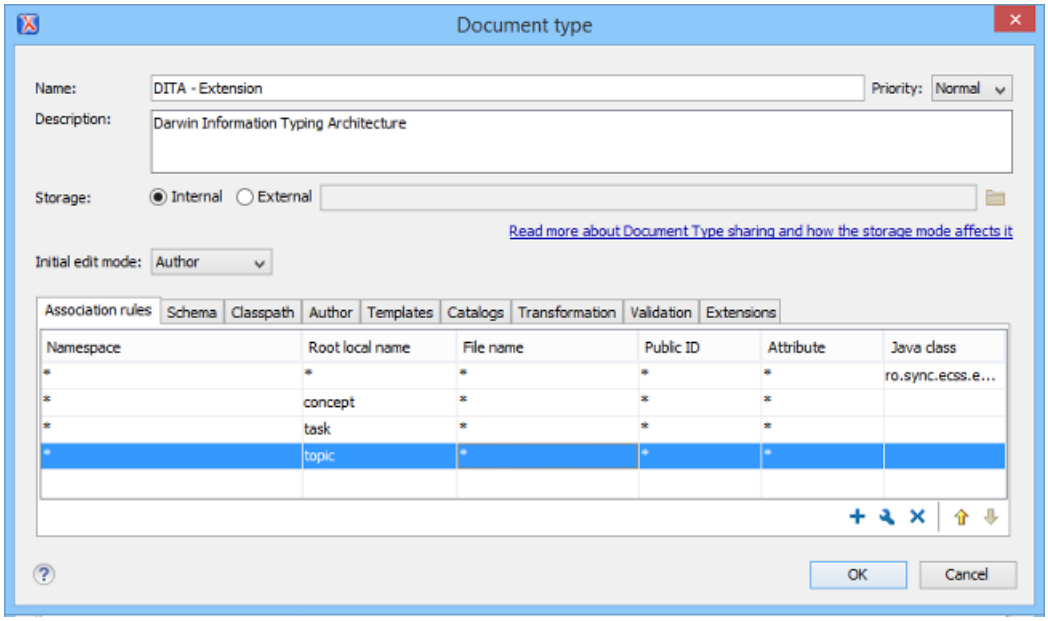


Figure 560: Document Type Configuration Dialog Box

You can specify the following properties for a document type:

- **Name** - The name of the document type.
- **Priority** - When multiple document types match the same document, the priority determines the order in which they are applied. It can be one of the following: Lowest, Low, Normal, High, Highest. The predefined document types that are already configured when the application is installed on the computer have the default Low priority.

  Note: *Frameworks* that have the same priority are sorted alphabetically.
- **Description** - The document type description displayed as a tool tip in the ***Document Type Association** preferences page*.
- **Storage** - The location where the document type is saved. If you select the **External** storage option, the document type is saved in the specified file with a mandatory `framework` extension, located in a subdirectory of the current `frameworks` directory. If you select the **Internal** storage option, the document type data is saved in the Oxygen XML Editor internal options file if *Global Options* is selected or in the current Oxygen XML Editor project file (`.xpr`) if *Project Options* is selected.
- **Initial edit mode** - Allows you to select the initial editing mode for this document type: **Editor specific**, **Text**, **Author, Grid** and **Design** (available only for the W3C XML Schema editor). If the **Editor specific** option is selected, the initial editing mode is determined based upon the editor type. You can find the mapping between editors and edit modes in the ***Edit modes** preferences page.* You can impose an initial mode for opening files that match the association rules of the document type. For example, if the files are usually edited in the **Author** mode you can set it in the **Initial edit mode** combo box.

Note: You can also customize the initial mode for a document type in the **Edit modes** preferences page. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Edit modes**.

You can specify the **Association rules** used for determining a document type for an opened XML document. A rule can define one or more conditions. All conditions need to be fulfilled for a specific rule to be chosen. Conditions can specify:

- **Namespace** - The namespace of the document that matches the document type.
- **Root local name of document** - The local name of the document that matches the document type.

- **File name** - The file name (including the extension) of the document that matches the document type.
- **Public ID** (for DTDs) - The PUBLIC identifier of the document that matches the document type.
- **Attribute** - This field allows you to associate a document type depending on a certain value of the attribute in the root.
- **Java class** - Name of the Java class that is called to determine if the document type should be used for an XML document. Java class must either implement the `ro.sync.ecss.extensions.api.DocumentTypeCustomRuleMatcher` interface or extend the `ro.sync.ecss.extensions.api.DocumentTypeAdvancedCustomRuleMatcher` abstract class from the *Author API*.

In the **Schema** tab, you can specify the type and URI of schema used for validation and content completion of all documents from the document type, when there is no schema detected in the document.

You can choose one of the following schema types:

- DTD
- Relax NG schema (XML syntax)
- Relax NG schema (XML syntax) + Schematron
- Relax NG schema (compact syntax)
- XML Schema
- XML Schema + Schematron rules
- NVDL schema

Related Information:

Configuring and Managing Multiple CSS Styles on page 1256
Customizing the Main CSS of a Framework on page 1237

Configure Actions, Menus, and Toolbars for a Framework

You can configure actions, menus, and toolbars that are specific to a *framework* (document type) in the **Author** mode to gain a productive editing experience, by using the ***Document Type*** *configuration dialog box*.

To add or configure actions, menus, or toolbars follow this procedure:

1. *Open the* ***Preferences*** *dialog box* ***(Options > Preferences)***, go to **Document Types Association**, and select the *framework* for which you want to create an action.
2. Click **Edit** and in the ***Document Type*** *configuration dialog box* go to the **Author** tab, then go to **Actions**.
3. Click the **+ New** button and use the ***Action*** *dialog box* to create an action.

Configure the Insert Section Action for a Framework

This topic describes the procedure for defining the **Insert Section** action for a custom *framework*. It is assumed that the icon files, § (`Section16.gif`) for the menu item and § (`Section20.gif`) for the toolbar, are already available. Although you could use the same icon size for both the menu and toolbar, usually the icons from the toolbars are larger than the ones found in the menus. These files should be placed in your custom *framework* directory (`[OXYGEN_INSTALL_DIR]\frameworks\[CUSTOM_FRAMEWORK_DIR]`).

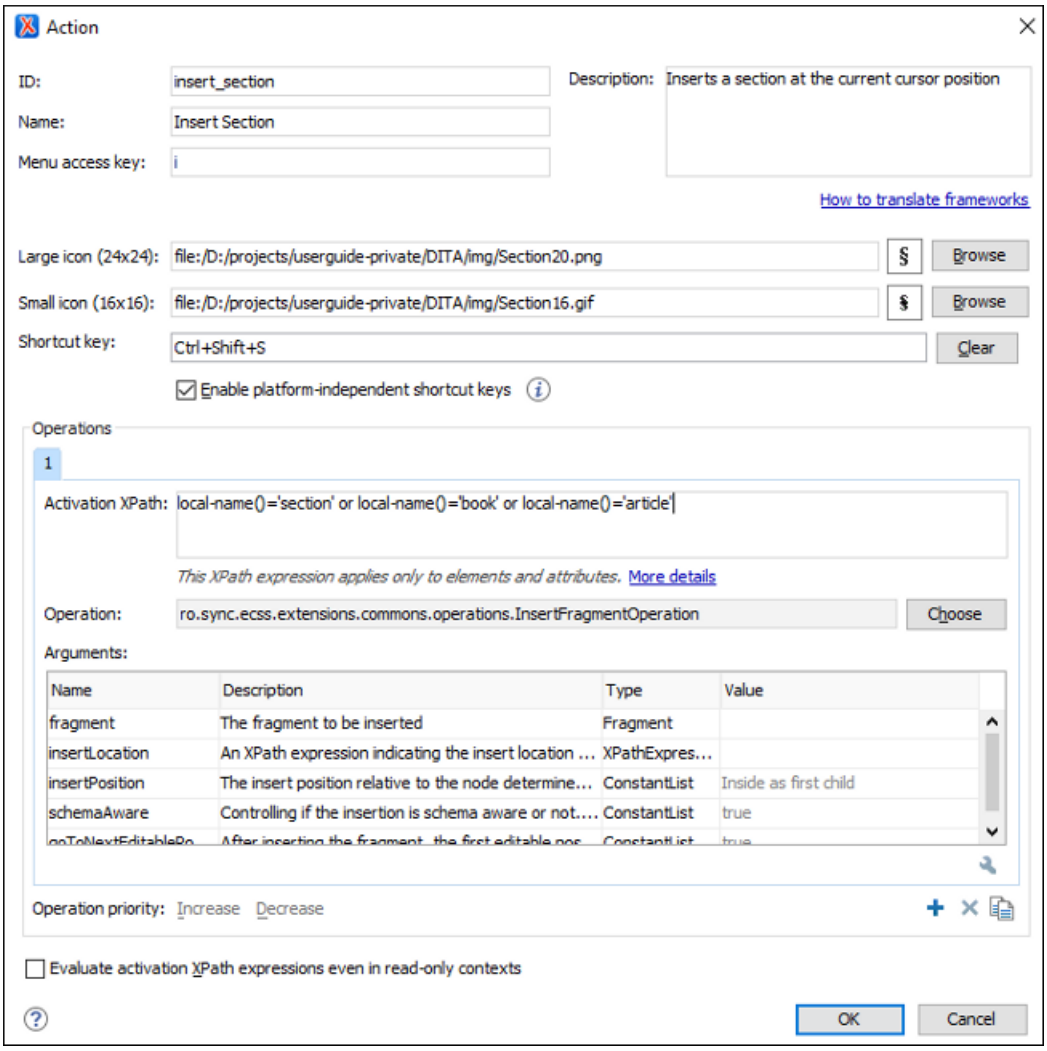


Figure 561: Action Dialog Box

1. Set the **ID** field to **insert_section**. This is a unique action identifier.
2. Set the **Name** field to **Insert Section**. This will be the action's name, displayed as a tooltip when the action is placed in the toolbar, or as the menu item name.
3. Set the **Menu access key** to **i**. On Windows, the menu items can be accessed using **ALT+letter** keys combination, when the menu is visible. The letter is visually represented by underlining the first letter from the menu item name having the same value.
4. Add a **Description**.
5. Set the **Large icon (20x20)** field to `${frameworks}/sdf/Section20.gif`. A good practice is to store the image files inside the *framework* directory and use *editor variable* *${framework}* to make the image relative to the *framework* location.

   If the images are bundled in a *JAR* archive together with some Java operations implementation, for instance, it might be convenient for you to reference the images not by the file name, but by their relative path location in the class-path.

   If the image file `Section20.gif` is located in the **images** directory inside the *JAR* archive, you can reference it by using **/images/Section20.gif**. The *JAR* file must be added into the **Classpath** list.
6. Set the **Small icon (16x16)** field to `${frameworks}/sdf/Section16.gif`.
7. Click the text field next to **Shortcut key** and set it to **Ctrl+Shift+S (Meta+Shift+S on Mac OS)**. This will be the key combination to trigger the action using the keyboard only.

   The shortcut is enabled only by *adding the action to the main menu of **Author** mode*, which contains all the actions that the author will have in a menu for the current document type.

8. At this time the action has no functionality added to it. Next you must define how this action operates. An action can have multiple operation modes. The first action mode enabled by the evaluation of its associated XPath expression will be executed when the action is triggered by the user. The Xpath expression needs to be version 2.0 and its scope must be only element and attribute nodes of the edited document. Otherwise, the expression will not return a match and will not trigger the action. If the expression is left empty, the action will be enabled anywhere in the scope of the root element. For this example we'll suppose you want allow the action to add a section only if the current element is either a `book`, `article` or another `section`.
   a) Set the XPath expression field to:

   ```
   local-name()='section' or local-name()='book' or
    local-name()='article'
   ```

   b) Set the **invoke operation** field to `InsertFragmentOperation` built-in operation, designed to insert an XML fragment at cursor position. This belongs to a set of built-in operations, a complete list of which can be found in the *Author Default Operations* section. This set can be expanded with your own Java operation implementations.
   c) Configure the arguments section as follows:

   ```
   <section xmlns=
   "http://www.oxygenxml.com/sample/documentation">
        <title/>
   </section>
   ```

   `insertLocation` - leave it empty. This means the location will be at the cursor position.

   `insertPosition` - select "**Inside**".

Configure the Insert Table Action for a Framework

This topic describes the procedure for defining the **Insert Table** action for a custom *framework*. Suppose that you want to create an action that inserts a table with three rows and three columns into a document and the first row is the table header. As with *the insert section action*, you will use the `InsertFragmentOperation` built-in operation.

Place the icon files for the menu item, and for the toolbar, in your custom *framework* directory (*`[OXYGEN_INSTALL_DIR]`*`\frameworks\[CUSTOM_FRAMEWORK_DIR]`).

1. Set **ID** field to **insert_table**.
2. Set **Name** field to **Insert table**.
3. Set **Menu access key** field to **t**.
4. Set **Description** field to **Adds a table element**.
5. Set **Toolbar icon** to **${framework} / toolbarIcon.png**.
6. Set **Menu icon** to **${framework} / menuIcon.png**.
7. Set **Shortcut key** to **Ctrl + Shift + T (Command + Shift + T on OS X)**.
8. Set up the action's functionality:
   a) Set **XPath expression** field to `true()`.

   `true()` is equivalent with leaving this field empty.
   b) Set **Invoke operation** to use **InsertFragmentOperation** built-in operation that inserts an XML fragment to the cursor position.
   c) Configure operation's arguments as follows:

   fragment - set it to:

   ```
   <table xmlns=
   "http://www.oxygenxml.com/sample/documentation">
     <header><td/><td/><td/></header>
     <tr><td/><td/><td/></tr>
     <tr><td/><td/><td/></tr>
   </table>
   ```

   `insertLocation` - to add tables at the end of the section use the following code:

   ```
   ancestor::section/*[last()]
   ```

`insertPosition` - Select **After**.

Configure the Main Menu for a Framework

Defined actions can be grouped into customized menus in the Oxygen XML Editor menu bar.

1. Open the ***Document Type*** *configuration dialog box*, select your custom *framework*, and go to the **Author** tab.
2. Go to the **Menu** subtab. In the left side you have the list of actions and some special entries:
   - **Submenu** - Creates a submenu. You can nest an unlimited number of menus.
   - **Separator** - Creates a separator into a menu. This way you can logically separate the menu entries.
3. The right side of the panel displays the current actions for that menu tree. To change its name, click this label to select it, then press the **Edit** button.
4. Select the **Submenu** label in the left panel section and the appropriate label in the right panel section, then press the **Add as child** button. Change the submenu name to **Table**, using the **Edit** button.
5. Select the **Insert section** action in the left panel section and the **Table** label in the right panel section, then press the **Add as sibling** button.
6. Now select the **Insert table** action in the left panel section and the **Table** in the right panel section. Press the **Add as child** button.

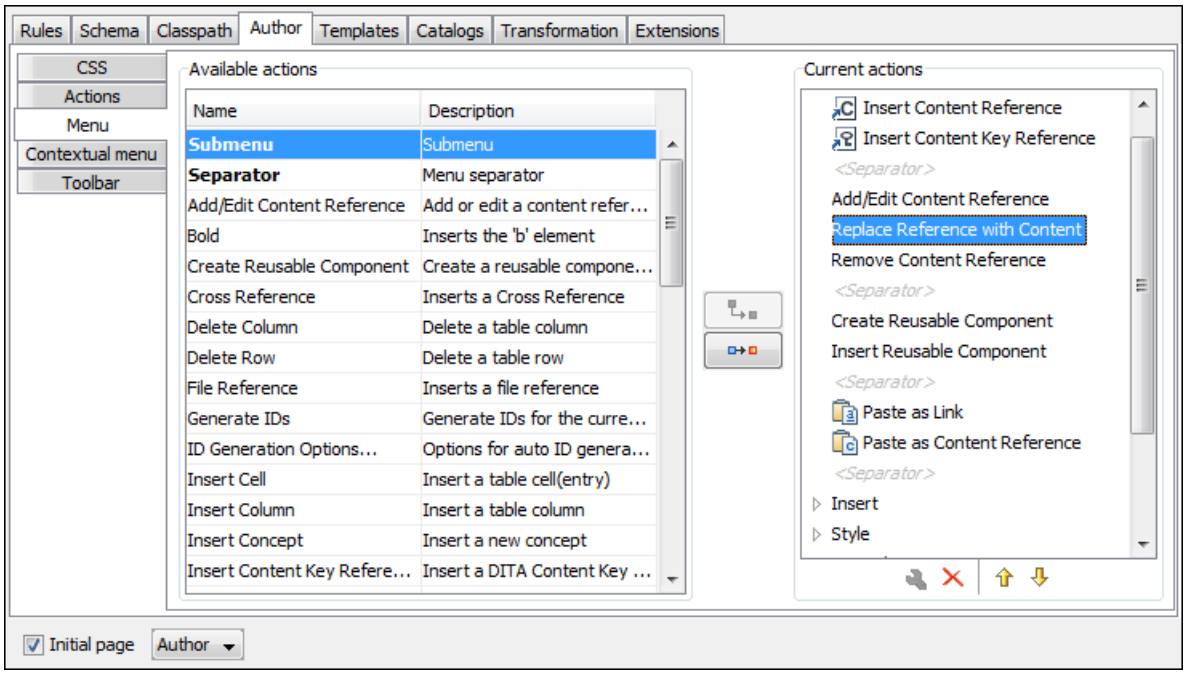


Figure 562: Configuring the Menu

When opening a test document for a custom *framework* in **Author** mode, the menu you created is displayed in the editor menu bar, between the **Tools** and the **Document** menus. The upper part of the menu contains generic **Author** mode actions (common to all document types) and the two actions created previously (with **Insert table** under the **Table** submenu).

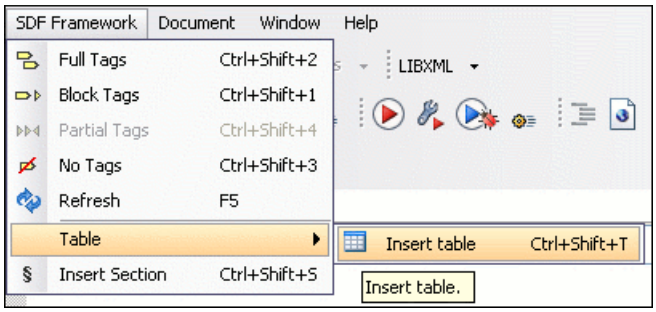


Figure 563: Author Mode Menu

Configure the Contextual Menu for a Framework

The contextual menu is displayed when you right-click in the **Author** editing area. You can only configure the bottom part of the menu, since the top part is reserved for a list of generic actions (such as Copy, Paste, Undo, etc.)

1. Open the ***Document Type*** *configuration dialog box* for the particular *framework* and go to the **Author** tab. Next, go to the **Contextual Menu** subtab.
2. Follow the same steps as explained in the *Configuring the Main Menu*, except changing the menu name because the contextual menu does not have a name.

   Note: You can choose to reuse a submenu that contains general authoring actions. In this case, all actions (both general and *framework*-specific ones) are grouped together under the same submenu.

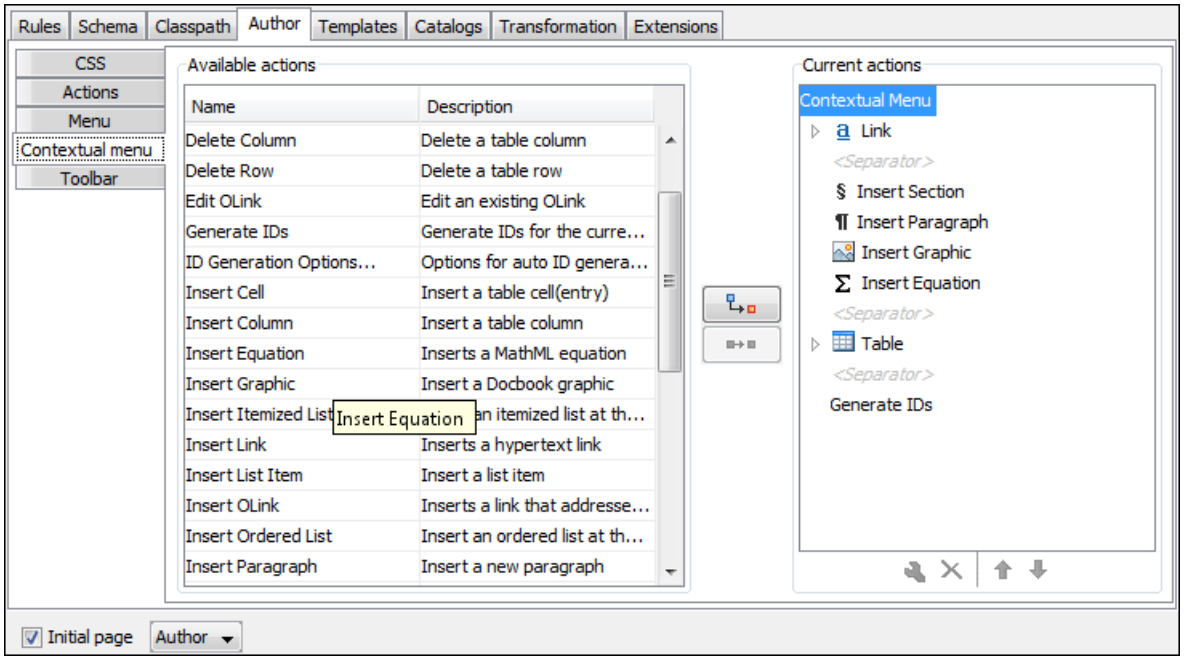


Figure 564: Configuring the Contextual Menu

To test it, open the test file, and open the contextual menu. In the lower part there is shown the **Table** sub-menu and the **Insert section** action.

Configure the Toolbars for a Framework

The procedure below describes how to add defined actions to a toolbar for a custom *framework*. These steps use examples from the two previous help topics that described how to define the ***Insert Section*** and ***Insert Table*** actions. You can also configure additional toolbars to add other custom actions.

1. Open the ***Document Type*** *configuration dialog box* for your custom *framework* and select the **Author** tab. Next, go to the **Toolbar** subtab.

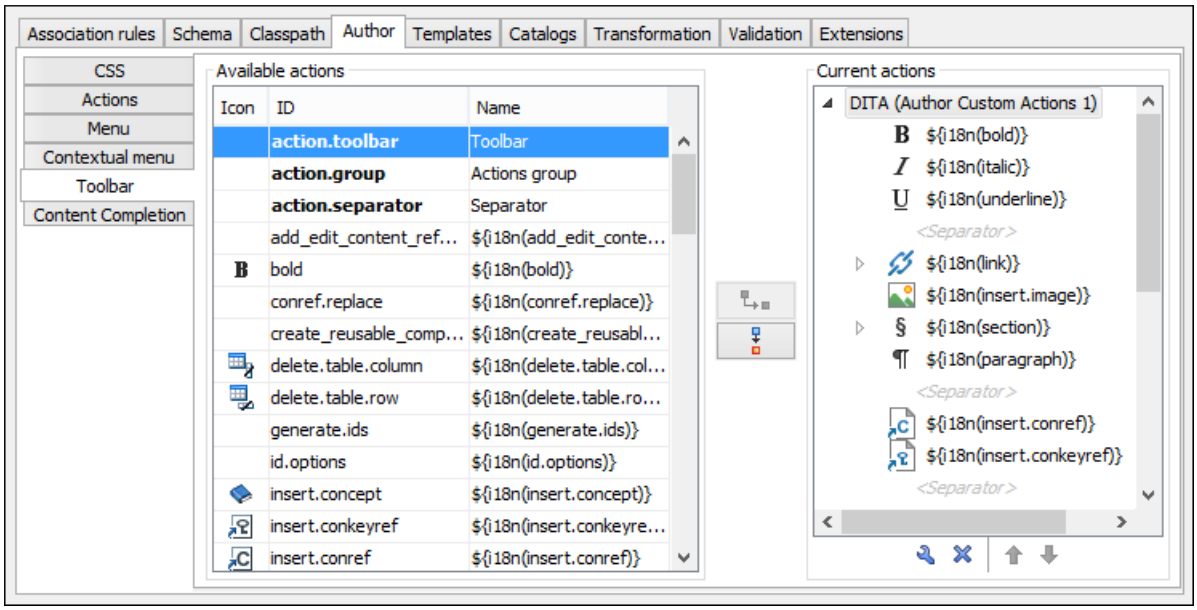


Figure 565: Configuring the Toolbar

The panel is divided in two sections: the left side contains a list of actions, while the right one contains an action tree, displaying the list of actions added in the toolbar. The special entry called *Separator* allows you to visually separate the actions in the toolbar.

2. Select the **Insert section** action in the left panel section and the **Toolbar** label in the right panel section, then press the **Add as child** button.
3. Select the **Insert table** action in the left panel section and the **Insert section** in the right panel section. Press the **Add as sibling** button.
   When opening a document of the particular *framework* in **Author** mode, the toolbar with the new buttons will be displayed in the toolbar area.

   Tip: If you have many custom toolbar actions, or want to group actions according to their category, add more toolbars with custom names and split the actions to better suit your purpose. If your toolbar is not displayed when switching to the **Author** mode, right-click the main toolbar, select **Configure Toolbars**, and make sure the appropriate toolbar (such as the **Author Custom Actions** toolbar) is selected.

Configure Content Completion for a Framework

You can customize the content of the following **Author** controls, adding items (which, when invoked, perform custom actions) or filtering the default contributed ones:

- *Content Completion Assistant* window
- ***Elements** view*
- **Insert Element** menus (from the ***Outline** view* or *breadcrumb* contextual menus)

You can use the content completion customization support in a custom *framework* by following this procedure:

1. Open the ***Document type** configuration dialog box* for your custom *framework* and select the **Author** tab. Next, go to the **Content Completion** tab.

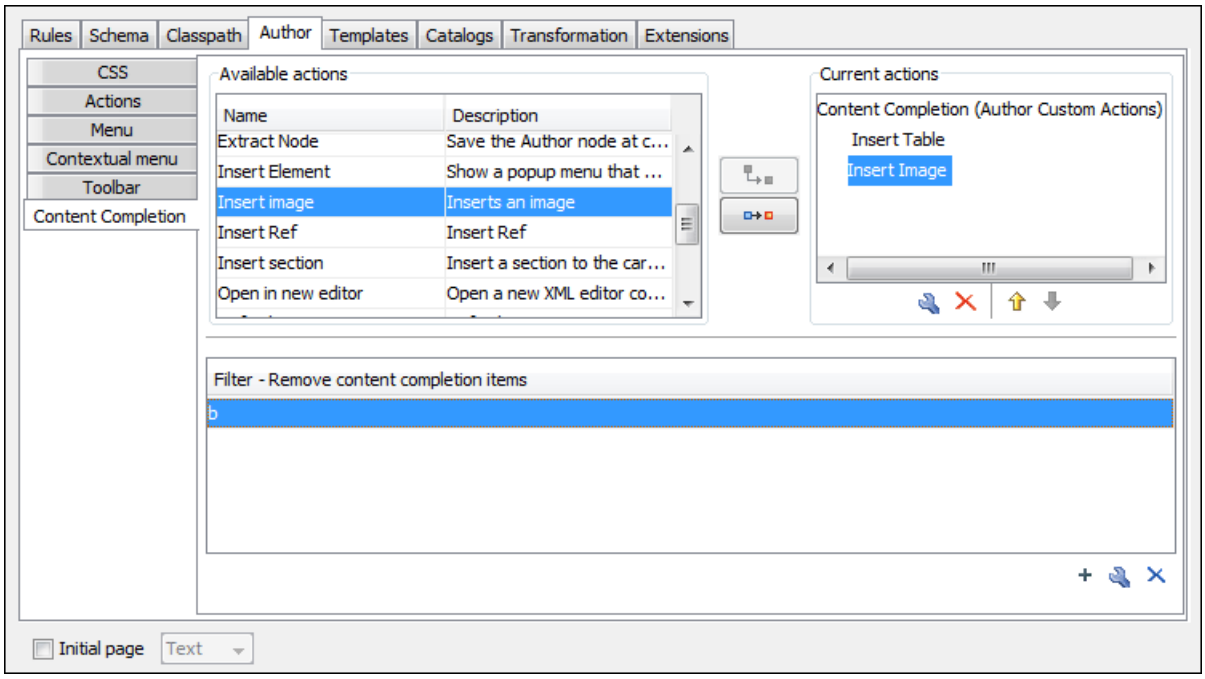


Figure 566: Customize Content Completion

The top side of the **Content Completion** section contains the list with all the actions defined within the custom *framework* and the list of actions that you decided to include in the *Content Completion Assistant* list of proposals. The bottom side contains the list with all the items that you decided to remove from the *Content Completion Assistant* list of proposals.

2. If you want to add a custom action to the list of current **Content Completion** proposals, select the action item from the **Available actions** list and press the **Add as child** or **Add as sibling** button to include it in the **Current actions** list. An **Insert Action** dialog box appears, giving you the possibility to select where to provide the selected action.

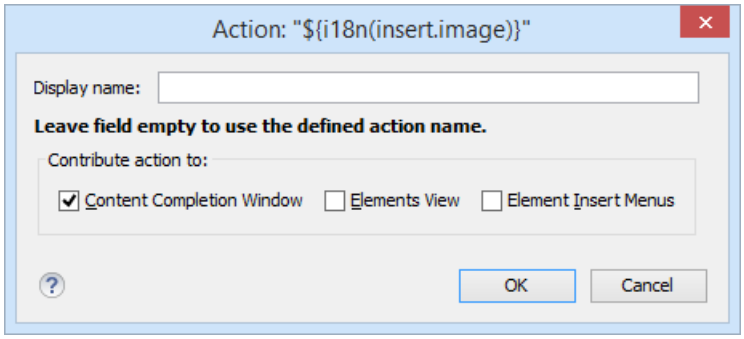


Figure 567: Insert Action Dialog Box

3. If you want to exclude a certain item from the **Content Completion** proposals, you can use the **Add** button from the **Filter - Remove content completion items** list. The **Remove item** dialog box is displayed, allowing you to input the item name and to choose the controls that filter it. The **Item name** combo box accepts wildcards.

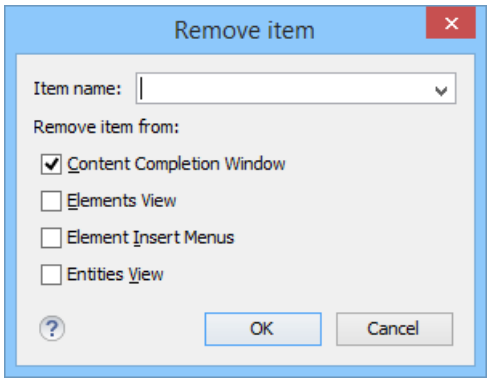


Figure 568: Remove Item Dialog Box

Author Mode Default Operations

The default operations for the **Author** mode, along with their arguments are as follows:

- **`ChangeAttributeOperation`**

  This operation allows you to add/modify/remove an attribute. You can use this operation in your own custom **Author** mode action to modify the value for a certain attribute on a specific XML element. The arguments of the operation are:

  - **`name`** - The attribute local name.
  - **`namespace`** - The attribute namespace.
  - **`elementLocation`** - The XPath location that identifies the element.
  - **`value`** - The new value for the attribute. If empty or null the attribute will be removed.
  - **`editAttribute`** - If an in-place editor exists for this attribute, it will automatically activate the in-place editor and start editing.
  - **`removeIfEmpty`** - The possible values are `true` and `false`. True means that the attribute should be removed if an empty value is provided. The default behavior is to remove it.
- **`ChangePseudoClassesOperation`**

  Operation that sets a list of pseudo-class values to nodes identified by an XPath expression. It can also remove a list of values from nodes identified by an XPath expression. The operation accepts the following parameters:

  - **`setLocations`** - An XPath expression indicating a list of nodes for which the specified list of pseudo-classes will be set. If it is not defined, then the element at the cursor position will be used.
  - **`setPseudoClassNames`** - A space-separated list of pseudo-class names that will be set on the matched nodes.
  - **`removeLocations`** - An XPath expression indicating a list of nodes from which the specified list of pseudo-classes will be removed. If it is not defined, then the element at the cursor position will be used.
  - **`removePseudoClassNames`** - A space-separated list of pseudo-class names that will be removed from the matched nodes.
  - **`includeAllNodes`** - The possible values are `yes` and no. If set to `yes`, comments, CDATA, and text nodes are included when evaluating XPath expressions. If set to no, they are ignored.
- **`DeleteElementOperation`**

  Deletes the node indicated by the `elementLocation` parameter XPath expression. If missing, the operation will delete the node at the cursor location.
- **`DeleteElementsOperation`**

  Deletes the nodes indicated by the `elementLocations` parameter XPath expression. If missing, the operation will delete the node at the cursor location.
- **`ExecuteCommandLineOperation`**

  This operation allows you to start a process executing a given command line. It has the following arguments:

- `name` - The name of the operation (or name of the console panel that corresponds to the process run by an action built over this operation).
- `workingDirectory` - The path to the directory where the command line is executed. The default value is `"."` (current directory).
- `cmdLine` - The command line to be executed (accepts *editor variables*).
- `showConsole` - If set to `true`, the console panel will be displayed in Oxygen XML Editor. The default value is `false`.

- **`ExecuteMultipleActionsOperation`**

  This operation allows the execution of a sequence of actions, defined as a list of action IDs. The actions must be defined by the corresponding *framework*, or one of the common actions for all *frameworks* supplied by Oxygen XML Editor.

  - **`actionIDs`** - The action IDs list that will be executed in sequence, the list must be a string sequence containing the IDs separated by commas or new lines.
- **`ExecuteMultipleWebappCompatibleActionsOperation`**

  An implementation of an operation that runs a sequence of Oxygen XML Web Author Component-compatible actions, defined as a list of IDs.
- **`ExecuteTransformationScenariosOperation`**

  This operation allows running one or more transformation scenarios defined in the current *document type association*. It is useful to add to the toolbar buttons that trigger publishing to various output formats. The argument of the operation is:

  - **`scenarioNames`** - The list of scenario names that will be executed, separated by new lines.
- **`InsertEquationOperation`**

  Inserts a fragment containing a MathML equation at the cursor offset. The argument of this operation is:

  - **`fragment`** - The XML fragment containing the MathML content that should be inserted.
- **`InsertFragmentOperation`**

  Inserts an XML fragment at the current cursor position. The selection - if there is one, remains unchanged. The fragment will be inserted in the current context of the cursor position meaning that if the current XML document uses some namespace declarations then the inserted fragment must use the same declarations. The namespace declarations of the inserted fragment will be adapted to the existing namespace declarations of the XML document. For more details about its list of parameters, see *Arguments of InsertFragmentOperation Operation* on page 1193.
- **`InsertOrReplaceFragmentOperation`**

  Similar to **`InsertFragmentOperation`**, except it removes the selected content before inserting the fragment. For more details about its list of parameters, see *Arguments of InsertFragmentOperation Operation*.
- **`InsertOrReplaceTextOperation`**

  Inserts a text at current position removing the selected content, if any. The argument of this operation is:

  - **`text`** - The text section to insert.
- **`InsertXIncludeOperation`**

  Insert an **`XInclude`** element at the cursor offset. Opens a dialog box that allows you to browse and select content to be included in your document and automatically generates the corresponding XInclude instruction.
- **`JSOperation`**

  Allows you to call the Java API from custom JavaScript content. For some sample `JSOperation` implementations, see *https://github.com/oxygenxml/javascript-sample-operations*.

  Notice: For the Oxygen XML Web Author Component, this operation cannot be invoked using the JavaScript API.

  This operation accepts the following parameter:

  - **`script`**

The JavaScript content to execute. It must have a function called `doOperation()`, which can use the predefined `authorAccess` variable. The `authorAccess` variable has access to the entire *`ro.sync.ecss.extensions.api.AuthorAccess`* Java API.

The following example is a script that retrieves the current value of the **type** attribute on the current element, allows the end user to edit its new value and sets the new value in the document:

```
function doOperation(){
 //The current node is either entirely selected...
 currentNode = authorAccess.getEditorAccess().getFullySelectedNode();
 if(currentNode == null){
 //or the cursor is placed in it
 caretOffset = authorAccess.getEditorAccess().getCaretOffset();
 currentNode = authorAccess.getDocumentController().getNodeAtOffset
                         (caretOffset);
 }
 //Get current value of the @type attribute
  currentTypeValue = "";
  currentTypeValueAttr = currentNode.getAttribute("type");
  if(currentTypeValueAttr != null){
    currentTypeValue = currentTypeValueAttr.getValue();
  }
 //Ask user for new value for attribute.
 newTypeValue = javax.swing.JOptionPane.showInputDialog
                        ("Input @type value", currentTypeValue);
 if(newTypeValue != null){
   //Create and set the new attribute value for the @type attribute.
   attrValue = new Packages.ro.sync.ecss.extensions.api.node.AttrValue
                        (newTypeValue);
   authorAccess.getDocumentController().setAttribute
                        ("type", attrValue, currentNode);
 }
}
```

Tip: You can call functions defined inside a script called `commons.js` from your custom script content so that you can use that external script file as a library of functions. Note that this `commons.js` file must be placed in the root of the *framework* directory (for example, *`[OXYGEN_INSTALL_DIR]`*`/frameworks/dita/commons.js`) because that is the only location where Oxygen XML Editor will look for it.

- **`MoveCaretOperation`**

  Flexible operation for moving the cursor within a document and it is also capable of performing a selection. The operation accepts the following arguments:

  - **`xpathLocation`** - An XPath expression that identifies the node relative to where the cursor will be moved. If the expression identifies more than one node, only the first one will be taken into account.
  - **`position`** - The position relative to the node obtained from the XPath expression where the cursor will be moved. When also choosing to perform a selection, you can use the following possible values:
    - `Before` - Places the cursor at the beginning of the selection.
    - `Inside, at the beginning` - Places the cursor at the beginning of the selection.
    - `After` - Places the cursor at the end of the selection.
    - `Inside, at the end` - Places the cursor at the end of the selection.
  - **`selection`** - Specifies if the operation should select the element obtained from the XPath expression, its content, or nothing at all. The possible values of the argument are: `None`, `Element`, and `Content`.
- **`MoveElementOperation`**

  Flexible operation for moving an XML element to another location from the same document. XPath expressions are used to identify the source element and the target location. The operation takes the following parameters:

  - **`sourceLocation`** - XPath expression that identifies the content to be moved.
  - **`deleteLocation`** - XPath expression that identifies the node to be removed. This parameter is optional. If missing, the **`sourceLocation`** parameter will also identify the node to be deleted.
  - **`surroundFragment`** - A string representation of an XML fragment. The moved node will be wrapped in this string before moving it in the destination.
  - **`targetLocation`** - XPath expression that identifies the location where the node must be moved to.
  - **`insertPosition`** - Argument that indicates the insert position.
  - **`moveOnlySourceContentNodes`** - When `true`, only the content of the source element is moved.

- **processTrackedChangesForXpathLocations** - When nodes are located via an XPath expression and the nodes are deleted with *Change Tracking* enabled, they are considered as being present by default (thus, the *change tracking* is ignored). If you set this argument to `true` and *change tracking* is enabled, deleted nodes will be ignored when the XPath locations are computed (thus, the *change tracking* is NOT ignored).

- **OpenInSystemAppOperation**

  Opens a resource in the system application that is associated with the resource in the operating system. The arguments of this operation is:

  - **resourcePath** - An XPath expression that, when executed, returns the path of the resource to be opened. *Editor variables* are expanded in the value of this parameter, before the expression is executed.
  - **isUnparsedEntity** - Possible values are `true` or `false`. If the value is `true`, the value of the **resourcePath** argument is treated as the name of an unparsed entity.

- **RemovePseudoClassOperation**

  An operation that removes a pseudo-class from an element. Accepts the following parameters:

  - **name** - Name of the pseudo-class to be removed.
  - **includeAllNodes** - The possible values are `yes` and no. If set to `yes`, comments, CDATA, and text nodes are included when evaluating XPath expressions. If set to no, they are ignored.
  - **elementLocation** - The XPath location that identifies the element. Leave it empty for the current element. It can also identify multiple elements, in which case the pseudo class will be removed from all of them.

    Example:

    Suppose that there is a pseudo-class called `myClass` on the element `paragraph` and there are CSS styles matching the pseudo-class.

    ```
    paragraph:myClass{
      font-size:2em;
      color:red;
    }
    paragraph{
      color:blue;
    }
    ```

    In the previous example, by removing the pseudo-class, the layout of the `paragraph` is rebuilt by matching the other rules (in this case, the foreground color of the `paragraph` element will become blue.

- **RenameElementOperation**

  This operation allows you to rename all occurrences of the elements identified by an XPath expression. The operation requires two parameters:

  - **elementName** - The new element name
  - **elementLocation** - The XPath expression that identifies the element occurrences to be renamed. If this parameter is missing, the operation renames the element at current cursor position.

- **SetPseudoClassOperation**

  An operation that sets a pseudo-class to an element. The operation accepts the following parameters:

  - **elementLocation** - An XPath expression indicating the element for which the pseudo-class will be set. If it is not defined, then the element at cursor position will be used.
  - **name** - The pseudo-class local name.
  - **includeAllNodes** - The possible values are `yes` and no. If set to `yes`, comments, CDATA, and text nodes are included when evaluating XPath expressions. If set to no, they are ignored.

- **ShowElementDocumentationOperation**

  Opens the associated specification HTML page for the current element. The operation accepts as parameter a URL pattern that points to the HTML page containing the documentation.

- **SurroundWithFragmentOperation**

  Surrounds the selected content with a text fragment. Since the fragment can have multiple nodes, the surrounded content will be always placed in the first leaf element. If there is no selection, the operation

will simply insert the fragment at the cursor position. For more details about the list of parameters go to *Arguments of SurroundWithFragmentOperation* on page 1195.

- **`SurroundWithTextOperation`**

  This operation has two arguments (two text values) that will be inserted before and after the selected content. If there is no selected content, the two sections will be inserted at the cursor position. The arguments of the operation are:

  - **`header`** - The text that is placed before the selection.
  - **`footer`** - The text that is placed after the selection.
- **`TogglePseudoClassOperation`**

  An implementation of an operation to toggle on/off the pseudo-class of an element. Accepts the following parameters:

  - **`name`** - Name of the pseudo-class to be toggled on/off.
  - **`includeAllNodes`** - The possible values are `yes` and no. If set to `yes`, comments, CDATA, and text nodes are included when evaluating XPath expressions. If set to no, they are ignored.
  - **`elementLocation`** - The XPath location that identifies one or more elements for which the pseudo class will be toggled. Leave it empty for the current element.

    Example:

    ```
    paragraph:myClass{
      color:red;
    }
    paragraph{
      color:blue;
    }
    ```

    By default, the paragraph content is rendered in blue. Suppose that we have a `TogglePseudoClassOperation` configured for the `myClass` pseudo-class. Invoking it the first time will set the `myClass` pseudo-class and the paragraph will be rendered in red. Invoking the operation again, will remove the pseudo-class and the visible result will be a blue rendered `paragraph` element.
- **`ToggleSurroundWithElementOperation`**

  This operation allows wrapping and unwrapping content in a specific wrapper element that can have certain attributes specified on it. It is useful to implement toggle actions such as highlighting text as bold, italic, or underline. The operation supports processing multiple selection intervals, such as multiple cells within a table column selection. The arguments of the operation are:

  - **`element`** - The element to wrap or unwrap content.
  - **`schemaAware`** - This argument applies only on the surround with element operation and controls whether or not the insertion is valid, based upon the schema. If the insertion is not valid, then wrapping action will be broken up into smaller intervals until the wrapping action is valid. For example, if you try to wrap a *paragraph* element with a *bold* element, it would not be valid, so the operation will wrap the text inside the paragraph instead, since it would be valid at that position.
- **`UnwrapTagsOperation`**

  This operation allows removing the element tags either from the current element or for an element identified with an XPath location. The argument of the operation is

  - **`unwrapElementLocation`** - An XPath expression indicating the element to unwrap. If it is not defined, the element at the cursor position is unwrapped.
- **`XQueryUpdateOperation`**

  Allows you to execute an XQuery Update script directly over content in **Author** mode.

  Notice: This operation is not applicable to the Oxygen XML Author Component or the Oxygen XML Web Author Component.

  It has one argument:

  - **`script`**

The XQuery Update script to be executed. The value can either be an XQuery script or a URL that points to the XQuery Update script. You can use the *${framework}* or *${frameworkDir}* editor variables to refer the scripts from the *framework* directory.

The script will be executed in the context of the node at the cursor position. If the script declares the following variable, it will also receive the selected nodes (assuming that entire nodes are selected):

```
declare variable $oxyxq:selection external;
```

An example of an XQuery Update Script that converts paragraphs to list items:

```
declare namespace oxyxq = "http://www.oxygenxml.com/ns/xqu";
(: This variable will be linked to the selected nodes assuming that there are
actually fully selected nodes. For example this selection will return null:
<p>{SEL_START}text{SEL_END} in para</p>
but this will give two "p" elements:
{SEL_END}<p>text</p><p>text2</p>{SEL_END}

If a multiple selection exists it will also be processed and forwarded.
 Again, only fully selected nodes will be passed.
:)
declare variable $oxyxq:selection external;

(: We will process either the selection or the context node :)
let $toProcess := if (empty($oxyxq:selection)) then
    (.)
else
    ($oxyxq:selection)

returnif (not(empty($toProcess))) then
        (
        (: Create the list :)
        let $ul :=
        <ul>
            {
                for $sel in $toProcess
                return
                    <li>{$sel}</li>
            }
        </ul>

        return
            (
            (: Delete the processed nodes :)
            for $sel in $toProcess
            return
                delete node $sel,
            (: Inserts the constructed list :)
            insert node $ul
                before $toProcess[1]
            )
        )
    else
        ()
```

- **`XSLTOperation`** and **`XQueryOperation`**

  Applies an XSLT or XQuery script on a source element and then replaces or inserts the result in a specified target element.

  Notice: For the Oxygen XML Web Author Component, these operations cannot be invoked using the *JavaScript API*.

  These operations have the following parameters:

  - **`sourceLocation`**

    An XPath expression indicating the element that the script will be applied on. If it is not defined, then the element at the cursor position will be used.

    There may be situations in which you want to look at an ancestor of the current element and take decisions in the script based on this. To do this, you can set the `sourceLocation` to point to an ancestor node then declare a parameter called `currentElementLocation` in your script and use it to re-position in the current element, as in the following example:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0"
  xpath-default-namespace="http://docbook.org/ns/docbook"
  xmlns:saxon="http://saxon.sf.net/" exclude-result-prefixes="saxon">
    <!-- This is an XPath location to be sent by the operation to the script -->
    <xsl:param name="currentElementLocation"/>

    <xsl:template match="/">
```

```
        <!-- Evaluate the XPath of the current element location -->
        <xsl:apply-templates
        select="saxon:eval(saxon:expression($currentElementLocation))"/>
    </xsl:template>

    <xsl:template match="para">
        <!-- And the context is again inside the current element,
        but we can use information from the entire XML -->
        <xsl:variable
            name="keyImage" select="//imagedata[@fileref='images/lake.jpeg']
              /ancestor::inlinemediaobject/@xml:id/string()"/>
                <xref linkend="{$keyImage}" role="key_include"
                  xmlns="http://docbook.org/ns/docbook">
                    <xsl:value-of
                          select="$currentElementLocation"></xsl:value-of>
        </xref>
    </xsl:template>
</xsl:stylesheet>
```

- **`targetLocation`**

  An XPath expression indicating the insert location for the result of the transformation. If it is not defined then the insert location will be at the cursor location.
- **`script`**

  The script content (XSLT or XQuery). The base system ID for this will be the *framework* file, so any include/import reference will be resolved relative to the `.framework` file that contains this action definition.

  For example, for the following script, the imported `xslt_operation.xsl` needs to be located in the current *framework* directory.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
            version="1.0">
    <xsl:import href="xslt_operation.xsl"/>
</xsl:stylesheet>
```

  You can also use a path for an included or imported reference. When using a path, the following apply:

  - A relative path is resolved to the *framework* directory.
  - The *`${framework}` editor variable* can also be used to reference resources from the *framework* directory.
  - The path is passed through the catalog mappings. It helps to use an absolute URL (for instance, `http://www.oxygenxml.com/fr/testy.xsl`) and map it in the `catalog.xml` file from the *framework* directory to a resource from the *framework*.
- **`action`**

  The insert action relative to the node determined by the target XPath expression. It can be: Replace, At cursor position, Before, After, Inside as first child or Inside as last child.
- **`caretPosition`**

  The position of the cursor after the action is executed. It can be: `Preserve`, `Before`, `Start`, `First editable position`, End, or `After`. If this parameter is not set, you can still indicate the position of the cursor by using the *`${caret}` editor variable* in the inserted content.
- **`expandEditorVariables`**

  Parameter controlling the expansion of *editor variables* returned by the script processing. Expansion is enabled by default.

Editor Variables in Author Mode Operations

Author mode operations can include parameters that contain the following *editor variables*:

- *${caret}* - The position where the cursor is located. This variable can be used in a code template, in **Author** mode operations, or in a **selection** *plugin*.

Note: The **${caret}** editor variable is available only for parameters that take XML content as values. It is replaced with the **${UNIQUE_CARET_MARKER_FOR_AUTHOR}** macro. The default Author operations process this macro and position the cursor at the designated offset.

- *${selection}* - The current selected text content in the current edited document. This variable can be used in a code template, in **Author** mode operations, or in a **selection** *plugin*.

- *${ask('message', type, ('real_value1':'rendered_value1'; 'real_value2':'rendered_value2'; ...), 'default_value')}* - To prompt for values at runtime, use the *ask('message', type, ('real_value1':'rendered_value1'; 'real_value2':'rendered_value2'; ...), 'default-value")* editor variable. You can set the following parameters:
  - *'message'* - The displayed message. Note the quotes that enclose the message.
  - *type* - Optional parameter, with one of the following values:

    Note: The title of the dialog box will be determined by the type of parameter and as follows:

    - For *url* and *relative_url* parameters, the title will be the name of the parameter and the value of the *'message'*.
    - For the other parameters listed below, the title will be the name of that respective parameter.
    - If no parameter is used, the title will be "Input".

Parameter	
url	**Format:** *${ask('message', url, 'default_value')}*
	Description: Input is considered a URL. Oxygen XML Editor checks that the provided URL is valid.
	Example: • *${ask('Input URL', url)}* - The displayed dialog box has the name *Input URL*. The expected input type is URL. • *${ask('Input URL', url, 'http://www.example.com')}* - The displayed dialog box has the name *Input URL*. The expected input type is URL. The input field displays the default value *http://www.example.com*.
password	**Format**: *${ask('message', password, 'default')}*
	Description: The input is hidden with bullet characters.
	Example: • *${ask('Input password', password)}* - The displayed dialog box has the name *'Input password'* and the input is hidden with bullet symbols. • *${ask('Input password', password, 'abcd')}* - The displayed dialog box has the name *'Input password'* and the input hidden with bullet symbols. The input field already contains the default `abcd` value.
generic	**Format**: *${ask('message', generic, 'default')}*
	Description: The input is considered to be generic text that requires no special handling.
	Example: • *${ask('Hello world!')}* - The dialog box has a *Hello world!* message displayed. • *${ask('Hello world!', generic, 'Hello again!')}* - The dialog box has a *Hello world!* message displayed and the value displayed in the input box is *'Hello again!'*.
relative_url	**Format:** *${ask('message', relative_url, 'default')}*
	Description: Input is considered a URL. Oxygen XML Editor tries to make the URL relative to that of the document you are editing. **Note:** If the *$ask* editor variable is expanded in content that is not yet saved (such as an *untitled* file, whose path cannot be determined), then Oxygen XML Editor will transform it into an absolute URL.

Parameter	
	Example: • *${ask('File location', relative_url, 'C:/example.txt')}* - The dialog box has the name *'File location'*. The URL inserted in the input box is made relative to the current edited document location.
combobox	**Format:** *${ask('message', combobox, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default')}*
	Description: Displays a dialog box that offers a drop-down menu. The drop-down menu is populated with the given *rendered_value* values. Choosing such a value will return its associated value (*real_value*). **Note:** The *'default'* parameter specifies the default selected value and can match either a key or a value.
	Example: • *${ask('Operating System', combobox, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'osx')}* - The dialog box has the name *'Operating System'*. The drop-down menu displays the three given operating systems. The associated value will be returned based upon your selection. **Note:** In this example, the default value is indicated by the *osx* key. However, the same result could be obtained if the default value is indicated by *Mac OS X*, as in the following example: *${ask('Operating System', combobox, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'Mac OS X')}* • *${ask('Mobile OS', combobox, ('win':'Windows Mobile';'ios':'iOS';'and':'Android'), 'Android')}*
editable_combobox	**Format:** *${ask('message', editable_combobox, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default')}*
	Description: Displays a dialog box that offers a drop-down menu with editable elements. The drop-down menu is populated with the given *rendered_value* values. Choosing such a value will return its associated real value (*real_value*) or the value inserted when you edit a list entry. **Note:** The *'default'* parameter specifies the default selected value and can match either a key or a value.
	Example: • *${ask('Operating System', editable_combobox, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'osx')}* - The dialog box has the name *'Operating System'*. The drop-down menu displays the three given operating systems and also allows you to edit the entry. The associated value will be returned based upon your selection or the text you input.
radio	**Format:** *${ask('message', radio, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default')}*

Parameter	
	Description: Displays a dialog box that offers a series of radio buttons. Each radio button displays a *'rendered_value* and will return an associated *real_value*. **Note:** The *'default'* parameter specifies the default selected value and can match either a key or a value.
	Example: • *${ask('Operating System', radio, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'osx')}* - The dialog box has the name *'Operating System'*. The radio button group allows you to choose between the three operating systems. **Note:** In this example `Mac OS X` is the default selected value and if selected it would return *osx* for the output.

  - 'default-value' - optional parameter. Provides a default value.
- *${timeStamp}* - Time stamp, that is the current time in Unix format. For example, it can be used to save transformation results in multiple output files on each transformation.
- *${uuid}* - Universally unique identifier, a unique sequence of 32 hexadecimal digits generated by the Java *UUID* class.
- *${id}* - Application-level unique identifier. It is a short sequence of 10-12 letters and digits that is not guaranteed to be universally unique.
- *${cfn}* - Current file name without extension and without parent folder. The current file is the one currently opened and selected.
- *${cfne}* - Current file name with extension. The current file is the one currently opened and selected.
- *${cf}* - Current file as file path, that is the absolute file path of the current edited document.
- *${cfd}* - Current file folder as file path, that is the path of the current edited document up to the name of the parent folder.
- *${frameworksDir}* - The path (as file path) of the `[OXYGEN_INSTALL_DIR]/frameworks` directory.
- *${pd}* - The file path for the parent folder of the current project selected in the **Project** view.
- *${oxygenInstallDir}* - Oxygen XML Editor installation folder as file path.
- *${homeDir}* - The path (as file path) of the user home folder.
- *${pn}* - Current project name.
- *${env(VAR_NAME)}* - Value of the *VAR_NAME* environment variable. The environment variables are managed by the operating system. If you are looking for Java System Properties, use the *${system(var.name)}* editor variable.
- *${system(var.name)}* - Value of the *var.name* Java System Property. The Java system properties can be specified in the command line arguments of the Java runtime as *-Dvar.name=var.value*. If you are looking for operating system environment variables, use the *${env(VAR_NAME)}* editor variable instead.
- *${date(pattern)}* - Current date. The allowed patterns are equivalent to the ones in the *Java SimpleDateFormat class*. **Example:** *yyyy-MM-dd*;

  Note: This editor variable supports both the xs:date and xs:datetime parameters. For details about xs:date, go to *http://www.w3.org/TR/xmlschema-2/#date*. For details about xs:datetime, go to *http://www.w3.org/TR/xmlschema-2/#dateTime*.

Related Information:

Arguments of InsertFragmentOperation Operation

Arguments of SurroundWithFragmentOperation

Arguments of `InsertFragmentOperation` Operation

`fragment`

This argument has a textual value. This value is parsed by Oxygen XML Editor as it was already in the document at the cursor position. You can use entity references declared in the document and it is namespace aware. The fragment may have multiple roots.

You can even use namespace prefixes that are not declared in the inserted fragment, if they are declared in the document where the insertion is done. For the sake of clarity, you should always prefix and declare namespaces in the inserted fragment!

If the fragment contains namespace declarations that are identical to those found in the document, the namespace declaration attributes will be removed from elements contained by the inserted fragment.

There are two possible scenarios:

1. **Prefixes that are not bound explicitly**

   For instance, the fragment:

   ```
   <x:item id="dty2"/>
   &ent;
   <x:item id="dty3"/>
   ```

   Can be correctly inserted in the document: ('|' marks the insertion point):

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <!DOCTYPE x:root [
       <!ENTITY ent "entity">
   ]>

   <x:root xmlns:x="nsp">
     |
   </x:root>
   ```

   Result:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <!DOCTYPE x:root [
       <!ENTITY ent "entity">
   ]>
   <x:root xmlns:x="nsp">
       <x:item id="dty2"/>
       &ent;
       <x:item id="dty3"/>
   </x:root>
   ```

2. **Default namespaces**

   If there is a default namespace declared in the document and the *document fragment* does not declare a namespace, the elements from the fragment are considered to be in **no namespace**.

   For instance, the fragment:

   ```
   <item id="dty2"/>
   <item id="dty3"/>
   ```

   Inserted in the document:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <root xmlns="nsp">
   |
   </root>
   ```

   Gives the result document:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <root xmlns="nsp">
       <item xmlns="" id="dty2"/>
       <item xmlns="" id="dty3"/>
   </root>
   ```

`insertLocation`

An XPath expression that is relative to the current node. It selects the reference node for the fragment insertion. When missing, the fragment will be inserted at the cursor position.

`insertPosition`

Specifies where the insertion is made relative to the reference node selected by the `insertLocation`. It can be one of the following constants:

- **Inside as first child** (default value) - The fragment is inserted as first child of the reference node.
- **Inside as last child** - The fragment is inserted as the last child of the reference node.
- **After** - The fragment is inserted after the reference node.
- **Before** - The fragment is inserted before the reference node.

`goToNextEditablePosition`

After inserting the fragment, the first editable position is detected and the cursor is placed at that location. It handles any in-place editors used to edit attributes. It will be ignored if the fragment specifies a cursor position using the *${caret} editor variable*. The possible values of this action are **true** and **false**.

`schemaAware`

This argument applies only on the surround with element operation and controls whether or not the insertion is valid, based upon the schema. If the insertion is not valid, then wrapping action will be broken up into smaller intervals until the wrapping action is valid. For example, if you try to wrap a *paragraph* element with a *bold* element, it would not be valid, so the operation will wrap the text inside the paragraph instead, since it would be valid at that position.

Arguments of `SurroundWithFragmentOperation`

The **Author** mode operation `SurroundWithFragmentOperation` has only one argument:

- fragment

  The XML fragment that will surround the selection. For example, consider the fragment:

```
<F>
   <A></A>
   <B>
     <C></C>
   </B>
</F>
```

  and the document:

```
<doc>
  <X></X>
  <Y></Y>
  <Z></Z>
<doc>
```

  Considering the selected content to be surrounded is the sequence of elements X and Y, then the result is:

```
<doc>
   <F>
      <A>
         <X></X>
         <Y></Y>
      </A>
      <B>
        <C></C>
      </B>
   </F>
  <Z></Z>
<doc>
```

  Since the element A was the first leaf in the fragment, it received the selected content. The fragment was then inserted in the place of the selection.

Add a Custom Operation to an Existing Framework

This task explains how to add a custom **Author** mode operation to an existing *framework* (document type).

1. Setup a sample project by following the *instructions for installing the SDK*. The *framework* project is `oxygen-sample-framework`.

2. A number of classes in the `simple.documentation.framework.operations` package implement the `ro.sync.ecss.extensions.api.AuthorOperation` interface. Depending on your use-case, modify one of these classes.
3. Pack the operation class inside a Java *jar* library.
4. Copy the *jar* library to the `[OXYGEN_INSTALL_DIR]/frameworks/[FRAMEWORK_DIR]` directory.
5. *Open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, and edit the document type (you need write access to the `[OXYGEN_INSTALL_DIR]`) to open the ***Document Type** configuration dialog box*.
   a) In the **Classpath** tab, add a new entry similar to: `${framework}/customAction.jar`.
   b) In the **Author** tab, add a new action that uses your custom operation.
   c) Mount the action to the toolbars or menus.
6. Share the modifications with your colleagues. The files that should be shared are your `customAction.jar` library and the `.framework` configuration file from the `[OXYGEN_INSTALL_DIR]/frameworks/[FRAMEWORK_DIR]` directory.

Related Information:

Adding Retina/HiDPI Icons in a Framework

Higher resolution icons can also be included in customized *frameworks* for rendering them in a Retina or HiDPI display. The icons can be referenced directly from the ***Document Type Congifuration** dialog box* (from the ***Action** dialog box*) or from an API (`ro.sync.exml.workspace.api.node.customizer.XMLNodeRendererCustomizer`).

As with any image, the higher resolution icons are stored in the same images folder as the normal resolution images and they are identified by a scaling factor that is included in the name of the image files. For instance, icons with a Retina scaling factor of 2 will include *@2x* in the name (for example, `myIcon@2x.png`).

Developers should not specify the path of the alternate icons (*@2x* or *@3x*) in the ***Action** dialog box* or the *`XMLNodeRendererCustomizer` API*. When using a Retina or HiDPI display, Oxygen XML Editor automatically searches the folder of the *normal* icon for a corresponding image file with a Retina scaling factor in the name. If the higher resolution icon file does not exist, the *normal* icon is scaled and used instead.

Related Information:

Using Retina/HiDPI Images in Author Mode on page 413

Java API - Extending Author Functionality through Java

Oxygen XML Editor **Author** mode has a built-in set of operations covering the insertion of text and XML fragments (see the *Author Default Operations*) and the execution of XPath expressions on the current document edited in **Author** mode. However, there are situations in which you need to extend this set. The following examples are just a few of the possible situations:

- You need to enter an element whose attributes will be edited by the user through a graphical user interface.
- The user must send selected element content (or the whole document) to a server for some kind of processing.
- Content authors need to extract pieces of information from a server and insert it directly into the edited XML document.
- You need to apply an XPath expression on the current document and process the nodes of the resulting node set.

To extend the Oxygen XML Editor **Author** mode functionality through Java, you will need the *Oxygen SDK* available *on the Oxygen XML Editor website*. It includes the source code of the **Author** mode operations in the predefined document types and the full documentation (in Javadoc format) of the public API available for **Author** mode custom actions.

The subsequent Java examples make use of AWT classes. If you are developing extensions for the Oxygen XML Editor XML Editor plugin for Eclipse, you will have to use their SWT counterparts.

Attention: Make sure the Java classes of your custom **Author** mode operations are compiled with the same Java version used by Oxygen XML Editor. Otherwise, the classes may not be loaded by the Java virtual machine. For example, if you run Oxygen XML Editor with a Java 1.6 virtual machine but the Java

classes of your custom **Author** mode operations are compiled with a Java 1.7 virtual machine then the custom operations cannot be loaded and used by the Java 1.6 virtual machine.

Related Information:

Example 1- Simple Use of a Dialog Box from an Author Mode Operation

In this example, we start adding functionality for inserting images in a custom *framework*. The images are represented by the `image` element. The location of the image file is represented by the value of the `href` attribute. In the Java implementation, a dialog box will be displayed with a text field, in which the user can enter a full URL or browse for a local file.

1. Setup a sample project following *this set of instructions*. The *framework* project is **oxygen-sample-framework**.
2. Modify the `simple.documentation.framework.InsertImageOperation` class that implements the *`ro.sync.ecss.extensions.api.AuthorOperation`* interface. This interface defines three methods: `doOperation`, `getArguments` and `getDescription`

   A short description of these methods follows:

   - The `doOperation` method is invoked when the action is performed either by pressing the toolbar button, by selecting the menu item or by pressing the shortcut key. The arguments taken by this methods can be one of the following combinations:
     - an object of type *`ro.sync.ecss.extensions.api.AuthorAccess`* and a map
     - argument names and values
   - The `getArguments` method is used by Oxygen XML Editor when the action is configured. It returns the list of arguments (name and type) that are accepted by the operation.
   - The `getDescription` method is used by Oxygen XML Editor when the operation is configured. It returns a description of the operation.

   Example:

   Here is the implementation of these three methods:

```
/**
 * Performs the operation.
 */
public void doOperation(
            AuthorAccess authorAccess,
            ArgumentsMap arguments)
    throws IllegalArgumentException,
               AuthorOperationException {

JFrame oxygenFrame = (JFrame) authorAccess.getWorkspaceAccess().getParentFrame()
;
    String href = displayURLDialog(oxygenFrame);
     if (href.length() != 0) {
       // Creates the image XML fragment.
       String imageFragment =
         "<image xmlns='http://www.oxygenxml.com/sample/documentation' href='"
         + href + "'/>";

       // Inserts this fragment at the cursor position.
       int caretPosition = authorAccess.getEditorAccess().getCaretOffset();
       authorAccess.getDocumentController().insertXMLFragment
(imageFragment, caretPosition);
 }
}

/**
 * Has no arguments.
 *
 * @return null.
 */
public ArgumentDescriptor[] getArguments() {
 return null;
}

/**
 * @return A description of the operation.
 */
public String getDescription() {
 return "Inserts an image element. Asks the user for a URL reference.";
}
```

   Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

Important:

Make sure you always specify the namespace of the inserted fragments.

```
<image xmlns='http://www.oxygenxml.com/sample/documentation'
  href='path/to/image.png'/>
```

3. Package the compiled class into a *JAR* file. An example of an Ant script that packages the `classes` folder content into a *JAR* archive named `sdf.jar` is listed below:

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="project" default="dist">
    <target name="dist">
   <jar destfile="sdf.jar" basedir="classes">
    <fileset dir="classes">
     <include name="**/*"/>
     </fileset>
   </jar>
    </target>
</project>
```

4. Copy the `sdf.jar` file into your custom *framework* directory (*`[OXYGEN_INSTALL_DIR]`*`\frameworks\[CUSTOM_FRAMEWORK_DIR]`).
5. Add the `sdf.jar` to the class path. To do this, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, select **SDF**, and press the **Edit** button.
6. Select the **Classpath** tab in the lower part of the ***Document Type** configuration dialog box* and press the **Add** button . In the displayed dialog box, enter the location of the *JAR* file, relative to the Oxygen XML Editor `frameworks` folder.
7. We now create the action that will use the defined operation. Go to the **Actions** subtab.
   Copy the icon files for the menu item and for the toolbar in your custom *framework* directory (*`[OXYGEN_INSTALL_DIR]`*`\frameworks\[CUSTOM_FRAMEWORK_DIR]`).
8. Define the action's properties:
   - Set **ID** to **insert_image**.
   - Set **Name** to **Insert image**.
   - Set **Menu access key** to letter **i**.
   - Set **Toolbar action** to **${framework}/toolbarImage.png**.
   - Set **Menu icon** to **${framework}/menuImage.png**.
   - Set **Shortcut key** to **Ctrl (Meta on Mac OS)+Shift+i**.
9. Next, we set up the operation. You want to add images only if the current element is a `section`, book or `article`.
   - Set the value of **XPath expression** to

     ```
     local-name()='section' or local-name()='book'
      or local-name()='article'
     ```

   - Set the **Invoke operation** field to `simple.documentation.framework.InsertImageOperation`.

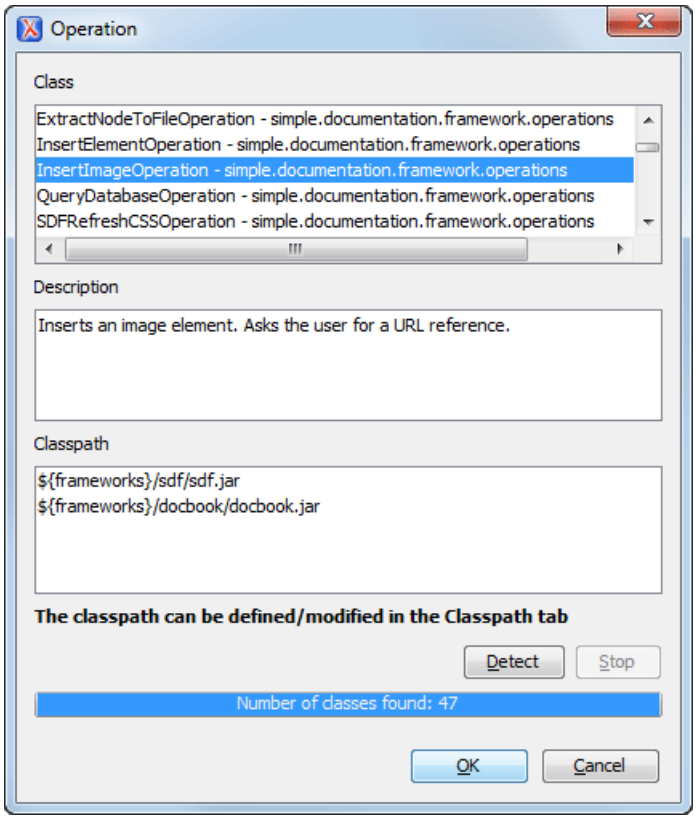


Figure 569: Selecting the Operation

10. Add the action to the toolbar, using the **Toolbar** panel.

To test the action, you can open the *`sdf_sample.xml`* sample, then place the cursor inside a `section` between two para elements (for instance). Press the button associated with the action from the toolbar. In the dialog box, select an image URL and press **OK**. The image is inserted into the document.

Example 2- Operations with Arguments. Report from Database Operation

In this example you will create an operation that connects to a relational database and executes an SQL statement. The result should be inserted in the edited XML document as a `table`. To make the operation fully configurable, it will have arguments for the *database connection string*, the *user name*, the *password* and the *SQL expression*.

1. Setup a sample project following *this set of instructions*. The *framework* project is **oxygen-sample-framework**.
2. Create the class `simple.documentation.framework.QueryDatabaseOperation`. This class must implements the *`ro.sync.ecss.extensions.api.AuthorOperation`* interface.

```
import ro.sync.ecss.extensions.api.ArgumentDescriptor;
import ro.sync.ecss.extensions.api.ArgumentsMap;
import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.AuthorOperation;
import ro.sync.ecss.extensions.api.AuthorOperationException;

public class QueryDatabaseOperation implements AuthorOperation{
```

3. Now define the operation's arguments. For each of them you will use a `String` constant representing the argument name:

```
private static final String ARG_JDBC_DRIVER ="jdbc_driver";
private static final String ARG_USER ="user";
private static final String ARG_PASSWORD ="password";
private static final String ARG_SQL ="sql";
private static final String ARG_CONNECTION ="connection";
```

4. You must describe each of the argument name and type. To do this, implement the `getArguments` method that will return an array of argument descriptors:

```
public ArgumentDescriptor[] getArguments() {
  ArgumentDescriptor args[] = new ArgumentDescriptor[] {
    new ArgumentDescriptor(
      ARG_JDBC_DRIVER,
      ArgumentDescriptor.TYPE_STRING,
      "The name of the Java class that is the JDBC driver."),
    new ArgumentDescriptor(
      ARG_CONNECTION,
      ArgumentDescriptor.TYPE_STRING,
      "The database URL connection string."),
    new ArgumentDescriptor(
      ARG_USER,
      ArgumentDescriptor.TYPE_STRING,
      "The name of the database user."),
    new ArgumentDescriptor(
      ARG_PASSWORD,
      ArgumentDescriptor.TYPE_STRING,
      "The database password."),
    new ArgumentDescriptor(
      ARG_SQL,
      ArgumentDescriptor.TYPE_STRING,
      "The SQL statement to be executed.")
  };
  return args;
 }
```

These names, types and descriptions will be listed in the **Arguments** table when the operation is configured.

5. When the operation is invoked, the implementation of the `doOperation` method extracts the arguments, forwards them to the method that connects to the database and generates the XML fragment. The XML fragment is then inserted at the cursor position.

```
public void doOperation(AuthorAccess authorAccess, ArgumentsMap map)
   throws IllegalArgumentException, AuthorOperationException {

  // Collects the arguments.
  String jdbcDriver =
   (String)map.getArgumentValue(ARG_JDBC_DRIVER);
  String connection =
   (String)map.getArgumentValue(ARG_CONNECTION);
  String user =
   (String)map.getArgumentValue(ARG_USER);
  String password =
   (String)map.getArgumentValue(ARG_PASSWORD);
  String sql =
   (String)map.getArgumentValue(ARG_SQL);

  int caretPosition = authorAccess.getCaretOffset();
  try {
   authorAccess.getDocumentController().insertXMLFragment(
     getFragment(jdbcDriver, connection, user, password, sql),
     caretPosition);
  } catch (SQLException e) {
   throw new AuthorOperationException(
     "The operation failed due to the following database error: "
     + e.getMessage(), e);
  } catch (ClassNotFoundException e) {
   throw new AuthorOperationException(
     "The JDBC database driver was not found. Tried to load ' "
     + jdbcDriver + "'", e);
  }
 }
```

6. The `getFragment` method loads the JDBC driver, connects to the database and extracts the data. The result is a `table` element from the `http://www.oxygenxml.com/sample/documentation` namespace. The `header` element contains the names of the SQL columns. All the text from the XML fragment is escaped. This means that the '<' and '&' characters are replaced with the '<' and '&' character entities to ensure the fragment is well-formed.

```
private String getFragment(
  String jdbcDriver,
  String connectionURL,
  String user,
  String password,
  String sql) throws
   SQLException,
   ClassNotFoundException {

      Properties pr = new Properties();
      pr.put("characterEncoding", "UTF8");
      pr.put("useUnicode", "TRUE");
      pr.put("user", user);
      pr.put("password", password);
```

```
        // Loads the database driver.
        Class.forName(jdbcDriver);
        // Opens the connection
        Connection connection =
           DriverManager.getConnection(connectionURL, pr);
        java.sql.Statement statement =
           connection.createStatement();
        ResultSet resultSet =
           statement.executeQuery(sql);

        StringBuffer fragmentBuffer = new StringBuffer();
        fragmentBuffer.append(
          "<table xmlns=" +
          "'http://www.oxygenxml.com/sample/documentation'>");

        //
        // Creates the table header.
        //
        fragmentBuffer.append("<header>");
        ResultSetMetaData metaData = resultSet.getMetaData();
        int columnCount = metaData.getColumnCount();
        for (int i = 1; i <= columnCount; i++) {
            fragmentBuffer.append("<td>");
            fragmentBuffer.append(
              xmlEscape(metaData.getColumnName(i)));
            fragmentBuffer.append("</td>");
        }
        fragmentBuffer.append("</header>");

        //
        // Creates the table content.
        //
        while (resultSet.next()) {
            fragmentBuffer.append("<tr>");
            for (int i = 1; i <= columnCount; i++) {
                fragmentBuffer.append("<td>");
                fragmentBuffer.append(
                  xmlEscape(resultSet.getObject(i)));
                fragmentBuffer.append("</td>");
            }
            fragmentBuffer.append("</tr>");
        }

        fragmentBuffer.append("</table>");

        // Cleanup
        resultSet.close();
        statement.close();
        connection.close();
        return fragmentBuffer.toString();
}
```

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

7. Package the compiled class into a *JAR* file.
8. Copy the *JAR* file and the JDBC driver files into your custom *framework* directory (`[OXYGEN_INSTALL_DIR]\frameworks\[CUSTOM_FRAMEWORK_DIR]`).
9. Add the *JARS* to the class path. To do this, open the ***Document Type Association*** *preferences page*, select **SDF** and press the **Edit** button. Select the **Classpath** tab in the lower part of the ***Document Type*** *configuration dialog box* and press the **Add** button . In the displayed dialog box, enter the location of the *JAR* file, relative to the Oxygen XML Editor `frameworks` folder.
10. Go to the **Actions** subtab. The action properties are:
    - Set **ID** to **clients_report**.
    - Set **Name** to **Clients Report**.
    - Set **Menu access key** to letter **r**.
    - Set **Description** to **Connects to the database and collects the list of clients**.
    - Set **Toolbar icon** to **${framework}/TableDB20.png** (image `TableDB20.png` is already stored in the `frameworks / sdf` folder).
    - Leave empty the **Menu icon**.
    - Set **shortcut key** to **Ctrl + Shift + C (Command + Shift + C on OS X)**.
11. The action will work only if the current element is a **section**. Set up the operation as follows:
    - Set **XPath expression** to:

```
local-name()='section'
```

- Use the Java operation defined earlier to set the **Invoke operation** field. Press the **Choose** button, then select `simple.documentation.framework.QueryDatabaseOperation`. Once selected, the list of arguments is displayed. In the figure below the first argument, *jdbc_driver*, represents the class name of the MySQL JDBC driver. The connection string has the URL syntax : *jdbc://<database_host>:<database_port>/<database_name>*.

  The SQL expression used in the example follows, but it can be any valid SELECT expression that can be applied to the database:

```
SELECT userID, email FROM users
```

12. Add the action to the toolbar, using the **Toolbar** panel.

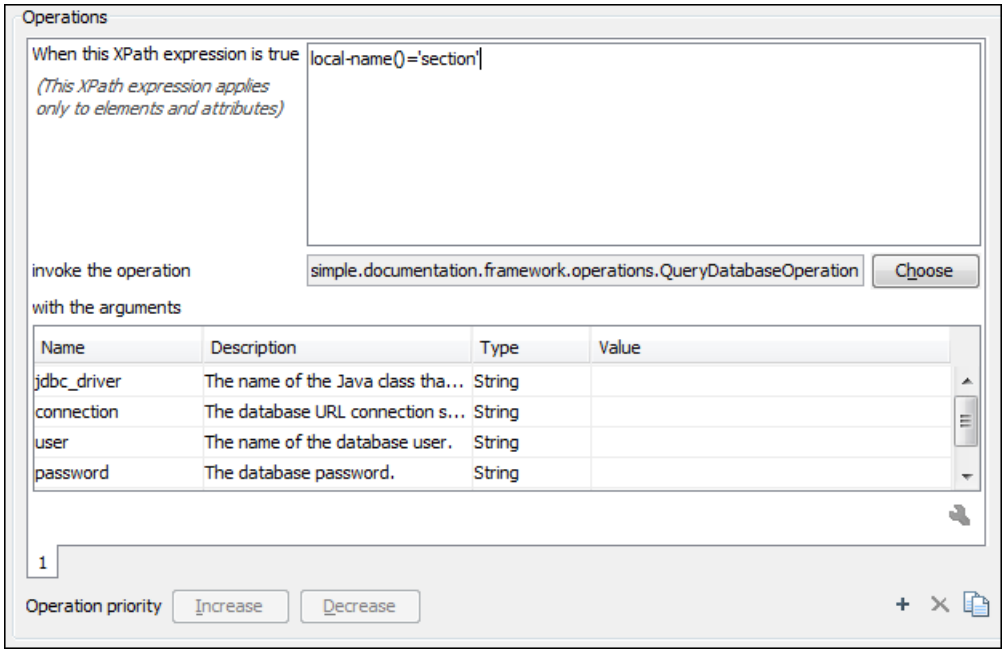


Figure 570: Java Operation Arguments Setup

To test the action, you can open the *`sdf_sample.xml`* sample and place the cursor inside a `section` between two para elements (for instance). Press the **Create Report** button from the toolbar. You can see below the toolbar with the action button and sample table inserted by the **Clients Report** action.

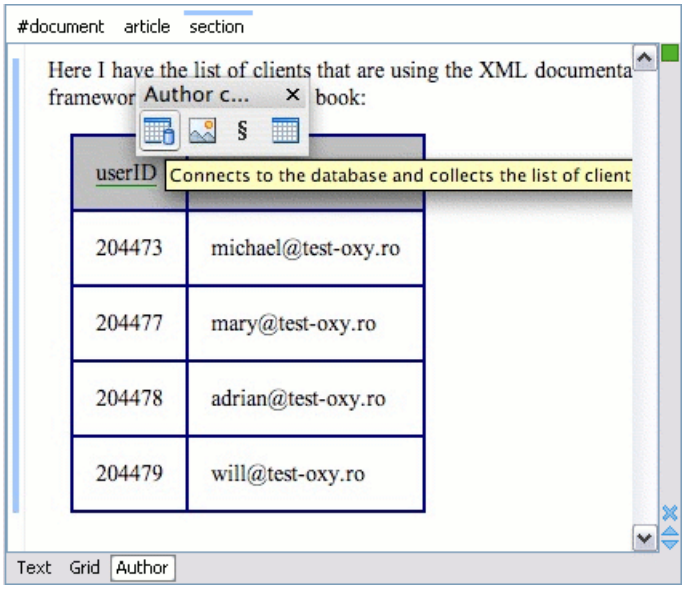


Figure 571: Table Content Extracted from the Database

Localizing Frameworks

Oxygen XML Editor supports *framework* localization (translating *framework* actions, buttons, and menu entries to various languages). This lets you develop and distribute a *framework* to users that speak other languages without changing the distributed *framework*. Changing the language used in Oxygen XML Editor in the Global preferences page is enough to set the right language for each *framework*.

To localize the content of a *framework*, create a `translation.xml` file that contains all the translation (key, value) mappings. The `translation.xml` has the following format:

```
<translation>
    <languageList>
        <language description="English" lang="en_US"/>
        <language description="German" lang="de_DE"/>
        <language description="French" lang="fr_FR"/>
    </languageList>
    <key value="list">
        <comment>List menu item name.</comment>
        <val lang="en_US">List</val>
        <val lang="de_DE">Liste</val>
        <val lang="fr_FR">Liste</val>
    </key>
.....................
</translation>
```

Oxygen XML Editor matches the GUI language with the language set in the `translation.xml` file. If this language is not found, the first available language declared in the `languagelist` tag for the corresponding *framework* is used.

Add the directory where this file is located to the **Classpath** list corresponding to the edited document type.

After you create this file, you can use the keys defined in it to customize the name and description of the following:

- Actions
- Menu entries
- Contextual menus
- Toolbars
- Static CSS content

For example, if you want to localize the bold action, *open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Document Type Association**. Use the **New** or **Edit** button to open the ***Document type** configuration dialog box*, go to **Author** > **Actions**, and rename the bold action to `${i18n(translation_key)}`. Actions with a name format other than `${i18n(translation_key)}` are not localized. `Translation_key` corresponds to the key from the translation.xml file.

Now open the `translation.xml` file and edit the translation entry if it exists or create one if it does not exist. This example presents an entry in the translation.xml file:

```
<key value="translation_key">
        <comment>Bold action name.</comment>
        <val lang="en_US">Bold</val>
        <val lang="de_DE">Bold</val>
        <val lang="fr_FR">Bold</val>
    </key>
```

To use a description from the `translation.xml` file in the Java code used by your custom *framework*, use the new *`ro.sync.ecss.extensions.api.AuthorAccess.getAuthorResourceBundle()`* API method to request the associated value for a certain key. This allows all the dialog boxes that you present from your custom operations to have labels translated in multiple languages.

You can also reference a key directly in the CSS content:

```
title:before{
    content:"${i18n(title.key)} : ";
}
```

Note: You can enter any language you want in the `languagelist` tag and any number of keys.

The `translation.xml` file for the DocBook *framework* is located here:*[OXYGEN_INSTALL_DIR]/*`frameworks/docbook/i18n/translation.xml`. In the **Classpath** list corresponding to the DocBook document type the following entry was added: `${framework}/i18n/`.

To see how the DocBook actions are defined to use these keys for their name and description, *open the **Preferences** dialog box **(Options > Preferences)*** and go to **Document Type Association** > **Author** > **Actions**. If you look in the Java class `ro.sync.ecss.extensions.docbook.table.SADocbookTableCustomizerDialog` available in the *oxygen-sample-framework* module of the *Oxygen SDK* Maven archetype, you can see how the new `ro.sync.ecss.extensions.api.AuthorResourceBundle` API is used to retrieve localized descriptions for various keys.

Packing and Deploying Plugins or Frameworks as Add-ons

Packing a Plugin or Framework as an Add-on

This procedure is suitable for developers who want a better control over the *add-on* package or those who want to automate some of the steps:

1. Pack the *plugin* or *framework* as a `ZIP` file or a *Java Archive*. Note that you should pack the entire root directory not just its contents.
2. Digitally sign the package. Note that you can perform this step only if you have created a *Java Archive* at the previous step. You will need a certificate signed by a trusted authority. To sign the *JAR* you can either use the `jarsigner` command line tool inside Oracle's Java Development Kit. (`[JDK_DIR]/bin/jarsigner.exe`) or, if you are working with *Apache Ant*, you can use the `signjar` task (a front for the `jarsigner` command line tool).

   Note: The benefit of having a signed add-on is that you can verify the integrity of the add-on issuer. If you do not have such a certificate you can generate one yourself using the *keytool* command line tool. This approach is mostly recommended for tests since anyone can create a self signed certificate.
3. Create a descriptor file. You can use a template that Oxygen XML Editor provides. To use this template, go to **File** > **New** and select the **Oxygen add-ons update site** template. Once deployed, this descriptor file is referenced as *update site*.

Alternatively, you can use the **Add-ons Packager** plugin by following this procedure:

1. Install the **Add-ons Packager** plugin from *https://www.oxygenxml.com/InstData/Addons/optional/updateSite.xml* as described in the *Installing Add-ons procedure*.
2. Restart Oxygen XML Editor. If the add-on is correctly installed, the **Add-ons packager** toolbar action is available.
3. Invoke the **Add-ons packager** toolbar action and input the required information in the displayed dialog box.
4. Press **OK** to complete the packaging process.

Deploying an Add-on

To deploy an add-on, copy the `ZIP` or *Java Archive* file and the descriptor file to an HTTP server. The URL to this location serves as the *Update Site URL*.

Configuring New File Templates

A customized *framework* (document type) can point to a directory, usually named `templates`, that contains the file templates. All files found here are considered templates for the respective document type. The template name is taken from the file name, and the template type is detected from the file extension.

To customize new file templates for a *framework*, follow these steps:

1. Go to your custom *framework* directory (*`[OXYGEN_INSTALL_DIR]`*`\frameworks\[CUSTOM_FRAMEWORK_DIR]`) and create a directory named `templates`.
   The directory tree of the custom documentation *framework* should be something like this:

```
[OXYGEN_INSTALL_DIR]
    frameworks
        [CUSTOM_FRAMEWORK_DIR]
            schema
            css
```

```
            templates
```

2. In the `templates` directory, create two files. A file for the *book* template and another one for the *article* template.

   An example for the `Book.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.oxygenxml.com/sample/documentation"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:abs="http://www.oxygenxml.com/sample/documentation/abstracts">
    <title>Book Template Title</title>
    <section>
        <title>Section Title</title>
        <abs:def/>
        <para>This content is copyrighted:</para>
        <table>
            <header>
                <td>Company</td>
                <td>Date</td>
            </header>
            <tr>
                <td/>
                <td/>
            </tr>
        </table>
    </section>
    </book>
```

   An example for the `Article.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<article
    xmlns="http://www.oxygenxml.com/sample/documentation"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <title></title>
    <section>
        <title></title>
        <para></para>
        <para></para>
    </section>
</article>
```

   You can also use *editor variables* in the content of the template files and they will be expanded when the files are opened.

   Note: You should avoid using the `${cfd}`,`${cf}`,`${currentFileURL}`, and `${cfdu}` *editor variables* when you save your documents in a data base.

3. Open the ***Document Type*** *configuration dialog box* for your custom *framework* and click the **Templates** tab. In the **Templates directory** text field, introduce the `${frameworkDir}/templates` path. It is recommended that all the file references made from a *Document Type Association* to be relative to the *${frameworkDir}* directory. Binding to an absolute file (e. g.: `C:\some_dir\templates`) makes the association difficult to share between users.
4. To test the templates settings, go to **File** > **New** to display the **New** document dialog box. You should see the new templates in the folder for your custom *framework* (in the **Framework templates** section). The names of the two templates are prefixed with the name of the *framework*. Selecting one of them should create a new XML file with the content specified in the template file.

Related Information:

Editor Variables on page 169
Custom Editor Variables on page 174

Configuring XML Catalogs

For cases where you need to reference the location of a schema file from a remote web location and an Internet connection may not be available, an *XML Catalog* may be used to map the web location to a local file system entry. The following procedure presents an example of using an *XML catalog* in a custom *framework* by modifying an XML Schema file.

1. Create a catalog file that will help the parser locate the schema for validating the XML document. The file must map the location of the schema to a local version of the schema.

   Example:

Create a new XML file called `catalog.xml` and save it in your custom *framework* directory (`[OXYGEN_INSTALL_DIR]\frameworks\[CUSTOM_FRAMEWORK_DIR]`). The content of the file should look like this:

```
<?xml version="1.0"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
    <uri name="http://www.oxygenxml.com/SDF/abs.xsd"
            uri="schema/abs.xsd"/>
    <uri name="http://www.oxygenxml.com/SDF/abs.xsd"
                uri="schema/abs.xsd"/>
</catalog>
```

2. Add catalog files to your custom *framework* using the ***Catalogs** tab* from the ***Document Type** configuration dialog box*.

To test the catalog settings, restart Oxygen XML Editor and try to validate a new sample document for your custom *framework*. There should be no errors.

Example:

The schema that validates the document refers the other file `abs.xsd` through an import element:

```
<xs:import namespace=
 "http://www.oxygenxml.com/sample/documentation/abstracts"
 schemaLocation="http://www.oxygenxml.com/SDF/abs.xsd"/>
```

The `schemaLocation` attribute references the `abs.xsd` file:

```
xsi:schemaLocation="http://www.oxygenxml.com/sample/documentation/abstracts"
    http://www.oxygenxml.com/SDF/abs.xsd"/>
```

The catalog mapping is:

```
http://www.oxygenxml.com/SDF/abs.xsd -> schema/abs.xsd
```

This means that all the references to http://www.oxygenxml.com/SDF/abs.xsd must be resolved to the `abs.xsd` file located in the `schema` directory (note that the `schema` directory needs to be in the same folder as the *XML Catalog*). The URI element is used by URI resolvers (for example, to resolve a URI reference used in an XSLT stylesheet).

Configuring Transformation Scenarios for a Framework

When distributing a *framework* to users, it is a good idea to have the transformation scenarios already configured. This helps the content authors publish their work in various formats. By being contained in the *framework* configuration, the scenarios can be distributed along with the actions, menus, toolbars, and catalogs.

These are the steps that allow you to create a transformation scenario for your *framework*.

1. Create an `xsl` folder inside your custom *framework* directory (`[OXYGEN_INSTALL_DIR]\frameworks\[CUSTOM_FRAMEWORK_DIR]`).

   The folder structure for the documentation *framework* should be:

   ```
   oxygen
     frameworks
        [CUSTOM_FRAMEWORK_DIR]
          schema
          css
          templates
          xsl
   ```

2. Create the `sdf.xsl` file in the `xsl` folder. The complete content of the `sdf.xsl` file is found in the *Example Files Listings*.
3. *Open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Document Type Associations**. Select the particular *framework*, click the **Edit** button to open ***Document Type** Configuration dialog box*, and choose the **Transformation** tab. Click the ✚ ˅**New** button and choose the appropriate type of transformation (for example, **XML transformation with XSLT**).

   In the **New scenario** dialog box, fill in the following fields:

   - Fill in the **Name** field with the name of your transformation scenario.

- Set the **XSL URL** field to path of your custom stylesheet (for example, `${framework}/xsl/mycustom.xsl`).

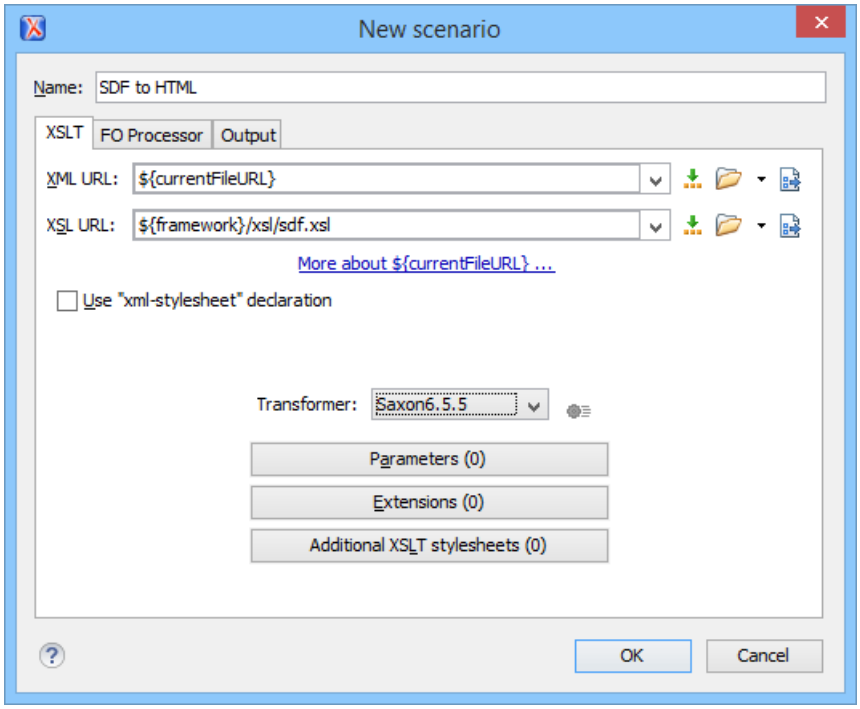


Figure 572: Configuring a New XSLT Transformation Scenario

4. Change to the **Output** tab. Configure the fields as follows:
   - Set the **Save as** field to `${cfd}/${cfn}.html`. This means the transformation output file will have the name of the XML file and the *html* extension and will be stored in the same folder.
   - Select the **Open in Browser/System Application** option.

     Note: To set the browser or system application that will be used, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.
   - Select the **Saved file** option.
5. Click the **OK** button to save the new scenario.

Now the scenario is listed in the **Transformation** tab:

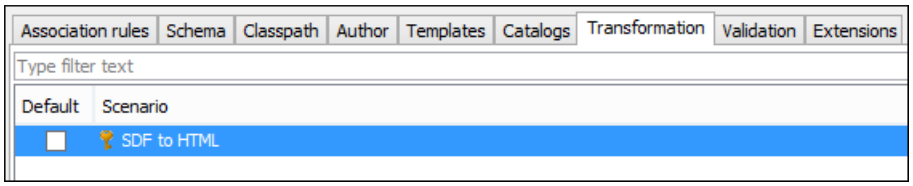


Figure 573: Transformation Tab

To test the transformation scenario that you just created, open the **SDF** XML sample from the *Example Files Listings*. Click the **Apply Transformation Scenario(s)** button to display the **Transform with** dialog box. The scenario list contains the scenario you defined earlier . Select the *SDF to HTML* scenario that you just defined and click the **Apply associated** button. The HTML file is saved in the same folder as the XML file and displayed in the browser.

Configuring Validation Scenarios for a Framework

You can distribute a *framework* with a series of already configured validation scenarios. Also, this provides enhanced validation support that allows you to use multiple grammars to check the document. For example,

you can use Schematron rules to impose guidelines that are otherwise impossible to enforce using conventional validation.

Note: If a *master file* is associated with the current file, the validation scenarios defined in the *master file*, along with any Schematron schema defined in the default scenarios for that particular *framework*, are used for the validation. These take precedence over other types of validation units defined in the default scenarios for the particular *framework*. For more information on *master files*, see *Master Files Support* on page 286 or *Working with Modular XML Files in the Master Files Context* on page 481.

To associate a validation scenario with a specific *framework*, follow these steps:

1. *Open the **Preferences** dialog box **(Options > Preferences)*** and go to **Document Type Association**.
2. Select the document type and click the **Edit** button to open the ***Document Type Configuration** dialog box*, then choose the **Validation** tab. This tab displays a list of document types in which you can define validation scenarios. To set one or more of the validation scenarios listed in this tab to be used as the default validation scenario (when another one is not specified in the validation process) for a specific document type, check the **Default** box for that specific document type.
3. To add a new scenario, press the **New** button.
   The **New scenarios** dialog box is displayed. It lists all the validation units for the scenario.

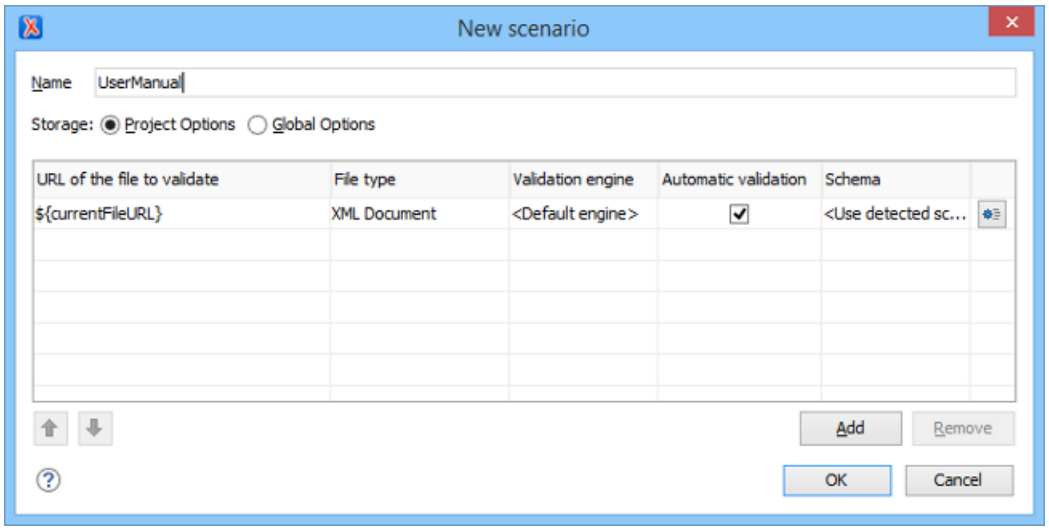


Figure 574: Create New Validation Scenario

This scenario configuration dialog box allows you to configure the following information and options:

Name

The name of the validation scenario.

Storage

You can choose between storing the scenario in the ***Project Options*** or ***Global Options***.

URL of the file to validate

The URL of the main module that includes the current module. It is also the entry module of the validation process when the current one is validated. To edit the URL, double-click its cell and specify the URL of the main module by doing one of the following:

- Enter the URL in the text field or select it from the drop-down list.
- Use the **Browse** drop-down button to browse for a local, remote, or archived file.
- Use the **Insert Editor Variable** button to insert an *editor variable* or a *custom editor variable*.

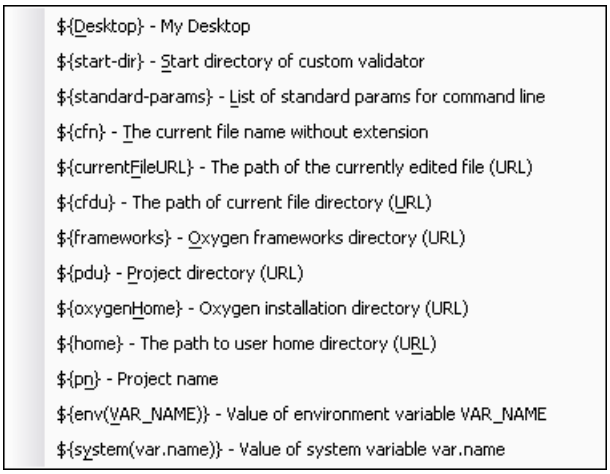


Figure 575: Insert an Editor Variable

File type

The type of the document that is validated in the current validation unit. Oxygen XML Editor automatically selects the file type depending on the value of the **URL of the file to validate** field.

Validation engine

You can select one of the engines available in Oxygen XML Editor for validation of the particular document type.

Default engine means that the default engine is used to run the validation for the current document type, as specified in the preferences page for that type of document (for example, *XSLT preferences page*, *XQuery preferences page*, *XML Schema preferences page*).

The **DITA Validation** engine performs DITA-specific checks in the context of the specifications (it is similar to the checks done with the **DITA Maps Manager** ***Validate and Check for Completeness*** *action*, but for a local file rather than an entire *DITA map*).

The **Table Layout Validation** engine looks for table layout problems (for more information, see *Report table layout problems* on page).

Automatic validation

If this option is selected, the validation operation defined by this row is also applied by *the automatic validation feature*. If the **Automatic validation** feature is disabled in the *Document Checking preferences page*, then this option is ignored, as the preference setting has a higher priority.

Schema

This option becomes active when you set the **File type** to **XML Document** and allows you to specify the schema used for the validation unit.

Specify Schema

Opens the **Specify Schema** dialog box that allows you to set a schema to be used for validating XML documents.

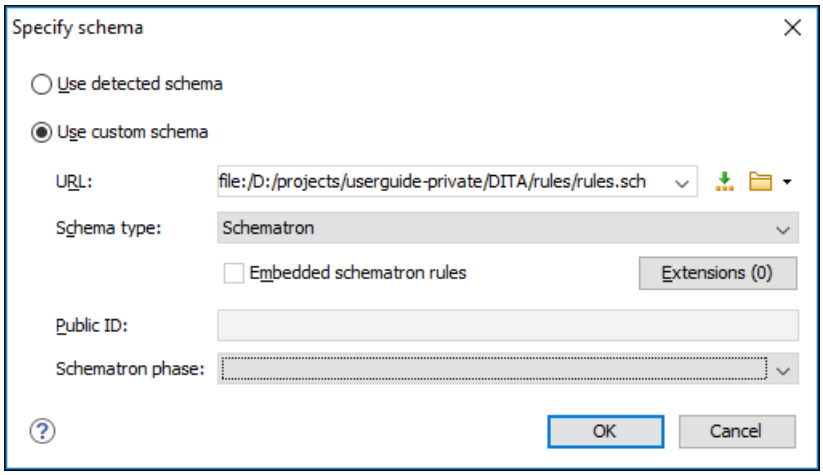


Figure 576: Specify Schema Dialog Box

The **Specify Schema** dialog box contains the following options:

Use detected schema

Uses the *schema detected for the particular document*.

Use custom schema

Allows you to specify the schema using the following options:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a *custom protocol*. You can specify the URL by using the text field, the history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.
- **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the **URL** field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.
- **Embedded schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.
- **Extensions**- Opens a dialog box that allows you to specify *Java extension JARs* to be used during the validation.
- **Public ID** - Allows you to specify a public ID if you have selected a DTD.
- **Schematron phase** - If you select a Schematron schema, this drop-down list allows you to select a Schematron phase that you want to use for validation. The listed phases are defined in the Schematron document.

Move Up

Moves the selected scenario up one spot in the list.

Move Down

Moves the selected scenario down one spot in the list.

Add

Adds a new validation unit to the list.

Remove

Removes an existing validation unit from the list.

4. Configure any of the existing validation units according to the information above, and you can use the buttons at the bottom of the table to add, remove, or move validation units. Note that if you add a Schematron validation unit, you can also select the **validation phase**.
5. Press **Ok**.

The newly created validation scenario is now included in the list of scenarios in the ***Validation** tab*. You can use the **Default** checkbox to specify that the new scenario be used as the default validation scenario when another specific scenario is not specified in the validation process.

Customizing Smart Paste Support

The *Smart Paste feature* preserves certain style and structure information when copying content from some of the most common applications and pasting into *frameworks (document types) that support Smart Paste* in Oxygen XML Editor. For other document types, the default behavior of the paste operation is to keep only the text content without the styling.

The style of the pasted content can be customized by editing an XSLT stylesheet for a particular document type (*framework*). The XSLT stylesheet must accept an XHTML flavor of the copied content as input, and transform it to the equivalent XML markup that is appropriate for the target document type of the paste operation.

How to Customize the Smart Paste Mapping

To customize the mapping between the markup of the copied content and the markup of the pasted content for a particular document type, follow these steps:

1. Make sure the particular *framework* contains a folder named `resources` in the following path structure:

   `[OXYGEN_INSTALL_DIR]``/frameworks/[Document Type]/resources`
2. Create an XSLT file named `xhtml2content.xsl` and save it in the `resources` folder for the particular *framework*.

   For example: *`[OXYGEN_INSTALL_DIR]`*`/frameworks/dita/resources/xhtml2content.xsl`
3. Add your customized styling in the XSLT file.
4. You can test modifications done in the stylesheet by pasting content without having to restart Oxygen XML Editor.

Result: When you paste content from external applications (such as a web browser or and Office document) to a document that is opened in **Author** mode, and that matches the particular *framework*, the styling from the `xhtml2content.xsl` stylesheet will be applied on the clipboard contents.

Customized Smart Paste Stylesheet Sample:

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xhtml="http://www.w3.org/1999/xhtml"
  xmlns="urn:hl7-org:v3"
  exclude-result-prefixes="xsl xhtml">

  <xsl:output method="xml" indent="no" omit-xml-declaration="yes"/>

  <xsl:template match="xhtml:b | xhtml:strong">
    <content styleCode="bold" >
      <xsl:apply-templates select="@* | node()"/>
    </content>
  </xsl:template>

  <xsl:template match="*">
    <xsl:apply-templates select="@* | node()"/>
  </xsl:template>

  <xsl:template match="@* | node()">
    <xsl:copy>
      <xsl:apply-templates select="@* | node()"/>
    </xsl:copy>
  </xsl:template>

</xsl:stylesheet>
```

Related Information:

Smart Paste Mechanism on page 343

Oxygen XML Blog: How Special Paste Works in Oxygen

Configuring Extensions

You can add *extensions* to your custom *framework* (document type) by using the **Extensions** tab from the ***Document Type** configuration dialog box*.

Note: It is possible for a *plugin* to share the same classes with a *framework*. For further details, go to *How to Share the Classloader Between a Framework and a Plugin*.

Configuring an Extensions Bundle

All *extensions* that are provided by Oxygen XML Editor are includes in a single bundle.

Note: The individual extensions can still be set (*open the **Preferences** dialog box **(Options** > **Preferences)***, go to **Document Type Association**, double-click a document type, and go to the extension tab), and if present, they take precedence over the single provider. However, this practice is discouraged and the single provider should be used instead.

The extensions bundle is represented by the `ro.sync.ecss.extensions.api.ExtensionsBundle` class. The provided implementation of the `ExtensionsBundle` is instantiated when the *Document Type Association* rules defined for the custom *framework* matches a document opened in the editor. Therefore, references to objects that need to be persistent throughout the application running session must not be kept in the bundle because the next detection event can result in creating another `ExtensionsBundle` instance.

To configure an extensions bundle, follow this procedure:

1. Create a new Java project in your IDE. Create a `lib` folder in the Java project folder and copy in it the `oxygen.jar` file from the `[OXYGEN_INSTALL_DIR]/lib` folder.
2. Create the class (for example, `simple.documentation.framework.SDFExtensionsBundle`) to extend the abstract class `ro.sync.ecss.extensions.api.ExtensionsBundle`.

   For example:

   ```
   public class SDFExtensionsBundle extends ExtensionsBundle {
   ```

3. A `Document Type ID` and a short description should be defined first by implementing the methods `getDocumentTypeID` and `getDescription`. The `Document Type ID` is used to uniquely identify the current *framework*. Such an ID must be provided especially if options related to the *framework* need to be persistently stored and retrieved between sessions.

   For example:

   ```
   public String getDocumentTypeID() {
        return "Simple.Document.Framework.document.type";
   }

   public String getDescription() {
        return "A custom extensions bundle used for the Simple Document" +
                       "Framework document type";
   }
   ```

4. To be notified about the activation of the custom *Author Extension* in relation with an opened document, `ro.sync.ecss.extensions.api.AuthorExtensionStateListener` should be implemented. The **activation** and **deactivation** events received by this listener should be used to perform custom initializations and to register or remove listeners such as `ro.sync.ecss.extensions.api.AuthorListener`, `ro.sync.ecss.extensions.api.AuthorMouseListener`, or `ro.sync.ecss.extensions.api.AuthorCaretListener`. The custom *Author Extension* state listener should be provided by implementing the `createAuthorExtensionStateListener` method.

   For example:

   ```
   public AuthorExtensionStateListener createAuthorExtensionStateListener() {
        return new SDFAuthorExtensionStateListener();
   }
   ```

   The `AuthorExtensionStateListener` is instantiated and notified about the activation of the *framework* when the rules of the *Document Type Association* match a document opened in the **Author** editing mode. The listener is notified about the deactivation when another *framework* is activated for the same document, the user switches to another mode or the editor is closed. A complete description and implementation of `ro.sync.ecss.extensions.api.AuthorExtensionStateListener` can be found in *Implementing an Author Extension State Listener*.

   If *Schema Aware mode* is active in Oxygen XML Editor, all actions that can generate invalid content will be redirected toward the `ro.sync.ecss.extensions.api.AuthorSchemaAwareEditingHandler`. The handler can resolve a specific case, let the default implementation take place, or reject the edit entirely by

throwing an *ro.sync.ecss.extensions.api.InvalidEditException*. The actions that are forwarded to this handler include typing, delete, or paste.

See *Implementing a Schema-Aware Editing Handler Adapter* on page 1215 for more details about this handler.

5. Customizations of the content completion proposals are permitted by creating a schema manager filter extension. The interface that declares the methods used for content completion proposals filtering is *ro.sync.contentcompletion.xml.SchemaManagerFilter*. The filter can be applied on elements, attributes, or on their values. The `createSchemaManagerFilter` method is responsible for creating the content completion filter. A new `SchemaManagerFilter` will be created each time a document matches the rules defined by the *Document Type Association* that contains the filter declaration.

   For example:

```
public SchemaManagerFilter createSchemaManagerFilter() {
     return new SDFSchemaManagerFilter();
}
```

   A detailed presentation of the schema manager filter can be found in the *Configuring a Content Completion Handler* section.

6. The **Author** mode supports link-based navigation between documents and document sections. Therefore, if the document contains elements defined as links to other elements (for example, links based on the `id` attributes), the extension should provide the means to find the referenced content. To do this, an implementation of the *ro.sync.ecss.extensions.api.link.ElementLocatorProvider* interface should be returned by the `createElementLocatorProvider` method. Each time an element pointed by a link needs to be located, the method is invoked.

   For example:

```
public ElementLocatorProvider createElementLocatorProvider() {
     return new DefaultElementLocatorProvider();
}
```

   For more information on how to implement an element locator provider, see the *Configuring a Link Target Element Finder* section.

7. The drag and drop functionality can be extended by implementing the *ro.sync.exml.editor.xmleditor.pageauthor.AuthorDnDListener* interface. Relevant methods from the listener are invoked when the mouse is dragged, moved over, or exits the **Author** editing mode, when the drop action changes, and when the drop occurs. Each method receives the `DropTargetEvent` containing information about the drag and drop operation. The drag and drop extensions are available in **Author** mode for both Oxygen XML Editor Eclipse plugin and standalone application. The **Text** mode corresponding listener is available only for Oxygen XML Editor Eclipse plugin. The methods corresponding to each implementation are: `createAuthorAWTDndListener`, `createTextSWTDndListener`, and `createAuthorSWTDndListener`.

```
public AuthorDnDListener createAuthorAWTDndListener() {
     return new SDFAuthorDndListener();
}
```

   For more details about the **Author** mode drag and drop listeners, see the *Configuring a custom Drag and Drop Listener* section.

8. Another extension that can be included in the bundle is the reference resolver. In our example, the references are represented by the **ref** element and the attribute indicating the referenced resource is **location**. To be able to obtain the content of the referenced resources you will have to implement a Java extension class that implements *ro.sync.ecss.extensions.api.AuthorReferenceResolver*. The method responsible for creating the custom references resolver is `createAuthorReferenceResolver`. The method is called each time a document opened in an **Author** editing mode matches the *Document Type Association* where the extensions bundle is defined. The instantiated references resolver object is kept and used until another extensions bundle corresponding to another document type is activated as result of the detection process.

   For example:

```
public AuthorReferenceResolver createAuthorReferenceResolver() {
     return new ReferencesResolver();
}
```

A more detailed description of the references resolver can be found in the *Configuring a References Resolver* section.

9. To be able to dynamically customize the default CSS styles for a certain *`ro.sync.ecss.extensions.api.node.AuthorNode`*, an implementation of *`ro.sync.ecss.extensions.api.StylesFilter`* can be provided. The extensions bundle method responsible for creating the `StylesFilter` is `createAuthorStylesFilter`. The method is called each time a document opened in an **Author** editing mode matches the *Document Type Association* where the extensions bundle is defined. The instantiated filter object is kept and used until another extensions bundle corresponding to another document type is activated as a result of the detection process.

   For example:

```
public StylesFilter createAuthorStylesFilter() {
     return new SDFStylesFilter();
}
```

   See the *Configuring CSS Styles Filter* section for more details about the styles filter extension.

10. To edit data in custom tabular format, implementations of the *`ro.sync.ecss.extensions.api.AuthorTableCellSpanProvider`* and *`the ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider`* interfaces should be provided. The two methods from the `ExtensionsBundle` specifying these two extension points are `createAuthorTableCellSpanProvider` and `createAuthorTableColumnWidthProvider`.

    For example:

```
public AuthorTableCellSpanProvider createAuthorTableCellSpanProvider() {
     return new TableCellSpanProvider();
}

public AuthorTableColumnWidthProvider
         createAuthorTableColumnWidthProvider() {
     return new TableColumnWidthProvider();
}
```

    The two table information providers are not reused for different tables. The methods are called for each table in the document so new instances should be provided every time. Read more about the cell span and column width information providers in *Configuring a Table Cell Span Provider* and *Configuring a Table Column Width Provider* sections.

    If the functionality related to one of the previous extension point does not need to be modified, then the developed *`ro.sync.ecss.extensions.api.ExtensionsBundle`* should not override the corresponding method and leave the default base implementation to return **null**.

11. An XML vocabulary can contain links to various areas of a document. If the document contains elements defined as links, you can choose to present a more relevant text description for each link. To do this, an implementation of the *`ro.sync.ecss.extensions.api.link.LinkTextResolver`* interface should be returned by the `createLinkTextResolver` method. This implementation is used each time *`the oxy_link-text() function`* is encountered in the CSS styles associated with an element.

    For example:

```
public LinkTextResolver createLinkTextResolver() {
  return new DitaLinkTextResolver();
}
```

    Oxygen XML Editor offers built-in implementations for DITA and DocBook: *`ro.sync.ecss.extensions.dita.link.DitaLinkTextResolver`* and *`ro.sync.ecss.extensions.docbook.link.DocbookLinkTextResolver`* respectively.

12. Pack the compiled class into a *JAR* file.

13. Copy the *JAR* file into your custom *framework* directory (for example, `frameworks/sdf`).

14. Add the *JAR* file to the class path. To do this, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Document Type Association**, select the document type (for example, *SDF*), press the **Edit** button, select the **Classpath** tab, and press the **+ Add** button . In the displayed dialog box, enter the location of the *JAR* file relative to the Oxygen XML Editor `frameworks` folder.

15. Register the Java class by going to the **Extensions** tab. Press the **Choose** button and select the name of the class (for example, `SDFExtensionsBundle`).

    Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

Related Information:

Customizing Elements that Wrap Profiled Content

For each *framework* (document type), you can configure the phrase-type elements that wrap the profiled content by setting a custom `ro.sync.ecss.extensions.api.ProfilingConditionalTextProvider`. This configuration is set by default for DITA and DocBook *frameworks*.

Implementing a Schema-Aware Editing Handler Adapter

The `AuthorSchemaAwareEditingHandlerAdapter` extension point allows you to handle certain **Author** mode actions in various ways. For example, implementing the `AuthorSchemaAwareEditingHandlerAdapter` makes it possible to handle events such as typing, the keyboard delete event at a given offset (using Delete or Backspace keys), delete element tags, delete selection, join elements, or paste fragment. It also makes it possible to improve solutions that are proposed by the paste mechanism in Oxygen XML Editor when pasting content (through the use of *some specific methods*).

How to Implement an `AuthorSchemaAwareEditingHandlerAdapter`

For this handler to be called, the ***Schema Aware Editing** option* must be set to **On** or **Custom** in the ***Schema-Aware** preferences page*. The handler can either resolve a specific case, let the default implementation take place, or reject the edit entirely by throwing an `InvalidEditException`.

To implement your own `AuthorSchemaAwareEditingHandlerAdapter`, follow this procedure:

1. Implement the `ro.sync.ecss.extensions.api.AuthorSchemaAwareEditingHandlerAdapter` extension.
2. To instruct Oxygen XML Editor to use this newly created implementation, *configure an extensions bundle* and return the `AuthorSchemaAwareEditingHandlerAdapter` implementation using the `ro.sync.ecss.extensions.api.ExtensionsBundle.getAuthorSchemaAwareEditingHandlerAdapter()` method.

Example

Typing events can be handled using the `handleTyping` method. For example, the `AuthorSchemaAwareEditingHandler` checks if the schema is not a learned one, was loaded successfully, and if the ***Smart paste and drag and drop** option* is selected. If these conditions are met, the event will be handled.

```
public class AuthorSchemaAwareEditingHandlerAdapter
              extends AuthorSchemaAwareEditingHandler {

/**
 * @see AuthorSchemaAwareEditingHandler#handleTyping
(int, char, ro.sync.ecss.extensions.api.AuthorAccess)
 */
public boolean handleTyping(int offset, char ch, AuthorAccess authorAccess)
throws InvalidEditException {
  boolean handleTyping = false;
  AuthorSchemaManager authorSchemaManager =
authorAccess.getDocumentController().getAuthorSchemaManager();
  if (!authorSchemaManager.isLearnSchema() &&
      !authorSchemaManager.hasLoadingErrors() &&
      authorSchemaManager.getAuthorSchemaAwareOptions().isEnableSmartTyping()) {
    try {
      AuthorDocumentFragment characterFragment =
authorAccess.getDocumentController().createNewDocumentTextFragment
(String.valueOf(ch));
      handleTyping = handleInsertionEvent
(offset, new AuthorDocumentFragment[] {characterFragment}, authorAccess);
    } catch (AuthorOperationException e) {
      throw new InvalidEditException
(e.getMessage(), "Invalid typing event: " + e.getMessage(), e, false);
    }
  }
  return handleTyping;
}
```

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

Methods for Improving the Paste Mechanism

`getAncestorDetectionOptions`

When pasting content in **Author** mode, if the result causes the document to become invalid, Oxygen XML Editor will propose solutions to make it valid. As a possible solution, Oxygen XML Editor might surround the pasted content in a sequence of ancestor elements. This *`getAncestorDetectionOptions`* method allows you to choose which parent elements might be a possible solution.

`canBeReplaced`

Allows you to improve solutions that might be proposed by the paste mechanism when pasting content in Oxygen XML Editor. For example, when pasting an element inside an empty element with the same name, this *`canBeReplaced`* method allows Oxygen XML Editor to replace the empty node rather than pasting it after or before the empty node. The callback could also reject this behavior if, for instance, the replacement node contains attributes.

Related Information:

`AuthorDocumentFragment Class`

Implementing an Edit Properties Handler for Author Mode

The `EditPropertiesHandler` extension point allows you to present a specialized dialog box when the action of double-clicking an element tag is intercepted in **Author** mode. For example, you could use it to present a dialog box that allows the user to editing the properties of an image.

How to Implement an `EditPropertiesHandler`

To implement your own `EditPropertiesHandler`, follow this procedure:

1. Implement the *`ro.sync.ecss.extensions.api.EditPropertiesHandler`* interface.
2. To instruct Oxygen XML Editor to use this newly created implementation, use either of the following methods:
   a. If you have *configured an extensions bundle*, you can return the `EditPropertiesHandler` implementation using the `ro.sync.ecss.extensions.api.ExtensionsBundle.createEditPropertiesHandler()` method.
   b. Specify the `EditPropertiesHandler` in the **Author edit properties handler** individual extension in the ***Extensions** tab* of the ***Document Type** configuration dialog box* for your particular document type.

Example

The following example illustrates an implementation for presenting a simple properties editing dialog box when a user double-clicks an `image` tag in **Author** mode (with tags displayed from the **Tags display mode** drop-down menu):

```
public class CustomEditPropertiesHandler implements EditPropertiesHandler {

  /**
   * @see ro.sync.ecss.extensions.api.Extension#getDescription()
   */
  @Override
  public String getDescription() {
    return "Sample implementation that handles properties for a table element.";
  }

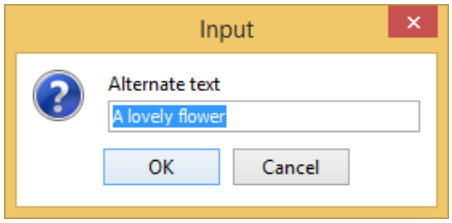
  /**
   * @see ro.sync.ecss.extensions.api.EditPropertiesHandler#canEditProperties
(ro.sync.ecss.extensions.api.node.AuthorNode)
   */
  @Override
  public boolean canEditProperties(AuthorNode authorNode) {
    // If this node is an image element we can edit its properties.
    return "image".equals(authorNode.getDisplayName());
  }

  /**
   * @see ro.sync.ecss.extensions.api.EditPropertiesHandler#editProperties
(ro.sync.ecss.extensions.api.node.AuthorNode,
```

```
 ro.sync.ecss.extensions.api.AuthorAccess)
   */
  @Override
  public void editProperties(AuthorNode authorNode, AuthorAccess authorAccess) {
    // If we receive this call then it surely an image.
    AuthorElement imageElement = (AuthorElement) authorNode;
    String currentValue = "";
    AttrValue altValue = imageElement.getAttribute("alt");
    if (altValue != null) {
      currentValue = altValue.getValue();
    }
    String newValue = JOptionPane.showInputDialog(
        (Component) authorAccess.getWorkspaceAccess().getParentFrame(),
        "Alternate text",
        currentValue);

    if (newValue != null) {
      authorAccess.getDocumentController().setAttribute
("alt", new AttrValue(newValue), imageElement);
    }
  }
}
```

Example result: If a user were to double-click an `image` tag icon (image) in **Author** mode, the following dialog box would be displayed that allows the user to edit the *alternate text* property for the image:



Implementing a Custom Attribute Value Editor

The `CustomAttributeValueEditor` extension point allows you customize the attribute value editing mechanisms in Oxygen XML Editor. It changes the **Browse** button found in the attribute editors to an **Edit** button. When a user clicks that **Edit** button, your custom attribute value editor will be presented.

The **Edit** button can be accessed in the following attribute editors:

- The ***Attributes*** *view in* ***Author*** *mode* (when the **Expand** button is used to reveal an expanded panel).
- The ***Attributes*** *view in* ***Text*** *mode* (when the **Expand** button is used to reveal an expanded panel).
- The ***In-place Attributes Editor*** when invoked in **Author** mode.
- The ***In-place Attributes Editor*** invoked in the **Outline** view when editing in **Author** mode.
- The ***In-place Attributes Editor*** invoked in the **Outline** view when editing in **Text** mode.

How to Implement a `CustomAttributeValueEditor`

To implement your own `CustomAttributeValueEditor`, follow this procedure:

1. Extend the *`ro.sync.ecss.extensions.api.CustomAttributeValueEditor`* abstract class.
2. To instruct Oxygen XML Editor to use this newly created implementation, use either of the following methods:
   a. If you have *configured an extensions bundle*, you can return the `CustomAttributeValueEditor` implementation using the `ro.sync.ecss.extensions.api.ExtensionsBundle.createCustomAttributeValueEditor()` method.
   b. Specify the `CustomAttributeValueEditor` in the **Author custom attribute value editor** individual extension in the ***Extensions*** *tab* of the ***Document Type*** *configuration dialog box* for your particular document type.

Example

The following example creates a very simple custom attribute value editor:

```
/**
```

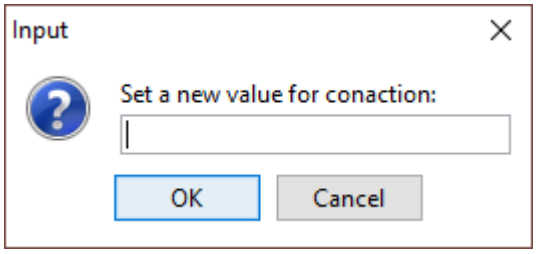
```
 * A custom attribute value editor.
 */
public class MyCustomAttributeValueEditor extends CustomAttributeValueEditor {

  /**
   * @see ro.sync.ecss.extensions.api.Extension#getDescription()
   */
  @Override
  public String getDescription() {
    return "My custom attribute value editor";
  }

  /**
   * @see ro.sync.ecss.extensions.api.CustomAttributeValueEditor#getAttributeValue
             (ro.sync.ecss.extensions.api.EditedAttribute, java.lang.Object)
   */
  @Override
  public String getAttributeValue(EditedAttribute attribute, Object parentComponent)
      throws CancelledByUserException {
    // Show an input dialog for collecting the new value
    return JOptionPane.showInputDialog
                    ("Set a new value for " + attribute.getAttributeQName() + ":");
  }

  /**
   * @see ro.sync.ecss.extensions.api.CustomAttributeValueEditor#shouldHandleAttribute
                      (ro.sync.ecss.extensions.api.EditedAttribute)
   */
  @Override
  public boolean shouldHandleAttribute(EditedAttribute attribute) {
    // Handle all attributes
    return true;
  }
}
```

Example result: If a user were to click the **Edit** button in any of the attribute editors, the following dialog box would be displayed that allows the user to insert a value for the particular attribute:



Implementing an Author Mode Action Event Handler

The `AuthorActionEventHandler` extension point allows you to handle certain **Author** mode actions in a special way. For example, a specific use-case would be if you want to insert new lines when you press **Enter** instead of it opening the *Content Completion Assistant*.

How to Implement an `AuthorActionEventHandler`

To implement your own `AuthorActionEventHandler`, follow this procedure:

1. Implement the *`ro.sync.ecss.extensions.api.AuthorActionEventHandler`* interface.
2. To instruct Oxygen XML Editor to use this newly created implementation, use either of the following methods:
   a. If you have *configured an extensions bundle*, you can return the `AuthorActionEventHandler` implementation using the `ro.sync.ecss.extensions.api.ExtensionsBundle.getAuthorActionEventHandler()` method.
   b. Specify the `AuthorActionEventHandler` in the **Author action event handler** individual extension in the ***Extensions** tab* of the ***Document Type** configuration dialog box* for your particular document type.

Example

The following example illustrates the use-case mentioned in the introduction, that is an implementation for inserting a new line when the user presses **Enter** in **Author** mode. It uses the `canHandleEvent` method to

make sure the insertion will be performed in an element that will preserve the `new-line` character. Then the `handleEvent` method inserts the new line at the current cursor position.

```
public class CustomAuthorActionEventHandler implements AuthorActionEventHandler
{

 /**
  * @see ro.sync.ecss.extensions.api.AuthorActionEventHandler#canHandleEvent
(AuthorAccess, AuthorActionEventType)
  */
  @Override
  public boolean canHandleEvent(AuthorAccess authorAccess,
AuthorActionEventType type) {
    boolean canHandle = false;

    if (type == AuthorActionEventType.ENTER) {
      AuthorDocumentController documentController =
authorAccess.getDocumentController();
    int caretOffset = authorAccess.getEditorAccess().getCaretOffset();
    try {
      AuthorNode nodeAtOffset = documentController.getNodeAtOffset(caretOffset);
      if (nodeAtOffset instanceof AuthorElement) {
        AuthorElement elementAtOffset = (AuthorElement) nodeAtOffset;
        AttrValue xmlSpace = elementAtOffset.getAttribute("xml:space");
        if (xmlSpace != null && xmlSpace.getValue().equals("preserve")) {
          canHandle = true;
        }
      }
    } catch (BadLocationException ex) {
        if (logger.isDebugEnabled()) {
          logger.error(ex.getMessage(), ex);
        }
      }
    }

    return canHandle;
  }

 /**
  * @see ro.sync.ecss.extensions.api.AuthorActionEventHandler#handleEvent
(ro.sync.ecss.extensions.api.AuthorAccess,
ro.sync.ecss.extensions.api.AuthorActionEventHandler.AuthorActionEventType)
  */
  @Override
  public boolean handleEvent(AuthorAccess authorAccess,
 AuthorActionEventType eventType) {
    int caretOffset = authorAccess.getEditorAccess().getCaretOffset();
    // Insert a new line
    authorAccess.getDocumentController().insertText(caretOffset, "\n");
    return true;
  }

  /**
   * @see ro.sync.ecss.extensions.api.Extension#getDescription()
   */
  @Override
  public String getDescription() {
    return "Insert a new line";
  }
}
```

Implementing an Image Decorator for Author Mode

The `AuthorImageDecorator` extension point allows you to add a custom decorator over images in **Author** mode. For example, you could use it to add a message over an image informing the user that they can double-click the image to edit it.

How to Implement an `AuthorImageDecorator`

To implement your own `AuthorImageDecorator`, follow this procedure:

1. Implement the *`ro.sync.ecss.extensions.api.AuthorImageDecorator`* interface.
2. To instruct Oxygen XML Editor to use this newly created implementation, use either of the following methods:
   a. If you have *configured an extensions bundle*, you can return the `AuthorImageDecorator` implementation using the `ro.sync.ecss.extensions.api.ExtensionsBundle.getAuthorImageDecorator()` method.
   b. Specify the `AuthorImageDecorator` in the **Author image decorator** individual extension in the ***Extensions** tab* of the ***Document Type** configuration dialog box* for your particular document type.

Example

The following example illustrates an implementation for presenting a simple message over an image that informs the user that they can double-click the image to edit it:

```
/**
 * Custom Author image decorator for drawing string over images.
 */
public class CustomAuthorImageDecorator extends AuthorImageDecorator {

  /**
   * @see ro.sync.ecss.extensions.api.AuthorImageDecorator#paint
(ro.sync.exml.view.graphics.Graphics, int, int, int, int,
 ro.sync.exml.view.graphics.Rectangle,
 ro.sync.ecss.extensions.api.node.AuthorNode,
 ro.sync.ecss.extensions.api.AuthorAccess, boolean)
   */
  @Override
  public void paint(Graphics g, int x, int y, int imageWidth, int imageHeight,
      Rectangle originalSize, AuthorNode element,
 AuthorAccess authorAccess, boolean wasAnnotated) {
    if ("image".equals(CommonsOperationsUtil.getLocalName(element.getName()))) {
      g.drawString(
          "[Double-click to edit image]",
          // Draw near the top-left corner
          x + 15,
          y + 15);
    }
  }
```

Example result: In the top-left corner of the image, the following message will be displayed: `[Double-click to edit image]`.

Implementing a State Listener for Author Mode

The *`ro.sync.ecss.extensions.api.AuthorExtensionStateListener`* implementation is notified when the **Author** mode extension (where the listener is defined) is activated or deactivated in the document type detection process.

```
import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.AuthorExtensionStateListener;

public class SDFAuthorExtensionStateListener implements
  AuthorExtensionStateListener {
  private AuthorListener sdfAuthorDocumentListener;
  private AuthorMouseListener sdfMouseListener;
  private AuthorCaretListener sdfCaretListener;
  private OptionListener sdfOptionListener;
```

When the association rules of the *framework* (document type) configuration match that of a document opened in the **Author** editing mode, the *activation* event received by this listener should be used to perform custom initializations and to register listeners such as *`ro.sync.ecss.extensions.api.AuthorListener`*, *`ro.sync.ecss.extensions.api.AuthorMouseListener`*, or *`ro.sync.ecss.extensions.api.AuthorCaretListener`*.

```
 public void activated(AuthorAccess authorAccess) {
   // Get the value of the option.
   String option = authorAccess.getOptionsStorage().getOption(
               "sdf.custom.option.key", "");
   // Use the option for some initializations...

   // Add an OptionListener.
   authorAccess.getOptionsStorage().addOptionListener(sdfOptionListener);

   // Add author DocumentListeners.
   sdfAuthorDocumentListener = new SDFAuthorListener();
   authorAccess.getDocumentController().addAuthorListener(
               sdfAuthorDocumentListener);

   // Add MouseListener.
   sdfMouseListener = new SDFAuthorMouseListener();
   authorAccess.getEditorAccess().addAuthorMouseListener(sdfMouseListener);

   // Add CaretListener.
   sdfCaretListener = new SDFAuthorCaretListener();
   authorAccess.getEditorAccess().addAuthorCaretListener(sdfCaretListener);

   // Other custom initializations...

 }
```

The authorAccess parameter received by the `activated` method can be used to gain access to specific **Author** mode actions and informations related to components such as the editor, document, workspace, tables, or the *change tracking* manager.

If options specific to the custom developed *Author Extension* need to be stored or retrieved, a reference to the *`ro.sync.ecss.extensions.api.OptionsStorage`* can be obtained by calling the `getOptionsStorage` method from the authorAccess. The same object can be used to register *`ro.sync.ecss.extensions.api.OptionListener`* listeners. An option listener is registered in relation with an option **key** and will be notified about the value changes of that option.

An `AuthorListener` can be used if events related to the **Author** mode document modifications are of interest. The listener can be added to the *`ro.sync.ecss.extensions.api.AuthorDocumentController`*. A reference to the document controller is returned by the `getDocumentController` method from the authorAccess. The document controller can also be used to perform operations involving document modifications.

To provide access to the **Author** mode component-related functionality and information, the authorAccess has a reference to the *`ro.sync.ecss.extensions.api.access.AuthorEditorAccess`* that can be obtained when calling the `getEditorAccess` method. At this level, `AuthorMouseListener` and `AuthorCaretListener` can be added to provide notification of mouse and cursor events that occur in the **Author** editor mode.

The *deactivation* event is received when another *framework* is activated for the same document, the user switches to another editor mode or the editor is closed. The `deactivate` method is typically used to unregister the listeners previously added on the `activate` method and to perform other actions. For example, options related to the deactivated *Author Extension* can be saved at this point.

```
public void deactivated(AuthorAccess authorAccess) {
  // Store the option.
  authorAccess.getOptionsStorage().setOption(
              "sdf.custom.option.key", optionValue);

  // Remove the OptionListener.
  authorAccess.getOptionsStorage().removeOptionListener(sdfOptionListener);

  // Remove DocumentListeners.
  authorAccess.getDocumentController().removeAuthorListener(
              sdfAuthorDocumentListener);

  // Remove MouseListener.
  authorAccess.getEditorAccess().removeAuthorMouseListener(sdfMouseListener);

  // Remove CaretListener.
  authorAccess.getEditorAccess().removeAuthorCaretListener(sdfCaretListener);

  // Other actions...

}
```

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

Configuring a Content Completion Handler

You can filter or contribute to proposals offered for content completion by implementing the *`ro.sync.contentcompletion.xml.SchemaManagerFilter`* interface.

```
import java.util.List;

import ro.sync.contentcompletion.xml.CIAttribute;
import ro.sync.contentcompletion.xml.CIElement;
import ro.sync.contentcompletion.xml.CIValue;
import ro.sync.contentcompletion.xml.Context;
import ro.sync.contentcompletion.xml.SchemaManagerFilter;
import ro.sync.contentcompletion.xml.WhatAttributesCanGoHereContext;
import ro.sync.contentcompletion.xml.WhatElementsCanGoHereContext;
import ro.sync.contentcompletion.xml.WhatPossibleValuesHasAttributeContext;

public class SDFSchemaManagerFilter implements SchemaManagerFilter {
```

You can implement the various callbacks of the interface either by returning the default values given by Oxygen XML Editor or by contributing to the list of proposals. The filter can be applied on elements, attributes or on their values. Attributes filtering can be implemented using the `filterAttributes` method and changing the default content completion list of *`ro.sync.contentcompletion.xml.CIAttribute`* for the element provided by the

current *ro.sync.contentcompletion.xml.WhatAttributesCanGoHereContext* context. For example, the SDFSchemaManagerFilter checks if the element from the current context is the table element and adds the frame attribute to the table list of attributes.

```
/**
 * Filter attributes of the "table" element.
 */
public List<CIAttribute> filterAttributes(List<CIAttribute> attributes,
    WhatAttributesCanGoHereContext context) {
  // If the element from the current context is the 'table' element add the
  // attribute named 'frame' to the list of default content completion proposals
  if (context != null) {
    ContextElement contextElement = context.getParentElement();
    if ("table".equals(contextElement.getQName())) {
      CIAttribute frameAttribute = new CIAttribute();
      frameAttribute.setName("frame");
      frameAttribute.setRequired(false);
      frameAttribute.setFixed(false);
      frameAttribute.setDefaultValue("void");
      if (attributes == null) {
        attributes = new ArrayList<CIAttribute>();
      }
      attributes.add(frameAttribute);
    }
  }
  return attributes;
}
```

The elements that can be inserted in a specific context can be filtered using the filterElements method. The SDFSchemaManagerFilter uses this method to replace the td child element with the th element when header is the current context element.

```
public List<CIElement> filterElements(List<CIElement> elements,
    WhatElementsCanGoHereContext context) {
  // If the element from the current context is the 'header' element remove the
  // 'td' element from the list of content completion proposals and add the
  // 'th' element.
  if (context != null) {
    Stack<ContextElement> elementStack = context.getElementStack();
    if (elementStack != null) {
      ContextElement contextElement = context.getElementStack().peek();
      if ("header".equals(contextElement.getQName())) {
        if (elements != null) {
          for (Iterator<CIElement> iterator =
elements.iterator(); iterator.hasNext();) {
            CIElement element = iterator.next();
            // Remove the 'td' element
            if ("td".equals(element.getQName())) {
              elements.remove(element);
              break;
            }
          }
        } else {
          elements = new ArrayList<CIElement>();
        }
        // Insert the 'th' element in the list of content completion proposals
        CIElement thElement = new SDFElement();
        thElement.setName("th");
        elements.add(thElement);
      }
    }
  } else {
    // If the given context is null then the given list of content completion
    // elements contains global elements.
  }
  return elements;
}
```

The elements or attributes values can be filtered using the filterElementValues or filterAttributeValues methods.

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

Configuring a Link Target Element Finder

The link target reference finder represents the support for finding references from links that indicate specific elements inside an XML document. This support will only be available if a schema is associated with the document type.

If you do not define a custom link target reference finder, the DefaultElementLocatorProvider implementation will be used by default. The interface that should be implemented for a custom link target reference finder is *ro.sync.ecss.extensions.api.link.ElementLocatorProvider*. As an alternative,

the *ro.sync.ecss.extensions.commons.DefaultElementLocatorProvider* implementation can also be extended.

The used `ElementLocatorProvider` will be queried for an *ElementLocator* when a link location must be determined (when a link is clicked). Then, to find the corresponding (linked) element, the obtained `ElementLocator` will be queried for each element from the document.

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

`DefaultElementLocatorProvider` Implementation

The `DefaultElementLocatorProvider` implementation offers support for the most common types of links:

- Links based on ID attribute values.
- XPointer element() scheme.

The method `getElementLocator` determines what `ElementLocator` should be used. In the default implementation, it checks if the link is an XPointer element() scheme. Otherwise, it assumes it is an ID. A non-null `IDTypeVerifier` will always be provided if a schema is associated with the document type.

The `link` string argument is the *anchor* part of the of the URL that is composed from the value of the link property specified for the link element in the CSS.

```
public ElementLocator getElementLocator(IDTypeVerifier idVerifier,
               String link) {
  ElementLocator elementLocator = null;
  try {
    if(link.startsWith("element(")){
      // xpointer element() scheme
      elementLocator = new XPointerElementLocator(idVerifier, link);
    } else {
      // Locate link element by ID
      elementLocator = new IDElementLocator(idVerifier, link);
    }
  } catch (ElementLocatorException e) {
    logger.warn("Exception when create element locator for link: "
        + link + ". Cause: " + e, e);
  }
  return elementLocator;
}
```

`XPointerElementLocator` Implementation

`XPointerElementLocator` is an implementation of the abstract class *ro.sync.ecss.extensions.api.link.ElementLocator* for links that have one of the following XPointer element() scheme patterns:

element (*elementID*)

Locate the element with the specified ID.

element (*/1/2/3*)

A child sequence appearing alone identifies an element by means of stepwise navigation, which is directed by a sequence of integers separated by slashes (/). Each integer n locates the nth child element of the previously located element.

element (*elementID/3/4*)

A child sequence appearing after a *NCName* identifies an element by means of stepwise navigation, starting from the element located by the given name.

The constructor separates the ID/integers, which are delimited by slashes(/) into a sequence of identifiers (an XPointer path). It will also check that the link has one of the supported patterns of the XPointer element() scheme.

```
public XPointerElementLocator(IDTypeVerifier idVerifier, String link)
                    throws ElementLocatorException {
  super(link);
  this.idVerifier = idVerifier;

  link = link.substring("element(".length(), link.length() - 1);

  StringTokenizer stringTokenizer = new StringTokenizer(link, "/", false);
  xpointerPath = new String[stringTokenizer.countTokens()];
```

```
  int i = 0;
  while (stringTokenizer.hasMoreTokens()) {
    xpointerPath[i] = stringTokenizer.nextToken();
    boolean invalidFormat = false;

    // Empty xpointer component is not supported
    if(xpointerPath[i].length() == 0){
      invalidFormat = true;
    }

    if(i > 0){
      try {
        Integer.parseInt(xpointerPath[i]);
      } catch (NumberFormatException e) {
        invalidFormat = true;
      }
    }

    if(invalidFormat){
      throw new ElementLocatorException(
        "Only the element() scheme is supported when locating XPointer links."
        + "Supported formats: element(elementID), element(/1/2/3),
              element(elemID/2/3/4).");
    }
    i++;
  }

  if(Character.isDigit(xpointerPath[0].charAt(0))){
    // This is the case when xpointer have the following pattern /1/5/7
    xpointerPathDepth = xpointerPath.length;
  } else {
    // This is the case when xpointer starts with an element ID
    xpointerPathDepth = -1;
    startWithElementID  = true;
  }
}
```

The method `startElement` will be invoked at the beginning of every element in the XML document(even when the element is empty). The arguments it takes are

uri

The namespace URI, or the empty string if the element has no namespace URI or if namespace processing is disabled.

localName

Local name of the element.

qName

Qualified name of the element.

atts

Attributes attached to the element. If there are no attributes, this argument will be empty.

The method returns `true` if the processed element is found to be the one indicated by the link.

The `XPointerElementLocator` implementation of the `startElement` will update the depth of the current element and keep the index of the element in its parent. If the `xpointerPath` starts with an element ID then the current element ID is verified to match the specified ID. If this is the case the depth of the XPointer is updated taking into account the depth of the current element.

If the XPointer path depth is the same as the current element depth then the kept indices of the current element path are compared to the indices in the XPointer path. If all of them match then the element has been found.

```
public boolean startElement(String uri, String localName,
        String name, Attr[] atts) {
  boolean linkLocated = false;
  // Increase current element document depth
  startElementDepth ++;

  if (endElementDepth != startElementDepth) {
    // The current element is the first child of the parent
    currentElementIndexStack.push(new Integer(1));
  } else {
    // Another element in the parent element
    currentElementIndexStack.push(new Integer(lastIndexInParent + 1));
  }

  if (startWithElementID) {
    // This the case when xpointer path starts with an element ID.
    String xpointerElement = xpointerPath[0];
    for (int i = 0; i < atts.length; i++) {
```

```
      if(xpointerElement.equals(atts[i].getValue())){
        if(idVerifier.hasIDType(
            localName, uri, atts[i].getQName(), atts[i].getNamespace())){
          xpointerPathDepth = startElementDepth + xpointerPath.length - 1;
          break;
        }
      }
    }
  }

  if (xpointerPathDepth == startElementDepth){
    // check if xpointer path matches with the current element path
    linkLocated = true;
    try {
      int xpointerIdx = xpointerPath.length - 1;
      int stackIdx = currentElementIndexStack.size() - 1;
      int stopIdx = startWithElementID ? 1 : 0;
      while (xpointerIdx >= stopIdx && stackIdx >= 0) {
        int xpointerIndex = Integer.parseInt(xpointerPath[xpointerIdx]);
        int currentElementIndex =
          ((Integer)currentElementIndexStack.get(stackIdx)).intValue();
        if(xpointerIndex != currentElementIndex) {
          linkLocated = false;
          break;
        }

        xpointerIdx--;
        stackIdx--;
      }

    } catch (NumberFormatException e) {
      logger.warn(e,e);
    }
  }
  return linkLocated;
}
```

The method `endElement` will be invoked at the end of every element in the XML document (even when the element is empty).

The `XPointerElementLocator` implementation of the `endElement` updates the depth of the current element path and the index of the element in its parent.

```
public void endElement(String uri, String localName, String name) {
  endElementDepth = startElementDepth;
  startElementDepth --;
  lastIndexInParent = ((Integer)currentElementIndexStack.pop()).intValue();
}
```

`IDElementLocator` Implementation

The `IDElementLocator` is an implementation of the abstract class *`ro.sync.ecss.extensions.api.link.ElementLocator`* for links that use an **ID**.

The constructor only assigns field values and the method `endElement` is empty for this implementation.

The method `startElement` checks each of the element's attribute values and when one matches the link, it considers the element found if one of the following conditions is satisfied:

- the qualified name of the attribute is `xml:id`
- the attribute type is ID

The attribute type is checked with the help of the method `IDTypeVerifier.hasIDType`.

```
public boolean startElement(String uri, String localName,
        String name, Attr[] atts) {
  boolean elementFound = false;
  for (int i = 0; i < atts.length; i++) {
    if (link.equals(atts[i].getValue())) {
      if("xml:id".equals(atts[i].getQName())) {
        // xml:id attribute
        elementFound = true;
      } else {
        // check if attribute has ID type
        String attrLocalName =
          ExtensionUtil.getLocalName(atts[i].getQName());
        String attrUri = atts[i].getNamespace();
        if (idVerifier.hasIDType(localName, uri, attrLocalName, attrUri)) {
          elementFound = true;
        }
      }
    }
  }

  return elementFound;
```

```
}
```

Creating a Customized Link Target Reference Finder

If you need to create a custom link target reference finder you can do so by creating the class that will implement the *`ro.sync.ecss.extensions.api.link.ElementLocatorProvider`* interface. As an alternative, your class could extend *`ro.sync.ecss.extensions.commons.DefaultElementLocatorProvider`*, the default implementation.

Note: The complete source code of the *`ro.sync.ecss.extensions.commons.DefaultElementLocatorProvider`*, *`ro.sync.ecss.extensions.commons.IDElementLocator`* or *`ro.sync.ecss.extensions.commons.XPointerElementLocator`* can be found in the **oxygen-sample-framework** project.

Configuring a Custom Drag and Drop Listener

Sometimes it is useful to perform various operations when certain objects are dropped from outside sources in the editing area. You can choose from three interfaces to implement depending on whether you are using the Eclipse plugin or the standalone version of the application or if you want to add the handler for the **Text** or **Author** modes.

Interfaces for the Drag and Drop Listener

`ro.sync.exml.editor.xmleditor.pageauthor.AuthorDnDListener`

Receives callbacks from the standalone application for Drag And Drop in **Author** mode.

`com.oxygenxml.editor.editors.author.AuthorDnDListener`

Receives callbacks from the Eclipse plugin for Drag And Drop in **Author** mode.

`com.oxygenxml.editor.editors.TextDnDListener`

Receives callbacks from the Eclipse plugin for Drag And Drop in **Text** mode.

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

Configuring a References Resolver

You need to provide a handler for resolving references and obtain the content they reference. In our case the element that has references is **ref** and the attribute indicating the referenced resource is **location**. You will have to implement a Java extension class for obtaining the referenced resources.

1. Create the class `simple.documentation.framework.ReferencesResolver`. This class must implement the *`ro.sync.ecss.extensions.api.AuthorReferenceResolver`* interface.

```
import ro.sync.ecss.extensions.api.AuthorReferenceResolver;
import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.node.AttrValue;
import ro.sync.ecss.extensions.api.node.AuthorElement;
import ro.sync.ecss.extensions.api.node.AuthorNode;

public class ReferencesResolver
      implements AuthorReferenceResolver {
```

2. The `hasReferences` method verifies if the handler considers the node to have references. It takes as argument an `AuthorNode` that represents the node that will be verified. The method will return `true` if the node is considered to have references. In our case, to be a reference the node must be an element with the name *ref* and it must have an attribute named *location*.

```
public boolean hasReferences(AuthorNode node) {
  boolean hasReferences = false;
  if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
    AuthorElement element = (AuthorElement) node;
    if ("ref".equals(element.getLocalName())) {
      AttrValue attrValue = element.getAttribute("location");
      hasReferences = attrValue != null;
    }
  }
  return hasReferences;
}
```

3. The method `getDisplayName` returns the display name of the node that contains the expanded referenced content. It takes as argument an `AuthorNode` that represents the node for which the display name is needed.

The referenced content engine will ask this `AuthorReferenceResolver` implementation for the display name for each node that is considered a reference. In our case, the display name is the value of the *location* attribute from the *ref* element.

```
public String getDisplayName(AuthorNode node) {
  String displayName = "ref-fragment";
  if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
    AuthorElement element = (AuthorElement) node;
    if ("ref".equals(element.getLocalName())) {
      AttrValue attrValue = element.getAttribute("location");
      if (attrValue != null) {
        displayName = attrValue.getValue();
      }
    }
  }
  return displayName;
}
```

4. The method `resolveReference` resolves the reference of the node and returns a `SAXSource` with the parser and its input source. It takes as arguments an `AuthorNode` that represents the node for which the reference needs resolving, the *systemID* of the node, the `AuthorAccess` with access methods to the **Author** mode data model and a SAX `EntityResolver` that resolves resources that are already opened in another editor or resolve resources through the *XML Catalog*. In the implementation you need to resolve the reference relative to the *systemID*, and create a parser and an input source over the resolved reference.

```
public SAXSource resolveReference(
    AuthorNode node,
    String systemID,
    AuthorAccess authorAccess,
    EntityResolver  entityResolver) {
  SAXSource saxSource = null;

  if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
    AuthorElement element = (AuthorElement) node;
    if ("ref".equals(element.getLocalName())) {
      AttrValue attrValue = element.getAttribute("location");
      if (attrValue != null) {
        String attrStringVal = attrValue.getValue();
        try {
          URL absoluteUrl = new URL(new URL(systemID),
          authorAccess.getUtilAccess().correctURL(attrStringVal));

          InputSource inputSource = entityResolver.resolveEntity(null,
              absoluteUrl.toString());
          if(inputSource == null) {
            inputSource = new InputSource(absoluteUrl.toString());
          }

          XMLReader xmlReader = authorAccess.newNonValidatingXMLReader();
          xmlReader.setEntityResolver(entityResolver);

          saxSource = new SAXSource(xmlReader, inputSource);
        } catch (MalformedURLException e) {
          logger.error(e, e);
        } catch (SAXException e) {
          logger.error(e, e);
        } catch (IOException e) {
          logger.error(e, e);
        }
      }
    }
  }

  return saxSource;
}
```

5. The method `getReferenceUniqueID` should return a unique identifier for the node reference. The unique identifier is used to avoid resolving the references recursively. The method takes as argument an `AuthorNode` that represents the node with the reference. In the implementation the unique identifier is the value of the *location* attribute from the *ref* element.

```
public String getReferenceUniqueID(AuthorNode node) {
  String id = null;
  if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
    AuthorElement element = (AuthorElement) node;
    if ("ref".equals(element.getLocalName())) {
      AttrValue attrValue = element.getAttribute("location");
      if (attrValue != null) {
        id = attrValue.getValue();
      }
    }
  }
  return id;
}
```

6. The method `getReferenceSystemID`should return the *systemID* of the referenced content. It takes as arguments an `AuthorNode` that represents the node with the reference and the `AuthorAccess` with access methods to the **Author** mode data model. In the implementation you use the value of the *location* attribute from the *ref* element and resolve it relatively to the XML base URL of the node.

```
public String getReferenceSystemID(AuthorNode node,
                                   AuthorAccess authorAccess) {
  String systemID = null;
  if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
    AuthorElement element = (AuthorElement) node;
    if ("ref".equals(element.getLocalName())) {
      AttrValue attrValue = element.getAttribute("location");
      if (attrValue != null) {
        String attrStringVal = attrValue.getValue();
        try {
          URL absoluteUrl = new URL(node.getXMLBaseURL(),
              authorAccess.getUtilAccess().correctURL(attrStringVal));
          systemID = absoluteUrl.toString();
        } catch (MalformedURLException e) {
          logger.error(e, e);
        }
      }
    }
  }
  return systemID;
}
```

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

In the listing below, the XML document contains the **ref** element:

```
<ref location="referenced.xml">Reference</ref>
```

When no reference resolver is specified, the reference has the following layout:

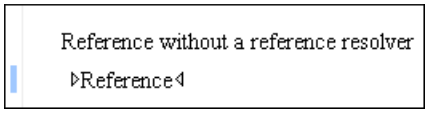


Figure 577: Reference with no specified reference resolver

When the above implementation is configured, the reference has the expected layout:

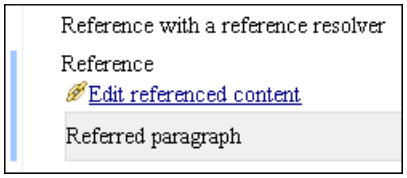


Figure 578: Reference with reference resolver

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

Configuring CSS Styles Filter

You can modify the CSS styles for each *`ro.sync.ecss.extensions.api.node.AuthorNode`* rendered in the **Author** mode using an implementation of *`ro.sync.ecss.extensions.api.StylesFilter`*. You can implement the various callbacks of the interface either by returning the default value given by Oxygen XML Editor or by contributing to the value. The received styles *`ro.sync.ecss.css.Styles`* can be processed and values can be overwritten with your own. For example, you can override the `KEY_BACKGROUND_COLOR` style to return your own implementation of *`ro.sync.exml.view.graphics.Color`* or override the `KEY_FONT` style to return your own implementation of *`ro.sync.exml.view.graphics.Font`*.

For instance, in our simple document example the filter can change the value of the `KEY_FONT` property for the `table` element:

```
package simple.documentation.framework;

import ro.sync.ecss.css.Styles;
```

```
import ro.sync.ecss.extensions.api.StylesFilter;
import ro.sync.ecss.extensions.api.node.AuthorNode;
import ro.sync.exml.view.graphics.Font;

public class SDFStylesFilter implements StylesFilter {

     public Styles filter(Styles styles, AuthorNode authorNode) {
         if (AuthorNode.NODE_TYPE_ELEMENT == authorNode.getType()
           && "table".equals(authorNode.getName())) {
           styles.setProperty(Styles.KEY_FONT, new Font(null, Font.BOLD, 12));
         }
         return styles;
     }
}
```

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

Configuring Tables

There are standard CSS properties used to indicate what elements are tables, table rows and table cells. What CSS is missing is the possibility to indicate the cell spanning, row separators or the column widths. Oxygen XML Editor offers support for adding extensions to solve these problems.

The table in this example is a simple one. The header must be formatted in a different way than the ordinary rows, so it will have a background color.

```
table{
    display:table;
    border:1px solid navy;
    margin:1em;
    max-width:1000px;
    min-width:150px;
}

table[width]{
  width:attr(width, length);
}

tr, header{
    display:table-row;
}

header{
    background-color: silver;
    color:inherit
}

td{
  display:table-cell;
  border:1px solid navy;
  padding:1em;
}
```

Since in the *schema*, the `td` tag has the attributes **row_span** and **column_span** that are not automatically recognized by Oxygen XML Editor, a Java extension will be implemented that will provide information about the cell spanning. See the section *Configuring a Table Cell Span Provider*.

The column widths are specified by the attributes **width** of the elements `customcol` that are not automatically recognized by Oxygen XML Editor. It is necessary to implement a Java extension that will provide information about the column widths. For more information, see *Configuring a Table Column Width Provider*.

The table from our example does not make use of the attributes `colsep` and `rowsep` (which are automatically recognized) but we still want the rows to be separated by horizontal lines. It is necessary to implement a Java extension that will provide information about the row and column separators. For more information, see *Configuring a Table Cell Row and Column Separator Provider* on page 1235.

Configuring a Table Column Width Provider

In a custom *framework*, the `table` element as well as the table columns can have specified widths. For these widths to be considered by **Author** mode we need to provide the means to determine them. As explained in *Configuring Tables* on page 1229, if you use the table element attribute **width** Oxygen XML Editor can determine the table width automatically. In this example the table has `col` elements with **width** attributes that are not recognized by default. You will need to implement a Java extension class to determine the column widths.

1. Create the class `simple.documentation.framework.TableColumnWidthProvider`. This class must implement the *`ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider`* interface.

```
import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.AuthorOperationException;
import ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider;
import ro.sync.ecss.extensions.api.WidthRepresentation;
import ro.sync.ecss.extensions.api.node.AuthorElement;

public class TableColumnWidthProvider
      implements AuthorTableColumnWidthProvider {
```

2. Method `init` is taking as argument an *`ro.sync.ecss.extensions.api.node.AuthorElement`* that represents the XML `table` element. In our case the column widths are specified in `col` elements from the `table` element. In such cases you must collect the span information by analyzing the `table` element.

```
public void init(AuthorElement tableElement) {
 this.tableElement = tableElement;
 AuthorElement[] colChildren = tableElement.getElementsByLocalName("customcol");
    if (colChildren != null && colChildren.length > 0) {
 for (int i = 0; i < colChildren.length; i++) {
   AuthorElement colChild = colChildren[i];
     if (i == 0) {
  colsStartOffset = colChild.getStartOffset();
 }
     if (i == colChildren.length - 1) {
  colsEndOffset = colChild.getEndOffset();
 }
     // Determine the 'width' for this col.
     AttrValue colWidthAttribute = colChild.getAttribute("width");
     String colWidth = null;
     if (colWidthAttribute != null) {
       colWidth = colWidthAttribute.getValue();
           // Add WidthRepresentation objects for the columns this 'customcol'
     //  specification spans over.
  colWidthSpecs.add(new WidthRepresentation(colWidth, true));
 }
     }
   }
}
```

3. The method `isTableAcceptingWidth` should check if the table cells are `td`.

```
public boolean isTableAcceptingWidth(String tableCellsTagName) {
  return "td".equals(tableCellsTagName);
}
```

4. The method `isTableAndColumnsResizable` should check if the table cells are `td`. This method determines if the table and its columns can be resized by dragging the edge of a column.

```
public boolean isTableAndColumnsResizable(String tableCellsTagName) {
  return "td".equals(tableCellsTagName);
}
```

5. Methods `getTableWidth` and `getCellWidth` are used to determine the table and column width. The table layout engine will ask this *`ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider`* implementation what is the table width for each table element and the cell width for each cell element from the table that was marked as cell in the CSS using the property `display:table-cell`. The implementation is simple and just parses the value of the **width** attribute. The methods must return `null` for the tables / cells that do not have a specified width.

```
 public WidthRepresentation getTableWidth(String tableCellsTagName) {
  WidthRepresentation toReturn = null;
  if (tableElement != null && "td".equals(tableCellsTagName)) {
   AttrValue widthAttr = tableElement.getAttribute("width");
   if (widthAttr != null) {
    String width = widthAttr.getValue();
    if (width != null) {
     toReturn = new WidthRepresentation(width, true);
    }
   }
  }
  return toReturn;
 }
```

```
        public List<WidthRepresentation>
getCellWidth(AuthorElement cellElement, int colNumberStart,
   int colSpan) {
  List<WidthRepresentation> toReturn = null;
  int size = colWidthSpecs.size();
  if (size >= colNumberStart && size >= colNumberStart + colSpan) {
   toReturn = new ArrayList<WidthRepresentation>(colSpan);
```

```
    for (int i = colNumberStart; i < colNumberStart + colSpan; i ++) {
     // Add the column widths
     toReturn.add(colWidthSpecs.get(i));
    }
   }
   return toReturn;
  }
```

6. Methods `commitTableWidthModification` and `commitColumnWidthModifications` are used to commit changes made to the width of the table or its columns when using the mouse drag gestures.

```
  public void commitTableWidthModification
 (AuthorDocumentController authorDocumentController,
  int newTableWidth, String tableCellsTagName) throws AuthorOperationException {
  if ("td".equals(tableCellsTagName)) {
   if (newTableWidth > 0) {
    if (tableElement != null) {
     String newWidth = String.valueOf(newTableWidth);

     authorDocumentController.setAttribute(
       "width",
       new AttrValue(newWidth),
       tableElement);
    } else {
  throw new AuthorOperationException("Cannot find the table element.");
     }
    }
   }
  }
```

```
 public void commitColumnWidthModifications
  (AuthorDocumentController authorDocumentController,
    WidthRepresentation[] colWidths, String tableCellsTagName)
  throws AuthorOperationException {
    if ("td".equals(tableCellsTagName)) {
      if (colWidths != null && tableElement != null) {
  if (colsStartOffset >= 0 && colsEndOffset >= 0
 && colsStartOffset < colsEndOffset) {
   authorDocumentController.delete(colsStartOffset,
     colsEndOffset);
   }
   String xmlFragment = createXMLFragment(colWidths);
    int offset = -1;
   AuthorElement[] header = tableElement.getElementsByLocalName("header");
    if (header != null && header.length > 0) {
     // Insert the cols elements before the 'header' element
     offset = header[0].getStartOffset();
    }
    if (offset == -1) {
   throw new AuthorOperationException("No valid offset to insert column width");
     }
     authorDocumentController.insertXMLFragment(xmlFragment, offset);
    }
   }
  }

  private String createXMLFragment(WidthRepresentation[] widthRepresentations) {
   StringBuffer fragment = new StringBuffer();
   String ns = tableElement.getNamespace();
   for (int i = 0; i < widthRepresentations.length; i++) {
    WidthRepresentation width = widthRepresentations[i];
    fragment.append("<customcol");
    String strRepresentation = width.getWidthRepresentation();
    if (strRepresentation != null) {
     fragment.append(" width=\"" + width.getWidthRepresentation() + "\"");
    }
    if (ns != null && ns.length() > 0) {
     fragment.append(" xmlns=\"" + ns + "\"");
    }
    fragment.append("/>");
   }
   return fragment.toString();
  }
```

7. The following three methods are used to determine what type of column width specifications the table column width provider support. In our case all types of specifications are allowed:

```
  public boolean isAcceptingFixedColumnWidths(String tableCellsTagName) {
   return true;
  }

  public boolean isAcceptingPercentageColumnWidths(String tableCellsTagName) {
   return true;
  }

  public boolean isAcceptingProportionalColumnWidths(String tableCellsTagName) {
   return true;
  }
```

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

In the listing below, the XML document contains the table element:

```
<table width="300">
    <customcol width="50.0px"/>
    <customcol width="1*"/>
    <customcol width="2*"/>
    <customcol width="20%"/>
    <header>
        <td>C1</td>
        <td>C2</td>
        <td>C3</td>
        <td>C4</td>
    </header>
    <tr>
        <td>cs=1, rs=1</td>
        <td>cs=1, rs=1</td>
        <td row_span="2">cs=1, rs=2</td>
        <td row_span="3">cs=1, rs=3</td>
    </tr>
    <tr>
        <td>cs=1, rs=1</td>
        <td>cs=1, rs=1</td>
    </tr>
    <tr>
        <td column_span="3">cs=3, rs=1</td>
    </tr>
</table>
```

When no table column width provider is specified, the table has the following layout:

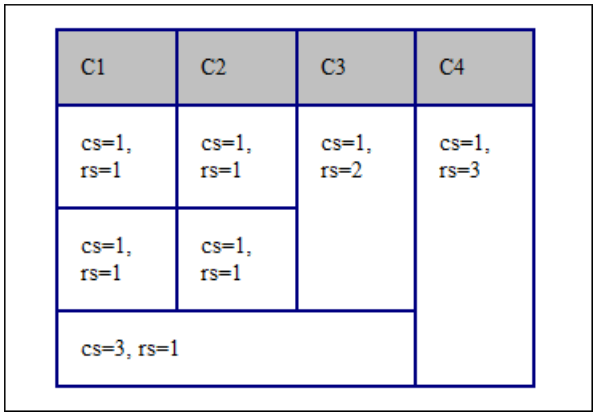


Figure 579: Table layout when no column width provider is specified

When the above implementation is configured, the table has the correct layout:

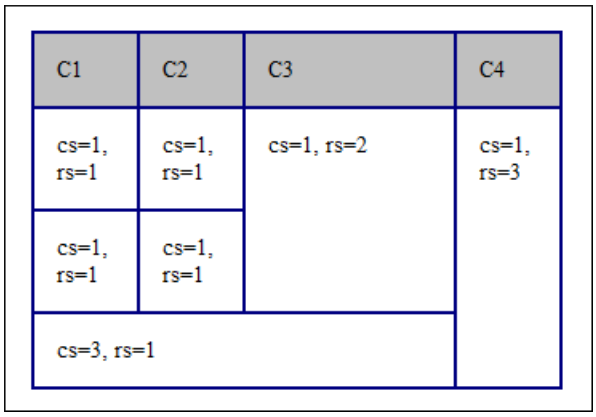


Figure 580: Columns with custom widths

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

Configuring a Table Cell Span Provider

In a custom *framework*, the `table` element can have cells that span over multiple columns and rows. As explained in *Configuring Tables* on page 1229, you need to indicate Oxygen XML Editor a method to determine the cell spanning. If you use the cell element attributes `rowspan` and `colspan` or `rows` and `cols`, Oxygen XML Editor can determine the cell spanning automatically. In our example the `td` element uses the attributes `row_span` and `column_span` that are not recognized by default. You will need to implement a Java extension class for defining the cell spanning.

1. Create the class `simple.documentation.framework.TableCellSpanProvider`. This class must implement the *`ro.sync.ecss.extensions.api.AuthorTableCellSpanProvider`* interface.

```
import ro.sync.ecss.extensions.api.AuthorTableCellSpanProvider;
import ro.sync.ecss.extensions.api.node.AttrValue;
import ro.sync.ecss.extensions.api.node.AuthorElement;

public class TableCellSpanProvider
      implements AuthorTableCellSpanProvider {
```

2. The `init` method is taking as argument the *`ro.sync.ecss.extensions.api.node.AuthorElement`* that represents the XML `table` element. In our case the cell span is specified for each of the cells so you leave this method empty. However, there are cases (such as the CALS table model) when the cell spanning is specified in the `table` element. In such cases, you must collect the span information by analyzing the `table` element.

```
public void init(AuthorElement table) {
}
```

3. The `getColSpan` method is taking as argument the table cell. The table layout engine will ask this `AuthorTableSpanSupport` implementation what is the column span and the row span for each XML element from the table that was marked as cell in the CSS using the property `display:table-cell`. The implementation is simple and just parses the value of **column_span** attribute. The method must return `null` for all the cells that do not change the span specification.

```
public Integer getColSpan(AuthorElement cell) {
  Integer colSpan = null;

  AttrValue attrValue = cell.getAttribute("column_span");
  if(attrValue != null) {
    // The attribute was found.
    String cs = attrValue.getValue();
    if(cs != null) {
      try {
        colSpan = new Integer(cs);
      } catch (NumberFormatException ex) {
        // The attribute value was not a number.
      }
    }
  }
  return colSpan;
}
```

4. The row span is determined in a similar manner:

```
public Integer getRowSpan(AuthorElement cell) {
  Integer rowSpan = null;

  AttrValue attrValue = cell.getAttribute("row_span");
  if(attrValue != null) {
    // The attribute was found.
    String rs = attrValue.getValue();
    if(rs != null) {
      try {
        rowSpan = new Integer(rs);
      } catch (NumberFormatException ex) {
        // The attribute value was not a number.
      }
    }
  }
  return rowSpan;
}
```

5. The method `hasColumnSpecifications` always returns `true` considering column specifications always available.

```
public boolean hasColumnSpecifications(AuthorElement tableElement) {
  return true;
}
```

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

6. In the listing below, the XML document contains the table element:

```
<table>
    <header>
        <td>C1</td>
        <td>C2</td>
        <td>C3</td>
        <td>C4</td>
    </header>
    <tr>
        <td>cs=1, rs=1</td>
        <td column_span="2" row_span="2">cs=2, rs=2</td>
        <td row_span="3">cs=1, rs=3</td>
    </tr>
    <tr>
        <td>cs=1, rs=1</td>
    </tr>
    <tr>
        <td column_span="3">cs=3, rs=1</td>
    </tr>
</table>
```

When no table cell span provider is specified, the table has the following layout:

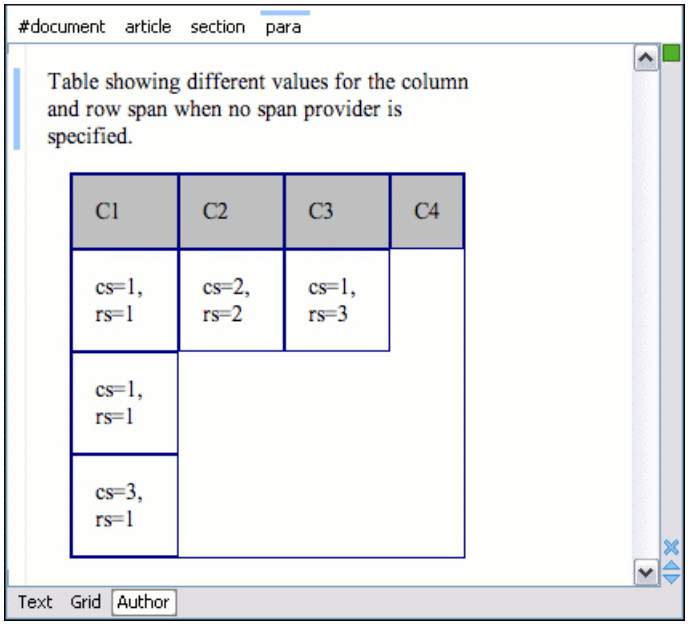


Figure 581: Table layout when no cell span provider is specified

When the above implementation is configured, the table has the correct layout:

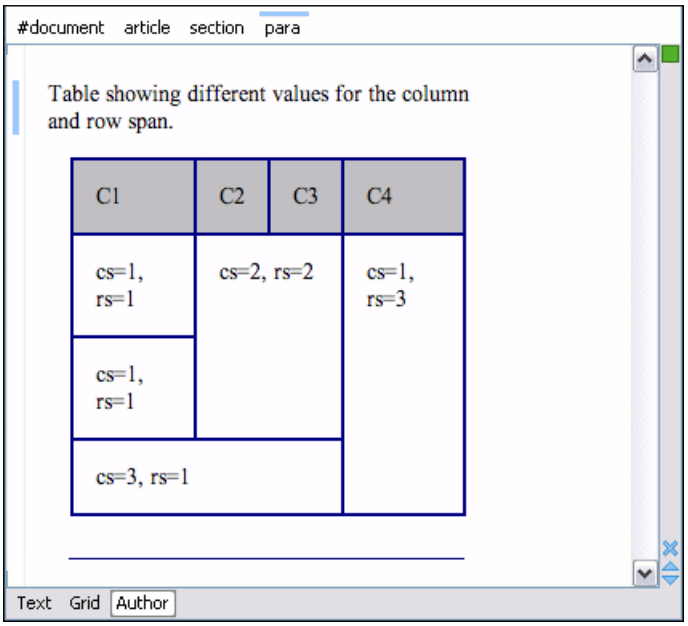


Figure 582: Cells spanning multiple rows and columns.

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

Configuring a Table Cell Row and Column Separator Provider

In a custom *framework*, the `table` element has separators between rows. As explained in *Configuring Tables* on page 1229, you need to indicate a method to determine the way rows and columns are separated. If you use the `rowsep` and `colsep` cell element attributes, or your table is conforming to the CALS table model, Oxygen XML Editor can determine the cell separators. Even if there are no attributes that define the separators, you can still force a separator between rows by implementing a Java extension.

1. Create the class `simple.documentation.framework.TableCellSepProvider`. This class must implement the *`ro.sync.ecss.extensions.api.AuthorTableCellSepProvider`* interface.

```
import ro.sync.ecss.extensions.api.AuthorTableCellSepProvider;
import ro.sync.ecss.extensions.api.node.AuthorElement;

public class TableCellSepProvider implements AuthorTableCellSepProvider{
```

2. The `init` method is taking as argument the *`ro.sync.ecss.extensions.api.node.AuthorElement`* that represents the XML `table` element. In our case the separator information is implicit, it does not depend on the current table, so you leave this method empty. However, there are cases (such as the CALS table model) when the cell separators are specified in the `table` element. In such cases, you should initialize your provider based on the given argument.

```
public void init(AuthorElement table) {
}
```

3. The `getColSep` method is taking as argument the table cell. The table layout engine will ask this `AuthorTableCellSepProvider` implementation if there is a column separator for each XML element from the table that was marked as cell in the CSS using the property `display:table-cell`. In our case we choose to return **false** since we do not need column separators.

```
  /**
   * @return false - No column separator at the right of the cell.
   */
  @Override
  public boolean getColSep(AuthorElement cellElement, int columnIndex) {
    return false;
  }
```

4. The row separators are determined in a similar manner. This time the method returns **true**, forcing a separator between the rows.

```
  /**
```

```
   * @return true - A row separator below each cell.
   */
  @Override
  public boolean getRowSep(AuthorElement cellElement, int columnIndex) {
    return true;
  }
```

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

5. In the example below, the XML document contains the table element:

```
<table>
     <header>
       <td>H1</td>
       <td>H2</td>
       <td>H3</td>
       <td>H4</td>
     </header>
     <tr>
       <td>C11</td>
       <td>C12</td>
       <td>C13</td>
       <td>C14</td>
     </tr>
     <tr>
       <td>C21</td>
       <td>C22</td>
       <td>C23</td>
       <td>C24</td>
     </tr>
     <tr>
       <td>C31</td>
       <td>C32</td>
       <td>C33</td>
       <td>C34</td>
     </tr>
   </table>
```

When the borders for the `td` element are removed from the CSS, the row separators become visible:

H1	H2	H3	H4
C11	C12	C13	C14
C21	C22	C23	C24
C31	C32	C33	C34

Figure 583: Row separators provided by the Java implementation.

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

Configuring a Unique Attributes Recognizer

The *`ro.sync.ecss.extensions.api.UniqueAttributesRecognizer`* interface can be implemented if you want to provide for your *framework* the following features:

- **Automatic ID generation** - You can automatically generate unique IDs for newly inserted elements. Implementations are already available for the DITA and DocBook *frameworks*. The following methods can be implemented to accomplish this: `assignUniqueIDs(int startOffset, int endOffset)`, `isAutoIDGenerationActive()`
- **Avoiding copying unique attributes when "Split" is called inside an element** - You can split the current *block element* by pressing the "Enter" key and then choosing "Split". This is a very useful way to create new paragraphs, for example. All attributes are by default copied on the new element but if those attributes are IDs you sometimes want to avoid creating validation errors in the editor. Implementing the following method, you can decide whether or not an attribute should be copied during the split: `boolean copyAttributeOnSplit(String attrQName, AuthorElement element)`

Tip: The *ro.sync.ecss.extensions.commons.id.DefaultUniqueAttributesRecognizer* class is an implementation of the interface that can be extended by your customization to provide easy assignation of IDs in your *framework*. You can also check out the DITA and DocBook implementations of *ro.sync.ecss.extensions.api.UniqueAttributesRecognizer* to see how they were implemented and connected to the extensions bundle.

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

Configuring an XML Node Renderer Customizer

You can use this API extension to customize the way an XML node is rendered in the ***Outline** view* in **Author** mode, *breadcrumb navigation bar* in **Author** mode, ***Outline** view* in **Text** mode, *Content Completion Assistant* window, or ***DITA Maps Manager** view*.

Note: Oxygen XML Editor uses `XMLNodeRendererCustomizer` implementations for the following *frameworks*: DITA, DITA Map, DocBook 4, DocBook 5, TEI, XHTML, XSLT, and XML Schema.

There are two methods to provide an implementation of *ro.sync.exml.workspace.api.node.customizer.XMLNodeRendererCustomizer*:

- As a part of a bundle, returning it from the `createXMLNodeCustomizer()` method of the *ExtensionsBundle* associated with your document type in the ***Document type** configuration dialog box* (**Extensions bundle** field in the **Extensions** tab).
- As an individual extension, associated with your document type in the ***Document type** configuration dialog box* (**XML node renderer customizer** field in the **Individual extensions** section of the **Extensions** tab).

Support for Retina/HiDPI Displays

To support Retina or HiDPI displays, the icons provided by the `XMLNodeRendererCustomizer` should be backed up by a copy of larger size using the proper *Retina/HiDPI naming convention*.

For example, for the `title` element, if the `XMLNodeRendererCustomizer` returns the path `${framework}/images/myImg.png`, then to support Retina images with a scaling factor of 2, an extra file (`myImg@2x.png`)should be added to the same images directory (`${framework}/images/myImg@2x.png`). If the higher resolution icon (the *@2x* file) does not exist, the *normal* icon is scaled and used instead.

For more information about using Retina/HiDPI images, refer to the *Using Retina/HiDPI Images in Author Mode* section.

Note: The complete source code for *framework* customization examples can be found in the **oxygen-sample-framework** module of the *Oxygen SDK* , available as a Maven archetype *on the Oxygen XML Editor website*.

Related Information:

Customizing the Rendering of Elements on page 1248

Adding Custom Persistent Highlights

The *Author API* includes a class that allows you to create or remove custom persistent highlights, set new properties for the highlights, and customize their appearance. An example of a possible use case would be if you want to implement your own way of editing review comments. The custom persistent highlights get serialized in the XML document as processing instructions, with the following format:

```
<?oxy_custom_start prop1="val1"....?> xml content <?oxy_custom_end?>
```

This functionality is available through the *AuthorPersistentHighlighter* class that is accessible through the *AuthorEditorAccess#getPersistentHighlighter()* method.

For more information, see the JavaDoc details for this class at *https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/ecss/extensions/api/highlights/AuthorPersistentHighlighter.html*.

Customizing the Main CSS of a Framework

The easiest way to customize the *main* CSS stylesheet of a *framework* is to create a new stylesheet, save it as an *alternate* CSS file that will be applied as an additional layer to the *main* CSS, and then select it from the **Styles** drop-down menu in **Author** mode.

For example, suppose that you want to customize the *main* CSS for DITA documents. To do this, follow these steps:

1. First, create a new CSS stylesheet and save it in the `[OXYGEN_INSTALL_DIR]/frameworks/dita/css/edit` folder (where the default *main* stylesheet named `style-basic.css` is located).
2. Edit the DITA *framework* and go to the ***CSS** subtab*:
   a) *Open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Document Type Association**.
   b) Select the DITA document type and press the **Edit** button.

   Tip: If you do not have write permissions to modify the document type, use the **Extend** button to create an extension of the *framework*.

   a) Go to the **CSS** subtab of the **Author** tab.

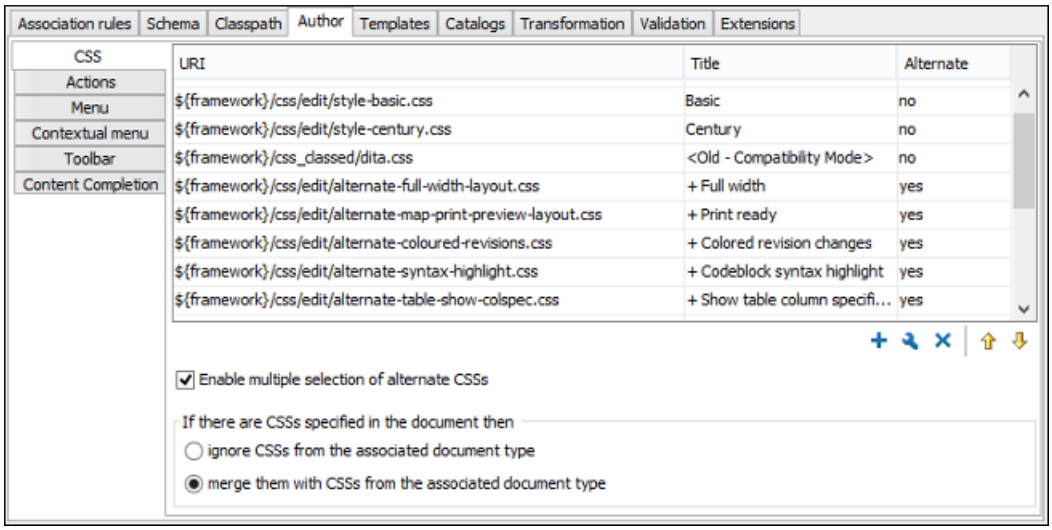


Figure 584: CSS Subtab of the Document Type Association Author Tab

3. Add the new stylesheet as an *alternate* CSS stylesheet:
   a) Click the **Add** button to open a dialog box that allows you to specify the URI and Title for your newly created stylesheet.
   b) Select the **Alternate** option to define it as an *alternate* stylesheet that will applied as an additional layer to the *main* CSS.

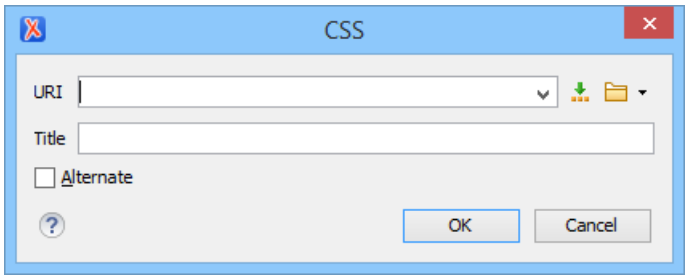


Figure 585: Add CSS Stylesheet Dialog Box

4. Press **OK** in all the dialog boxes to validate the changes.
5. Select your newly created CSS stylesheet from the **Styles** drop-down menu on the toolbar in **Author** mode. You can now edit DITA documents based on the new CSS stylesheet. You can also edit the new CSS stylesheet itself and see its effects on rendering DITA documents in the **Author** mode by using the **Refresh** action that is available on the **Author** toolbar and in the **DITA** menu.

Related Information:

Configuring and Managing Multiple CSS Styles on page 1256

Sharing a Framework (Document Type)

Oxygen XML Editor allows you to share customizations of a specific XML document type by *creating your own framework* in the ***Document Type Association*** *preferences page*.

A *framework* (document type) can be shared with other authors by using any of the following methods:

Save it in a Custom Folder

To share your customized *framework* with other members of your team, you can save it to a separate folder inside the `[OXYGEN_INSTALL_DIR]/frameworks` directory by following these steps:

Important: For this approach to work, the application must be installed in a folder with full write access.

1. Go to `[OXYGEN_INSTALL_DIR]/frameworks` and create a directory for your new *framework* (for example, `custom_framework`). This directory will contain the resources for your customized *framework*. See the **DocBook** *framework* structure from `[OXYGEN_INSTALL_DIR]/frameworks/docbook` as an example.
2. Create your custom document type (*framework*) and specify the `custom_framework` directory for the **External** storage option.
3. Configure the custom document type according to your needs. Take special care to make all file references relative to the `[OXYGEN_INSTALL_DIR]/frameworks` directory by using the *`${frameworks}` editor variable*. See the *Author Mode Customization* section for more details on creating and configuring a new document type (*framework*).
4. Add any additional resources (CSS files, new file templates, schemas used for validation, catalogs, etc.) to the directory you created in step 1.
5. After completing your customizations in the ***Document Type Association*** *preferences page*, you should have a new configuration file saved in: `[OXYGEN_INSTALL_DIR]/frameworks/custom_framework/custom.framework`.
6. To share the new *framework* directory with other users, have them copy it to their `[OXYGEN_INSTALL_DIR]/frameworks` directory. The new *framework* will be available in the list of document types when Oxygen XML Editor starts.

   Note: If you have a `frameworks` directory stored on your local drive, you can also go to the ***Document Type Association*** > ***Locations*** *preferences page* and add your `frameworks` directory in the **Additional frameworks directories** list.

Save it at Project Level

You can also share customized *frameworks* by saving it at *project level* in the ***Document Type Association*** *preferences page*. To do so, follow these steps:

1. On your local drive, create a directory with full write access. This directory will contain the resources for your customized *framework* and the project file.
2. Start Oxygen XML Editor, go to the ***Project*** *view* and create a project. Save it in the newly created directory.
3. In the ***Document Type Association*** *preferences page*, select ***Project Options*** at the bottom of the page.
4. Create your custom document type (*framework*) and use the default **Internal** storage option. It will actually be saved in the project file (`.xpr`).
5. Configure the custom document type according to your needs. Take special care to make all file references relative to the project directory by using the *`${pd}` editor variable*. See the *Advanced Framework Customization* on page 1169 section for more details on creating and configuring a new document type (*framework*).
6. Add any additional resources (CSS files, new file templates, schemas used for validation, catalogs, etc.) to your project.
7. You can then share the new project directory with other team members. When they open the customized project file in the ***Project*** *view*, the new *framework* becomes available in the list of document types.

Deploy it as an Add-on

You can also share your customized *framework* by deploying it as and add-on. To do so, follow the procedures in *Packing and Deploying Plugins or Frameworks as Add-ons* on page 1204.

Related Information:

Extending and Sharing a Framework (Document Type) on page 1240

Extending and Sharing a Framework (Document Type)

You can extend a predefined, built-in *framework* (document type), such as DITA or DocBook, using the ***Document Type Association*** *preferences page*, make modifications to it, and then share the extension with your team.

Extending a Framework

For the purpose of providing specific instructions for sharing an extended *framework*, suppose that you want to share an extension of the **DITA** *framework* in which you have removed certain elements from the content completion list. The following steps describe how you can create an extended *framework* that can be shared with others:

1. In a location where you have full write access, create a folder structure similar to this: `custom_frameworks/dita-extension`.
2. *Open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to ***Document Type Association > Locations***. In this preferences page, add the path to your `custom_frameworks` folder in the **Additional frameworks directories** list.
3. Go to the ***Document Type Association*** *preferences page* and select the **DITA** *framework* configuration and use the **Extend** button to create an extension for it.
4. Give the extension an appropriate name (for example, `DITA - Custom`), select **External** for the **Storage** option, and specify an appropriate path (for example, `path/to/.../custom_frameworks/dita-extension/dita-extension.framework`).
5. Make your changes to the extension. For example, you could go to the **Content Completion** subtab of the **Author** tab and in the **Filter - Remove content completion items** list, add elements that you do not want to be presented to the end users.
6. Click **OK** to close the dialog box and then **OK** or **Apply** to save the changes to the ***Document Type Association*** *preferences page*.

   Results

   After you perform these steps you will have a fully functional *framework* in the `dita-extension` folder and it can be shared with others.

Sharing the Extended Framework

There are several ways that you can share the extended *framework* with others:

- Copy it to their `[OXYGEN_INSTALL_DIR]/frameworks` directory.
- Create a `custom_frameworks` folder (anywhere on disk) and copy the extended *framework* into it. Then add the path to your `custom_frameworks` folder in the **Additional frameworks directories** list in the ***Document Type Association > Locations*** *preferences page*.
- Distribute the extended *framework* along with a project by following these steps:
  1. On your local drive, create a directory (with full write access) that contains the project files and a `custom_frameworks` folder with your extended *framework* inside.
  2. Start Oxygen XML Editor, go to the ***Project*** *view* and create a project. Save it in the newly created directory.
  3. In the ***Document Type Association > Locations*** *preferences page*, select ***Project Options*** at the bottom of the page.
  4. In the **Additional frameworks directories** list, add an entry like this: `${pd}/custom_frameworks`.
  5. Add other resources to your project (for example, you can have all your DITA content located inside the project folder).
  6. You can then share the new project directory with other users. For example, you can commit it to your version control system and have them update their working copy. When they open the customized project file in the ***Project*** *view*, the new *framework* becomes available in the list of document types.
- Deploy the extended *framework* configuration *as an add-on*.

After your team members install the *framework* they can check the list of document types in the ***Document Type Association** preferences page* to see if the *framework* is present and if it appears before the bundled **DITA** *framework* (meaning that it has higher priority).

Related Information:

Sharing a Framework (Document Type) on page 1239

Customizing the Content Completion Assistant

Oxygen XML Editor gathers information from the associated schemas (DTDs, XML Schema, RelaxNG) to determine the proposals that appear in the *Content Completion Assistant*. Oxygen XML Editor also includes support that allows you to customize the *Content Completion Assistant* to suit your specific needs.

There are two ways to customize the *Content Completion Assistant* in Oxygen XML Editor:

- You can add, modify, or remove actions that are proposed for each particular document type (*framework*) by using the ***Content Completion** subtab in the **Document Type Association** configuration dialog box*. To access this subtab, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, use the ***New**, **Edit**, **Duplicate**, or **Extend** button*, click on the **Author** tab, and then the **Content Completion** subtab.
- You can use a `cc_config.xml` configuration file that is specific to each document type (*framework*) to configure the values that are proposed in certain contexts, to customize the attributes or elements that are proposed, or to customize how certain aspects of the proposals are rendered in the interface. The rest of the topics in this section explain how you can use this configuration file to customize the content completion.

Configuring the Proposals for Attribute and Element Values

Oxygen XML Editor includes support for configuring the proposed values that appear in the *Content Completion Assistant*. To do so, a configuration file is used, along with the associated schema, to add or replace possible values for attributes or elements that are proposed in the *Content Completion Assistant*.

For an example of a specific use-case, suppose that you want the *Content Completion Assistant* to propose several possible values for the language code when you use an `xml:lang` attribute.

Setting up the Content Completion Configuration File

To customize the configuration file for the *Content Completion Assistant*, follow these steps:

1. Create a new `resources` folder (if it does not already exist) in the `frameworks` directory for the particular document type (for example, `OXYGEN_INSTALL_DIR/frameworks/dita/resources`).
2. *Open the **Preferences** dialog box (**Options** > **Preferences**)* and go to **Document Type Association**. Select the particular document type, click the **Edit** button, and in the ***Classpath** tab* add a link to that `resources` folder (if it does not already exist).
3. Create a new configuration file or edit an existing one.
   a. To easily create a new configuration file, you can use the *Content Completion Configuration* file template that is included in Oxygen XML Editor (**File** > **New** > **Framework templates** > **Oxygen Extensions** > **Content Completion Configuration**). The file template includes details about how each element and attribute is used in the configuration file.
   b. If a configuration file (`cc_config.xml`) already exists for the particular document type (in the `resources` folder), you can modify this existing file.
4. Make the appropriate changes to your custom configuration file.
5. Save the file in the `resources` folder for the particular document type, using the fixed name: `cc_config.xml` (for example, `OXYGEN_INSTALL_DIR/frameworks/dita/resources/cc_config.xml`).
6. Restart the application and open an XML document. In the *Content Completion Assistant* you should see your customizations.

   Tip: In some cases, you can simply use the **Refresh (F5)** action to test your customizations, without having to restart the application.

Configuring Proposed Values

For the purposes of adding or replacing the values that are proposed, the configuration file (`cc_config.xml`) includes a series of `match` instructions that will match an element or attribute name. You also have the possibility of using an attribute called `editable` on the `match` element to specify the editable state of the attribute values, as reflected in the ***Attributes*** *view* and the ***In-place Attributes Editor***. The possible values for the `editable` attribute are:

- `true` - The attribute values can be edited by choosing from a combo box or manually providing a value.
- `false` - The attribute values cannot be edited.
- `onlyAllowedItems` - The attribute values can be edited, but only by choosing from a list of proposed values, in a non-editable combo box.

A new value is specified inside one or more `item` elements, which are grouped inside an `items` element. The behavior of the `items` element is specified with the help of the `action` attribute, which can have any of the following values:

- `append` - Adds new values to appear in the proposals list (default value).
- `addIfEmpty` - Adds new values to the proposals list only if no other values are contributed by the schema.
- `replace` - Replaces the values contributed by the schema with new values to appear in the proposals list.

The values in the configuration file can be specified either directly or by calling an external XSLT file that will extract data from an external source.

Other Important Notes About the Configuration File

Important:

- This configuration file only affects the content completion assistance, not validation.
- To test the effects of your changes, you should restart the application.

Example: Specifying Values Directly

If you want to specify the values directly, the configuration file should look like this:

```
<!-- Replaces the values for an element with the local name "lg",
      from the given namespace -->
<match elementName="lg" elementNS="http://www.oxygenxml.com/ns/samples">
    <items action="replace">
        <item value="stanza"/>
        <item value="refrain"/>
    </items>
</match>

<!-- Adds two values for an attribute with the local name "type",
      from any namespace -->
<match attributeName="type" editable="onlyAllowedItems">
    <items>
        <item value="stanza"/>
        <item value="refrain"/>
    </items>
</match>
```

Example: Calling an External XSLT Script

If you want to collect values from an external XSLT script, the configuration file should include something like this:

```
<xslt href="../xsl/get_values_from_db.xsl" useCache="false" action="replace"/>
```

In this example, the `get_values_from_db.xsl` is executed to extract values from a database.

Tip: You can use `xsl:message` as a debugging mechanism. These messages are presented in the results area at the bottom of the application whenever the *Content Completion Assistant* is invoked.

Note: A comprehensive XSLT sample is included in the **Content Completion Configuration** file template.

Configuring Proposed Values in the Context that the Content Completion was Invoked

A more complex scenario is if you want to choose the possible values to propose, depending on the context of the element in which the content completion was invoked.

Suppose that you want to propose certain possible values for one property (for example, *color*) and other values for another property (for example, *shape*). If the property represents a color, then the values should represent applicable colors, while if the property represents a shape, then the values should represent applicable shapes. See the following code snippets:

Your main document:

```
<sampleArticle>
  <!-- The possible values for @value should be "red" and "blue" -->
  <property name="color" value=""/>
  <!-- The possible values for @value should be "square" and "rectangle" -->
  <property name="shape" value=""/>
</sampleArticle>
```

The content completion configuration file:

```
<config xmlns="http://www.oxygenxml.com/ns/ccfilter/config">
    <match elementName="property" attributeName="value">
      <xslt href="get_values.xsl" useCache="false" action="replace"/>
    </match>
</config>
```

The stylesheet that defines the possible values based on the context of the property on which the content completion was invoked:

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:saxon="http://saxon.sf.net/"
  exclude-result-prefixes="xs"
  version="2.0">

  <xsl:param name="documentSystemID" as="xs:string"></xsl:param>
  <xsl:param name="contextElementXPathExpression" as="xs:string"></xsl:param>

  <xsl:template name="start">
    <xsl:apply-templates select="doc($documentSystemID)"/>
  </xsl:template>

  <xsl:template match="/">
   <xsl:variable name="propertyElement"
    select="saxon:eval(saxon:expression($contextElementXPathExpression, ./*))"/>

   <items>
     <xsl:if test="$propertyElement/@name = 'color'">
       <item value='red'/>
       <item value='blue'/>
     </xsl:if>
     <xsl:if test="$propertyElement/@name = 'shape'">
       <item value='rectangle'/>
       <item value='square'/>
     </xsl:if>
   </items>
  </xsl:template>
</xsl:stylesheet>
```

The `contextElementXPathExpression` parameter will be bound to an XPath expression that identifies the element in the context for which the content completion was invoked.

Related Information:

Configuring the Proposals for Elements on page 1243
Customizing the Rendering of Elements on page 1248

Configuring the Proposals for Elements

There are many cases where elements have a relaxed content model and can accept a large number of child elements. For example, the DITA list item element (`li`) accepts more than 60 child elements. Oxygen XML Editor includes support to allow the content architect to put some constraints on the possible elements or attributes, or to impose some best practices in the way content is edited.

For an example of a specific use-case, suppose that you want restrict DITA list item elements (`li`) to only accept paragraph elements (p). In this case, the *Content Completion Assistant* should not offer any element other than

a paragraph (p) when a list item (`li`) is inserted into a document. It would also be helpful if the required child element (p) was automatically inserted whenever a list item (`li`) is inserted.

One method of changing the content model is to alter the element definition in the associated schema (XML Schema, DTD, RelaxNG), but this may be quite complicated in some cases. Fortunately, Oxygen XML Editor offers a simple, alternative method of using a configuration file to customize the content completion proposals for each element.

Setting up the Content Completion Configuration File

To customize the configuration file for the *Content Completion Assistant*, follow these steps:

1. Create a new `resources` folder (if it does not already exist) in the `frameworks` directory for the particular document type (for example, `OXYGEN_INSTALL_DIR/frameworks/dita/resources`).
2. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Document Type Association**. Select the particular document type, click the **Edit** button, and in the ***Classpath** tab* add a link to that `resources` folder (if it does not already exist).
3. Create a new configuration file or edit an existing one.
   a. To easily create a new configuration file, you can use the *Content Completion Configuration* file template that is included in Oxygen XML Editor (**File** > **New** > **Framework templates** > **Oxygen Extensions** > **Content Completion Configuration**). The file template includes details about how each element and attribute is used in the configuration file.
   b. If a configuration file (`cc_config.xml`) already exists for the particular document type (in the `resources` folder), you can modify this existing file.
4. Make the appropriate changes to your custom configuration file.
5. Save the file in the `resources` folder for the particular document type, using the fixed name: `cc_config.xml` (for example, `OXYGEN_INSTALL_DIR/frameworks/dita/resources/cc_config.xml`).
6. Restart the application and open an XML document. In the *Content Completion Assistant* you should see your customizations.

   Tip: In some cases, you can simply use the **Refresh (F5)** action to test your customizations, without having to restart the application.

Configuring Elements or Attributes that are Proposed for Each Element

For the purposes of customizing the elements or attributes that are proposed for each individual element, the configuration file (`cc_config.xml`) uses `elementProposals` elements. This element allows you to customize or filter the child elements and attributes for an element.

Elements:

To control the **elements** that are proposed for an element, you can use the following attributes for the `elementProposals` element:

- `path` - A path within the document that matches the element that will have its content completion proposals changed. For example, `"title"` matches all the `title` elements in the document, while `"chapter/title"` matches only the `title` elements that are direct children of the `chapter` element. You can use simplified forms of XPath in this attribute.

  The XPath expressions can accept multiple attribute conditions and inside each condition you can use *AND/OR* boolean operators and parentheses to override the priority.

  You can use one or more of the following attribute conditions (default attribute values are not taken into account):

  - *element[@attr]* - Matches all instances of the specified element that include the specified attribute.
  - *element[not(@attr)]* - Matches all instances of the specified element that do not include the specified attribute.
  - *element[@attr = "value"]* - Matches all instances of the specified element that include the specified attribute with the given value.

- *element[@attr != "value"]* - Matches all instances of the specified element that include the specified attribute and its value is different than the one given.

Example: The following are examples of how you could use multiple boolean operators and parentheses inside an attribute condition:

```
*[@a and @b or @c and @d]
*[@a and (@b or @c) and @d]
```

The following are just examples of how simplified XPath expressions might look like:

- `elementName`
- `//elementName`
- `/elementName1/elementName2/elementName3`
- `//xs:localName` Note: Using a namespace prefix requires that you declare it in the configuration file. For example: `<elementProposals xmlns:db5="http://docbook.org/ns/docbook" path="db5:listitem" insertElements="db5:para" />`
- `//xs:documentation[@lang="en"]`

Note: If the `path` attribute is missing, the customization will apply to the proposals for all elements. You can intentionally omit this attribute and use *`possibleElements`* or *`rejectElements`* to specify or restrict particular elements for a *framework*.

For example, suppose that in your DITA documents, you want to restrict your users from using `image` and `fig` elements because you do not want images to be included in your output. The configuration file should look like this:

```
<elementProposals rejectElements="image fig" />
```

Since the `path` attribute is missing, the specified element will be filtered out from the proposals for the entire *framework*.

- `insertElements` - A space-separated sequence of child element names. Each time the element specified in the `path` attribute is inserted into the document, these child elements will also be inserted in the order that they are listed. For example, `insertElements="b i"` will insert exactly one b element, followed by an `i` element. An empty value (`""`) means that no child elements should be inserted.

  Note: If this attribute is missing, the default required child elements will be inserted, as specified in the associated schema for the document.
- `possibleElements` - A space-separated list of element names that will be shown in the content completion list when invoked inside an element that is specified in the `path` attribute. For example, `"bold italic codeph ph"` means that the *Content Completion Assistant* will contain these four elements when invoked on the element specified in the `path` attribute. The following other possible values are also supported:

  - ***NONE*** - There will be no proposals in the content completion list.
  - ***ALL*** - All the possible elements specified in the associated schema will be presented in the content completion list. This is also the default behavior if this attribute is missing.
  - ***INSERTED*** - The proposals will be the same list of elements that are defined in the `insertElements` attribute.

  When using this attribute to specify multiple elements, only use one entry with the element names separated by a space:

```
<elementProposals possibleElements="bold italic codeph ph" />
```

- `rejectElements` - A space-separated list of element names that will be filtered out from the list of proposals that are presented in the content completion list. Each time the element specified in the `path` attribute is inserted into the document, the list of proposals in the *Content Completion Assistant* will include the entries that are defined in the associated schema, minus the elements specified in this attribute.

  When using this attribute to specify multiple elements, only use one entry with the element names separated by a space:

```
<elementProposals rejectElements="image fig imagemap foreign" />
```

Attributes:

To control the **attributes** that are proposed for an element, you can use the following attributes for the `elementProposals` element:

- `path` - A path within the document that matches the element that will have its attribute proposals changed. For example, `"title"` matches all the `title` elements in the document, while `"chapter/title"` matches only the `title` elements that are direct children of the `chapter` element. You can use simplified forms of XPath in this attribute. For examples of such forms of XPath expressions, see the *note in XML Preferences*.

  Note: If this attribute is missing, the customization will apply to the proposals for all elements. You can intentionally omit this attribute and use *`possibleAttributes`* or *`rejectAttributes`* to specify or restrict attributes for an entire *framework*.

  For example, suppose that you only want to allow a limited set of attributes in a customized *framework*. The configuration file should look like this:

  ```
  <elementProposals possibleAttributes="
                 id domains href scope format type conref
          props keyref class"/>
  ```

  Since the `path` attribute is missing, this applies to the entire *framework* and only the specified attributes will be proposed.
- `insertAttributes` - A space-separated sequence of attribute names that will be inserted along with the element.
- `possibleAttributes` - A space-separated list of attribute names that will be shown in the content completion list when invoked inside an element that is specified in the `path` attribute.

  When using this attribute to specify multiple attributes, only use one entry with the attribute names separated by a space:

  ```
  <elementProposals possibleAttributes="scope format type" />
  ```
- `rejectAttributes` - A space-separated list of attribute names that will be filtered out from the list of proposals that are presented in the content completion list. Each time the element specified in the `path` attribute is inserted into the document, the list of proposals in the *Content Completion Assistant* will include the entries that are defined in the associated schema, minus the attributes specified in this attribute.

  When using this attribute to specify multiple attributes, only use one entry with the attribute names separated by a space:

  ```
  <elementProposals rejectAttributes="importance platform product" />
  ```

Other Important Notes About the Configuration File

Important:

- By default, the element names that do not have a namespace prefix are considered from *no-namespace*. Consider declaring the namespace mapping on the root of the configuration file and prefixing the element names from the `elementPath` and `model` attributes.
- This configuration file only affects the content completion assistance, not validation.
- To test the effects of your changes, you should restart the application, although in some cases, you can simply use the **Refresh (F5)** action to test your customizations.
- When an XML element from the document is matched against a list of configured `elementProposals`, the first one in sequence takes precedence. Therefore, make sure you place the more specific `elementProposals` (those with a longer path) first in your configuration file.
- Regular expression patterns can be used in the following attributes: `possibleElements`, `rejectElements`, `possibleAttributes`, and `rejectAttributes`. For example, `code*`, `*block`, `con*ref`, `_*`.
- Only simple recursion cases are detected and avoided by the editor, and logged to the console. Therefore, if complex `elementProposals` patterns are defined, you should avoid *infinite recursions*.

Examples: Configuring the Element Proposals

- **Example 1: Automatically Insert Elements**

  Suppose that you want to automatically insert a paragraph element (p) whenever a DITA ordered list item element (`ol/li`) is inserted, and also to not allow any other element besides a paragraph inside the ordered list items.

  To achieve this, the configuration file should include the following:

```
<elementProposals path="ol/li" insertElements="p"
       possibleElements="_INSERTED_"/>
```

- **Example 2: Insert Complex Element Structure**

  For a more complex example, suppose that you want to insert a complex structure whenever a DITA `prolog` element is inserted.

  For instance, if you need to insert the following structure inside `prolog` elements:

```
    <prolog>
      <author></author>
      <metadata>
          <keywords>
              <keyword></keyword>
              <keyword></keyword>
          </keywords>
      </metadata>
    </prolog>
```

  The configuration file should include the following:

```
    <elementProposals path="prolog" insertElements="author metadata"/>
    <elementProposals path="prolog/metadata" insertElements="keywords"/>
    <elementProposals path="prolog/metadata/keywords"
           insertElements="keyword, keyword"/>
```

- **Example 3: Limit Possible Elements**

  Suppose that you also want to limit the proposals for the `keywords` element to only allow the user to insert `audience` or `keyword` elements. The configuration file should include the following:

```
<elementProposals path="prolog/metadata" insertElements="keywords"
           possibleElements="audience keywords"/>
```

  Suppose that you want to simply restrict your users from inserting `image` elements inside DITA list item elements (`li`), but still propose all the other elements that are defined in the associated schema. The configuration file should look like this:

```
<elementProposals path="li" rejectElements="image" />
```

Examples: Configuring the Attributes Proposals

- **Example 1: Automatically Insert Attributes**

  Suppose that you want to insert an `id` attribute (with an empty value) whenever a DITA list item element (`li`) is inserted. The configuration file should include the following:

```
<elementProposals path="li" insertAttributes="id"/>
```

- **Example 2: Limit Possible Attributes**

  Suppose that you also want to limit the number of choices for attributes that are presented to the user whenever a DITA list item element (`li`) is inserted. The configuration file should look like this:

```
<elementProposals path="li" insertAttributes="id"
          possibleAttributes="id product platform audience"/>
```

Suppose that you want to simply restrict your users from inserting `conref` attributes inside DITA topics (`topic` element), but still propose all the other attributes that are defined in the associated schema. The configuration file should look like this:

```
<elementProposals path="topic" rejectAttributes="conref" />
```

Related Information:

Configuring the Proposals for Attribute and Element Values on page 1241

Customizing the Rendering of Elements on page 1248

Customizing the Rendering of Elements

In addition to the support for configuring the proposals that appear in the *Content Completion Assistant*, Oxygen XML Editor also includes support for customizing how the elements are rendered. You can do this by using the *`XMLNodeRendererCustomizer` API extension*, but you can also use the same configuration file that is used to configure the content completion proposals.

For an example of a specific use-case, suppose that in DITA you want the names of paragraph elements (p) to be rendered as "**Paragraph**" instead of "**p**" in the various components in **Author** mode (such as in the ***Outline*** *view*, ***Elements*** *view*, ***Attributes*** *view*, and the breadcrumb navigation bar). To achieve this, you can use the `elementRenderings` element in the configuration file.

Setting up the Content Completion Configuration File

To customize the configuration file for the *Content Completion Assistant*, follow these steps:

1. Create a new `resources` folder (if it does not already exist) in the `frameworks` directory for the particular document type (for example, `OXYGEN_INSTALL_DIR/frameworks/dita/resources`).
2. *Open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)*** and go to **Document Type Association**. Select the particular document type, click the **Edit** button, and in the ***Classpath*** *tab* add a link to that `resources` folder (if it does not already exist).
3. Create a new configuration file or edit an existing one.
   a. To easily create a new configuration file, you can use the *Content Completion Configuration* file template that is included in Oxygen XML Editor (**File** > **New** > **Framework templates** > **Oxygen Extensions** > **Content Completion Configuration**). The file template includes details about how each element and attribute is used in the configuration file.
   b. If a configuration file (`cc_config.xml`) already exists for the particular document type (in the `resources` folder), you can modify this existing file.
4. Make the appropriate changes to your custom configuration file.
5. Save the file in the `resources` folder for the particular document type, using the fixed name: `cc_config.xml` (for example, `OXYGEN_INSTALL_DIR/frameworks/dita/resources/cc_config.xml`).
6. Restart the application and open an XML document. In the *Content Completion Assistant* you should see your customizations.

   Tip: In some cases, you can simply use the **Refresh (F5)** action to test your customizations, without having to restart the application.

Changing the Rendering of Elements (Their Names, Annotations, and Icons)

For the purposes of customizing how the content completion elements are rendered, you can use the `render` element inside a `elementRenderings` element to specify how element names, their annotations, and their icons are rendered.

You can use the following attributes for the `render` element:

- `element` - Identifies the element to be customized, in the form of a qualified name. If it does not have a prefix, it is considered to be from *noNamespace*.
- `as` - Provides the name (label) that will be displayed for the element in various components in **Author** mode (such as in the *Content Completion Assistant*, the breadcrumb navigation bar, the ***Full Tags*** *display mode*, and

the ***Outline***, ***Elements***, ***Attributes*** views). This attribute is optional. If it is missing, the name of the element is used.

- `iconPath` - Optional attribute that specifies the icon for the element. This is shown in the *Content Completion Assistant* and the ***Outline*** *view* in **Author** mode. If it is a relative path, the full path of the icon image file will be computed starting from the directory of the configuration file (for example, a value of `"myImg.png"` will cause Oxygen XML Editor to load `"frameworks/$ {framework}/resources/myImg.png"`). If you want to access a built-in resource, the value can begin with a forward slash `"/"`, and the image file will be searched for in the Oxygen XML Editor classpath resources (for example,`"/images/OrderedList16.png"` will load an icon from the built-in Oxygen XML Editor *JAR* file resources.
- `xml:lang` - Optional attribute that could be used to render the same element differently, depending on the language. If there are multiple `render` elements for the same `element` attribute (element name) and the `xml:lang` attribute is missing on one of them, that one will be considered the default fallback value to be used if none of the others match the language specified in the interface.

  Note: The default entry should be listed first, since the application tries to match them in sequence and the last match found is the one that is used.

  For example, suppose that you want the name of DITA paragraph elements (p) to be rendered as "Paragraphe" if the language is French, "Absatz" if the language is German, and "Paragraph" if the language is English (or any other language). Your configuration file should look something like this:

```
<elementRenderings>
  <render element="p" as="Paragraph"/>
  <render element="p" as="Paragraphe" xml:lang="fr"/>
  <render element="p" as="Absatz" xml:lang="de"/>
</elementRenderings>
```

You can also use the configuration file to customize the annotations for elements. For this purpose, the `render` element also accepts the following element to change the tooltip annotations for an element (in both **Author** mode and **Text** mode):

- `annotation` - This element can be used within the `render` element to customize the tooltip annotations that are displayed for the element in various components in **Author** mode (such as tooltips shown in the *Content Completion Assistant* documentation window, the breadcrumb navigation bar, the ***Full Tags*** *display mode*, and the ***Outline***, ***Elements***, ***Attributes*** views), as well as the tooltips that are displayed when you hover over elements in **Text** mode. You can use HTML content to style the annotations (see the *example below*).

  Note: If this element is missing, the *styling for the annotations for that element is collected from the associated schema*.

Other Important Notes About the Configuration File for Rendering Elements

Important:

- This configuration file only affects the content completion assistance, not validation.
- To test the effects of your changes, you should restart the application, although in some cases, you can simply use the **Refresh (F5)** action to test your customizations.
- If the *framework* has an associated *style guide*, then the annotations defined in the configuration file will take precedence over those defined in the *style guide*. To check to see if your *framework* uses a *style guide*, look for the following folder: `${oXygenInstallDir}frameworks/${framework}/styleguide/`. If that folder exists, it is recommended that you make your annotation changes directly in the *style guide*, rather than in the configuration file.
- If an *`XMLNodeRendererCustomizer` API extension* has been implemented for the *framework* and a configuration file is also used, the rendering customization for an element will be the result of merging the two. For example, if the `XMLNodeRendererCustomizer` implementation customizes the element name, while the configuration file specifies an icon for the element, the properties of both customizations will be rendered. However, if both implementations define the same property (for example, both specify the rendering of an element name), the customizations defined in the configuration file take precedence.
- The rendering customizations defined in the configuration file also applies to aspects of the Oxygen XML Web Author Component interface.

Example: Changing the Rendering of an Element

Suppose that you want to render the name of the DITA `title` element to begin with a capital letter, use a custom icon for it, and provide specific documentation for that element in the various components in **Author** mode. The configuration file should look like this:

```
<elementRenderings>
   <render element="title"   as="Title"  iconPath="cimg/AcceptAll16.png">
      <annotation>
         <html xmlns="http://www.w3.org/1999/xhtml">
           <head>
             <title>Documentation for the Title Element</title>
           </head>
             <body>
           <p>A <i>heading</i> or <b>label</b> for the main parts of a topic</p>
             </body>
         </html>
             </annotation>
   </render>
   </elementRenderings>
```

Related Information:

Configuring the Proposals for Attribute and Element Values on page 1241
Configuring the Proposals for Elements on page 1243
Configuring an XML Node Renderer Customizer on page 1237
Schema Annotations in Author Mode on page 347
Annotations for XML Elements and Attributes in Content Completion Assistant on page 1250

Annotations for XML Elements and Attributes in Content Completion Assistant

Oxygen XML Editor gathers documentation from the associated schemas (DTD, XML Schema, RelaxNG) and presents it for each element or attribute. For example, if you open the *Content Completion Assistant* for a recognized XML vocabulary, documentation is displayed for each element provided by the associated schema. Similar information is displayed when you hover over tag names presented in the ***Elements** view*. If you hover over attributes in the ***Attributes** view* you also see information about each attribute, gathered from the same schema.

If you have a *framework configuration* set up for your XML vocabulary, there is a special XML configuration file that can be added to provide additional documentation information or links to specification web pages for certain elements and attributes.

To provide this additional information in the *Content Completion Assistant*, follow these steps:

1. Create a new folder in the configuration directory for the document type.

   Example: `OXYGEN_INSTALL_DIR/frameworks/dita/styleguide`
2. Use the **New** document wizard to create a file using the **Content Completion Styleguide** file template (in the **Oxygen Extensions** section).
3. Save the file in the folder created in step 1, using the fixed name: `contentCompletionElementsMap.xml`.
4. *Open the **Preferences** dialog box **(Options > Preferences)***, go to **Document Type Association**, and edit the document type configuration for your XML vocabulary. Now you need to indicate where Oxygen XML Editor will locate your mapping file by doing one of the following:
   - In the **Classpath** tab add a link to the newly created folder.
   - In the **Catalogs** tab *add a new catalog file*. The selected file needs to contain the following:

     ```
     <?xml version="1.0" encoding="UTF-8"?>
     <!DOCTYPE catalog PUBLIC "-//OASIS//DTD XML Catalogs V1.1//EN"
         "http://www.oasis-open.org/committees/entity/release/1.1/catalog.dtd">
     <catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
         <uri name="http://www.oxygenxml.com/{processed_dt_name}/styleguide/
     contentCompletionElementsMap.xml" uri="contentCompletionElementsMap.xml"/>
     </catalog>
     ```

     where *{processed_dt_name}* is the name of the document type in lower case and with spaces replaced by underscores.

     Note: If Oxygen XML Editor finds a mapping file in both locations, the one in the **Catalogs** tab takes precedence.
5. Make the appropriate changes to your custom mapping file.

Example: You can look at how the DITA mapping file is configured: `OXYGEN_INSTALL_DIR/frameworks/dita/styleguide/contentCompletionElementsMap.xml`

The associated XML Schema contains additional details about how each element and attribute is used in the mapping file.

6. Re-open the application and open an XML document.

In the *Content Completion Assistant*, you should see the additional annotations for each element.

Related Information:

Customizing the Rendering of Elements on page 1248

Example Files for a Custom Framework

This section lists the files used in the customization tutorials: the XML Schema, CSS files, XML files, XSLT stylesheets.

XML Schema

XML Schema file listings.

`sdf.xsd`

This sample file can also be found in our sample framework package that can be downloaded at: *https://www.oxygenxml.com/php/get_oxygen_sample_framework.php*. This sample is in the `schema` directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.oxygenxml.com/sample/documentation"
    xmlns:doc="http://www.oxygenxml.com/sample/documentation"
    xmlns:abs="http://www.oxygenxml.com/sample/documentation/abstracts"
    elementFormDefault="qualified">

    <xs:import
        namespace="http://www.oxygenxml.com/sample/documentation/abstracts"
        schemaLocation="abs.xsd"/>

    <xs:element name="book" type="doc:sectionType"/>
    <xs:element name="article" type="doc:sectionType"/>
    <xs:element name="section" type="doc:sectionType"/>

    <xs:complexType name="sectionType">
        <xs:sequence>
            <xs:element name="title" type="xs:string"/>
            <xs:element ref="abs:def" minOccurs="0"/>
            <xs:choice>
                <xs:sequence>
                    <xs:element ref="doc:section"
                        maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:choice maxOccurs="unbounded">
                    <xs:element ref="doc:para"/>
                    <xs:element ref="doc:ref"/>
                    <xs:element ref="doc:image"/>
                    <xs:element ref="doc:table"/>
                </xs:choice>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="para" type="doc:paragraphType"/>

    <xs:complexType name="paragraphType" mixed="true">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="b"/>
            <xs:element name="i"/>
            <xs:element name="link"/>
        </xs:choice>
    </xs:complexType>

    <xs:element name="ref">
        <xs:complexType>
            <xs:attribute name="location" type="xs:anyURI"
                use="required"/>
        </xs:complexType>
    </xs:element>

    <xs:element name="image">
        <xs:complexType>
            <xs:attribute name="href" type="xs:anyURI"
                use="required"/>
        </xs:complexType>
    </xs:element>
```

```
    <xs:element name="table">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="customcol" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:attribute name="width" type="xs:string"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="header">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="td"
                                maxOccurs="unbounded"
                                type="doc:paragraphType"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="tr" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="td"
                                type="doc:tdType"
                                maxOccurs="unbounded"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="width" type="xs:string"/>
        </xs:complexType>
    </xs:element>


    <xs:complexType name="tdType">
        <xs:complexContent>
            <xs:extension base="doc:paragraphType">
                <xs:attribute name="row_span"
                    type="xs:integer"/>
                <xs:attribute name="column_span"
                    type="xs:integer"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

`abs.xsd`

This sample file can also be found in our sample framework package that can be downloaded at: *https://www.oxygenxml.com/php/get_oxygen_sample_framework.php*. This sample is in the `schema` directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace=
 "http://www.oxygenxml.com/sample/documentation/abstracts">
    <xs:element name="def" type="xs:string"/>
</xs:schema>
```

CSS

CSS file listing.

`sdf.css`

This sample file can also be found in our sample framework package that can be downloaded at: *https://www.oxygenxml.com/php/get_oxygen_sample_framework.php*. This sample is in the `css` directory.

```
/* Element from another namespace */
@namespace abs "http://www.oxygenxml.com/sample/documentation/abstracts";

abs|def{
    font-family:monospace;
    font-size:smaller;
}
abs|def:before{
    content:"Definition:";
    color:gray;
}

/* Vertical flow */
book,
section,
para,
title,
image,
ref {
    display:block;
}
```

```
/* Horizontal flow */
b,i {
    display:inline;
}

section{
    margin-left:1em;
    margin-top:1em;
}

section{
    -oxy-foldable:true;
    -oxy-not-foldable-child: title;
}

link[href]:before{
    display:inline;
    link:attr(href);
    content: "Click to open: " attr(href);
}

/* Title rendering*/
title{
    font-size: 2.4em;
    font-weight:bold;
}

* * title{
    font-size: 2.0em;
}
* * * title{
    font-size: 1.6em;
}
* * * * title{
    font-size: 1.2em;
}

book,
article{
    counter-reset:sect;
}
book > section,
article > section{
    counter-increment:sect;
}
book > section > title:before,
article > section > title:before{
    content: "Section: " counter(sect) " ";
}

/* Inlines rendering*/
b {
    font-weight:bold;
}

i {
    font-style:italic;
}

/*Table rendering */
table{
    display:table;
    border:1px solid navy;
    margin:1em;
    max-width:1000px;
    min-width:150px;
}

table[width]{
  width:attr(width, length);
}

tr, header{
    display:table-row;
}

header{
    background-color: silver;
    color:inherit
}

td{
  display:table-cell;
  border:1px solid navy;
  padding:1em;
}

image{
    display:block;
    content: attr(href, url);
    margin-left:2em;
}
```

XML

XML file listing.

`sdf_sample.xml`

This sample file can also be found in our sample framework package that can be downloaded at: *https://www.oxygenxml.com/php/get_oxygen_sample_framework.php*.

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.oxygenxml.com/sample/documentation"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:abs="http://www.oxygenxml.com/sample/documentation/abstracts">
    <title>My Technical Book</title>
    <section>
        <title>XML</title>
        <abs:def>Extensible Markup Language</abs:def>
        <para>In this section of the book I will explain
            different XML applications.</para>
    </section>
    <section>
        <title>Accessing XML data.</title>
        <section>
            <title>XSLT</title>
            <abs:def>Extensible stylesheet language
                transformation (XSLT) is a language for
                transforming XML documents into other XML
                documents.</abs:def>
            <para>A list of XSL elements and what they do..</para>
            <table>
                <header>
                    <td>XSLT Elements</td>
                    <td>Description</td>
                </header>
                <tr>
                    <td>
                        <b>xsl:stylesheet</b>
                    </td>
                    <td>The <i>xsl:stylesheet</i> element is
                        always the top-level element of an
                        XSL stylesheet. The name
                            <i>xsl:transform</i> may be used
                        as a synonym.</td>
                </tr>
                <tr>
                    <td>
                        <b>xsl:template</b>
                    </td>
                    <td>The <i>xsl:template</i> element has
                        an optional mode attribute. If this
                        is present, the template will only
                        be matched when the same mode is
                        used in the invoking
                            <i>xsl:apply-templates</i>
                        element.</td>
                </tr>
                <tr>
                    <td>
                        <b>for-each</b>
                    </td>
                    <td>The xsl:for-each element causes
                        iteration over the nodes selected by
                        a node-set expression.</td>
                </tr>
                <tr>
                    <td column_span="2">End of the list</td>
                </tr>
            </table>
        </section>
        <section>
            <title>XPath</title>
            <abs:def>XPath (XML Path Language) is a terse
                (non-XML) syntax for addressing portions of
                an XML document. </abs:def>
            <para>Some of the XPath functions.</para>
            <table>
                <header>
                    <td>Function</td>
                    <td>Description</td>
                </header>
                <tr>
                    <td>format-number</td>
                    <td>The <i>format-number</i> function
                        converts its first argument to a
                        string using the format pattern
                        string specified by the second
                        argument and the decimal-format
                        named by the third argument, or the
                        default decimal-format, if there is
                        no third argument</td>
                </tr>
                <tr>
```

```
                    <td>current</td>
                    <td>The <i>current</i> function returns
                        a node-set that has the current node
                        as its only member.</td>
                </tr>
                <tr>
                    <td>generate-id</td>
                    <td>The <i>generate-id</i> function
                        returns a string that uniquely
                        identifies the node in the argument
                        node-set that is first in document
                        order.</td>
                </tr>
            </table>
        </section>
    </section>
    <section>
        <title>Documentation frameworks</title>
        <para>One of the most important documentation
            frameworks is DocBook.</para>
        <image
            href="http://www.xmlhack.com/images/docbook.png"/>
        <para>The other is the topic oriented DITA, promoted
            by OASIS.</para>
        <image
href="http://www.oasis-open.org/images/standards/oasis_standard.jpg"
        />
    </section>
</book>
```

XSL

XSL file listing.

`sdf.xsl`

This sample file can also be found in our sample framework package that can be downloaded at: *https://www.oxygenxml.com/php/get_oxygen_sample_framework.php*. This sample is in the `xsl` directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0"
    xpath-default-namespace=
    "http://www.oxygenxml.com/sample/documentation">

    <xsl:template match="/">
        <html><xsl:apply-templates/></html>
    </xsl:template>

    <xsl:template match="section">
        <xsl:apply-templates/>
    </xsl:template>

    <xsl:template match="image">
        <img src="{@href}"/>
    </xsl:template>

    <xsl:template match="para">
        <p>
            <xsl:apply-templates/>
        </p>
    </xsl:template>

    <xsl:template match="abs:def"
        xmlns:abs=
        "http://www.oxygenxml.com/sample/documentation/abstracts">
        <p>
            <u><xsl:apply-templates/></u>
        </p>
    </xsl:template>

    <xsl:template match="title">
        <h1><xsl:apply-templates/></h1>
    </xsl:template>

    <xsl:template match="b">
        <b><xsl:apply-templates/></b>
    </xsl:template>

    <xsl:template match="i">
        <i><xsl:apply-templates/></i>
    </xsl:template>

    <xsl:template match="table">
        <table frame="box" border="1px">
            <xsl:apply-templates/>
        </table>
    </xsl:template>

    <xsl:template match="header">
```

```
        <tr>
            <xsl:apply-templates/>
        </tr>
    </xsl:template>

    <xsl:template match="tr">
        <tr>
            <xsl:apply-templates/>
        </tr>
    </xsl:template>

    <xsl:template match="td">
        <td>
            <xsl:apply-templates/>
        </td>
    </xsl:template>

    <xsl:template match="header/header/td">
        <th>
            <xsl:apply-templates/>
        </th>
    </xsl:template>

</xsl:stylesheet>
```

CSS Support in Author Mode

Author editing mode supports most CSS 2.1 selectors, numerous CSS 2.1 properties, and some CSS 3 selectors. Oxygen XML Editor also supports stylesheets coded with the LESS dynamic stylesheet language. Also, some custom functions and properties that extend the W3C CSS specification, and are useful for URL and string manipulation, are available to developers who create **Author** editing *frameworks*.

Configuring and Managing Multiple CSS Styles

Oxygen XML Editor provides a **Styles** drop-down menu on the toolbar that allows you to select one *main (non-alternate) CSS style* and multiple *alternate CSS styles*. This makes it easy to change the look of the document as it appears in **Author** mode and the output without having to continually edit the CSS stylesheets.

An example of a common use case is when content authors want to use custom styling within a document. You can select a *main* CSS stylesheet that styles the whole document and then apply *alternate* styles, as layers, to specific parts of the document.

Managing the CSS Styles

The *main* and *alternate* styles that are listed in the **Styles** drop-down menu can be controlled in the ***Document Type** configuration dialog box*. To access it, follow these steps:

1. *Open the **Preferences** dialog box* .
2. Go to **Document Type Association**.
3. Select the appropriate document type and press the **Edit** button.

   Important: If you do not have access rights to the folder where the *framework* files are stored, you can either elevate read/write permissions on that *framework* folder or you can extend the *framework* and customize the CSS stylesheets in the extension. If you want to share the customized extension with the rest of your team, see *Sharing the Extended Framework* on page 1240.

The CSS styles (CSS files) associated with the particular document type are listed in the ***CSS** subtab* of the **Author** tab.

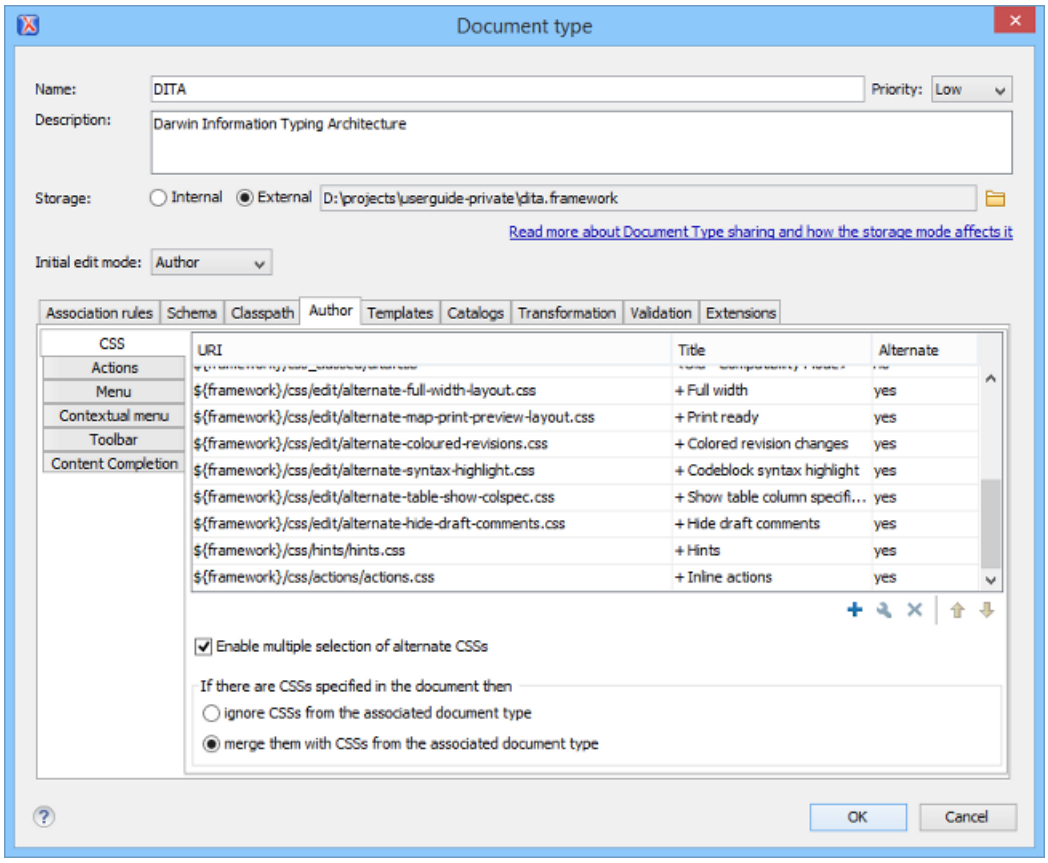


Figure 586: Main and Alternate CSS Styles in the Document Type Configuration Dialog Box

You can **Add**, **Edit**, or **Delete** styles from this dialog box to manage the *main* and *alternate* styles associated to the particular document type. You can also change the order of the styles by using the **Move Up** and **Move Down** buttons. This will also change the order that they appear in the **Styles** drop-down menu. The *alternate* styles are combined with the *main* CSS sequentially, in the order that they appear in this list. Therefore, if the same style rules are included in multiple CSS files, the rules that are defined in the last *alternate* style in this list will take precedence, since it is the last one to be combined (applied as a layer).

The **URI** column shows the path of each CSS file. The names listed in the **Styles** drop-down menu match the values in the **Title** column. The value in the **Alternate** column determines whether it is a *main* or *alternate* CSS. If the value is *no* it is a *main* CSS. If the value is *yes* it is an *alternate* CSS and the style can be combined with a *main* CSS or other *alternate* styles when using the **Styles** drop-down menu.

Note: To group alternate styles into categories (submenus), use a vertical bar character (|) in the **Title** column. You can use multiple vertical bars for multiple submenus. The text before each vertical bar will be rendered as the name of a submenu entry in the **Styles** drop-down menu, while the text after the final vertical bar will be rendered as the name of the style inside the submenu.

Example: Suppose that you want to add two alternate stylesheets in separate submenus, with the **Title** column set to `My Styles|User Assistance|Hints` and `My Styles|User Actions|Inline Actions`, respectively.

URI	Title	Alternate
${framework}/css/hints/hints.css	My Styles|User Assistance| Hints	yes
${framework}/css/actions/actions.css	My Styles|User Actions|Inline actions	yes

Oxygen XML Editor will add a `My Styles` submenu with two submenus (`User Assistance` that contains the `Hints` style, and `User Actions` that contains the `Inline Actions` style) in the **Styles** drop-down menu.

The ***Enable multiple selection of alternate CSSs*** *checkbox* at the bottom of the pane must be selected for the *alternate styles*to be combined. They are applied like layers and you can activate any number of them. If this option is not selected, the *alternate* styles are treated like *main CSS styles* and you can only select one at a time. By default, this option is selected. There are also a few options that allow you to specify how to handle the CSS if there are CSS styles specified in the document. You can choose to *ignore* or *merge* them.

The following rules apply for merging CSS styles:

- CSS files with the same title will be merged.
- CSS files without a title will contribute to all others.
- They are merged sequentially, in the order that they appear in the list.

Using the Styles Drop-Down Menu

You can use the **Styles** drop-down menu to select a *main style* that applies to the whole document and then select one or more *alternate styles* that behave like layers and are merged sequentially with the *main style*. The *main* styles are listed in the top section, while the *alternate styles* are listed in the bottom section. The styles (defined in their respective CSS files) that are selected in this drop-down menu are used to render your documents in **Author** mode and in the output.

Note: If you deselect the ***Enable multiple selection of alternate CSSs*** *option* in the **CSS** subtab of the ***Document Type*** *configuration dialog box*, the *alternate styles* are treated like *main CSS styles* and you can only select one at a time.

The selections from the **Styles** drop-down menu are persistent, meaning that Oxygen XML Editor will remember the selections when subsequent documents are opened.

EXAMPLE: CSS Styles in DITA

Oxygen XML Editor comes with a set of predefined CSS layer stylesheets for DITA documents (as well as some that are specifically for *DITA maps*). In the subsequent figure, a DITA document has the **Century** style selected for the *main CSS style* and the *alternate styles* **Full width**, **Show table column specification**, **Hints**, and **Inline actions** are combined for additive styling to specific parts of the document.

Tip: The **Hints** style displays tooltips throughout DITA documents that offer additional information to help you with the DITA structure. The **Inline actions** style displays possible elements that are allowed to be inserted at various locations throughout DITA documents.

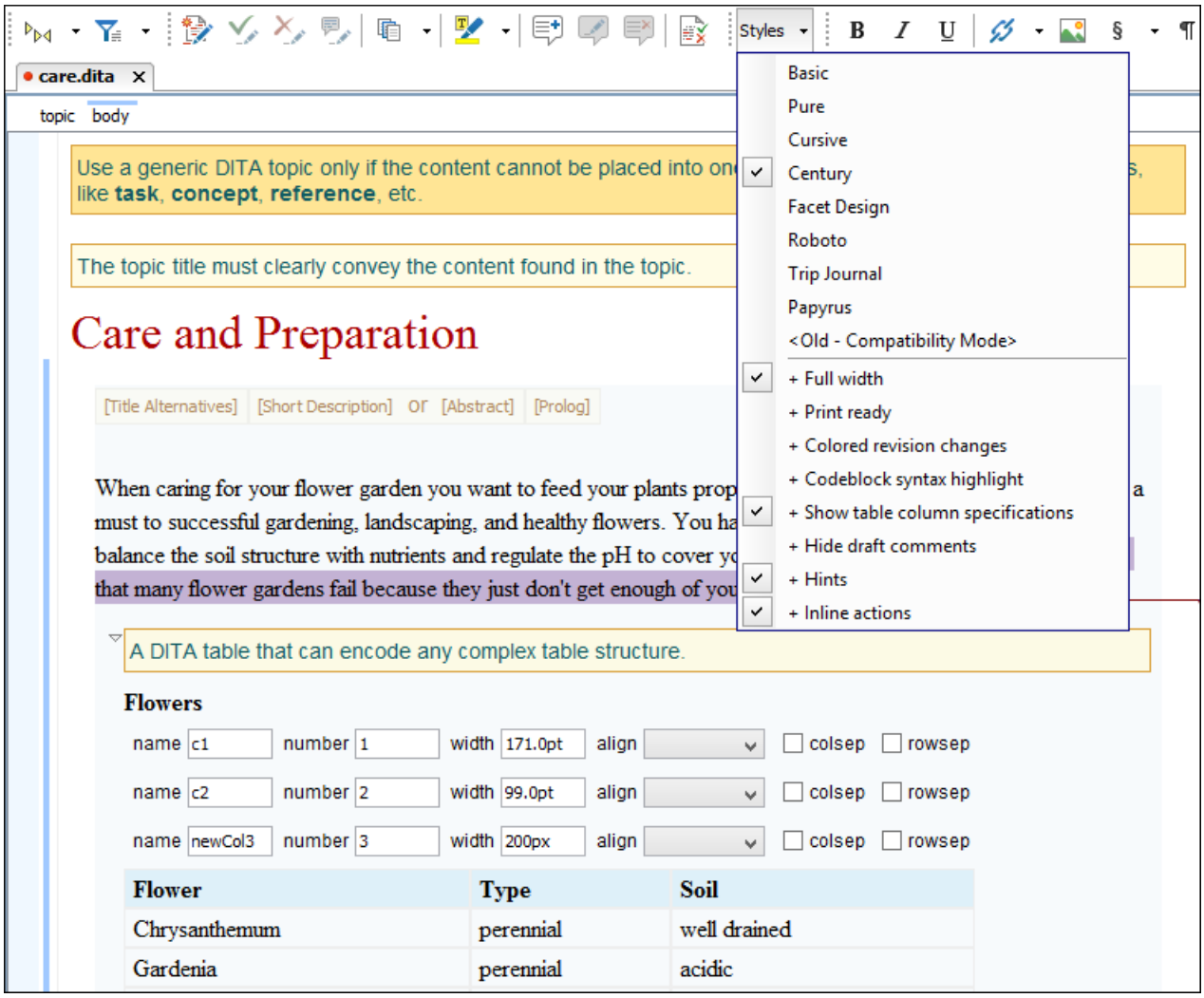


Figure 587: Styles Drop-down Menu in a DITA Document

Related Information:

CSS Subtab on page 61
Customizing the Main CSS of a Framework on page 1237

Handling CSS Imports

When a CSS document contains imports to other CSS documents, the references are also passed through the *XML Catalog* URI mappings to determine an indirect CSS referenced location.

Example: CSS Import

For example, if you can have a CSS import, such as:

```
@import "http://host/path/to/location/custom.css";
```

and then add your own *XML Catalog* file that maps the location to a custom CSS in the ***XML Catalog** preferences page*:

```
<uri name="http://host/path/to/location/custom.css"
      uri="path/to/custom.css"/>
```

Add a Custom Default CSS for Every XML Document

To add a custom CSS that is applied to every XML document, add a mapping in your *XML Catalog* file that looks like this:

```
<uri name="http://www.oxygenxml.com/extensions/author/css/userCustom.css"
      uri="path/to/custom.css"/>
```

This extra mapped CSS location will be parsed every time the application processes the CSS stylesheets used to render the opened XML document in the visual **Author** editing mode. This allows your custom CSS to be used without the need to modify all other CSS stylesheets contributed in the document type configuration.

Editor Variables in CSS Imports

You can use various *editor variables* in CSS imports. When editing an XML document with an associated CSS in **Author** mode, the editor variables will be expanded and resolved.

Example: Editor Variable in a CSS Import

For example, the following editor variable:

```
@import "${framework(DITA)}/custom.css";
```

is resolved in the DITA *framework* folder where the `custom.css` is placed.

oxygen Media Type

The CSS stylesheets can specify how a document is presented on different types of media (on the screen, paper, etc.) You can specify that some of the selectors from your CSS should be taken into account only in the Oxygen XML Editor **Author** mode and ignored in other media types. This can be accomplished by using the oxygen media type.

Example: *oxygen* Media Type

```
b{
 font-weight:bold;
 display:inline;
}

@media oxygen{
 b{
  text-decoration:underline;
 }
}
```

This example results in the text being bold if the document is opened in a web browser that does not recognize `@media oxygen`, while the text is bold and underlined when opened in Oxygen XML Editor **Author** mode.

You can also use the oxygen media type to specify CSS selectors to be applied in certain operating systems or platforms by using the `os` and `platform` properties. For example, you can specify one set of style rules for displaying Oxygen XML Editor in Windows, and another set of style rules for Mac OS. The supported properties are as follows:

- **os** - The possible values are: `win`, `linux`, or `mac`.
- **platform** - The possible values are: `standalone`, `eclipse`, or `webapp`.

Example: *os* and *platform* Properties

```
@media oxygen AND (os:"win") AND (platform:"standalone") {
    p{
        content:"PPP";
    }
}
```

Related Information:

@media Rule on page 1261

CSS At-Rules

Oxygen XML Editor supports some of the standard at-rules specified by CSS Level 2.1 and 3. The `@media` rule also include support for some style rules that are specific to Oxygen XML Editor.

@font-face At-Rule

Oxygen XML Editor allows you to use custom fonts in the **Author** mode by specifying them in the CSS using the `@font-face` media type. Only the `src` and `font-family` CSS properties can be used for this media type.

Example: *@font-face* Rule

```
@font-face{
    font-family:"Baroque Script";
    /*The location of the loaded TTF font must be relative to the CSS*/
    src:url("BaroqueScript.ttf");
}
```

@media Rule

The @media rule allows you to set different style rules for multiple types of media in the same stylesheet. For example, you can set the font size to be different on the screen than on paper. Oxygen XML Editor supports several media types, allowing you to set the style rules for presenting a document on various media (on screen, paper, etc.)

Supported Media Types

- `screen` - The styles marked with this media type are used only for rendering a document on screen.
- `print` - The styles marked with this media type are used only for printing a document.
- `all` - The styles marked with this media type are used for rendering a document in all supported types of media.
- `oxygen` - The styles marked with this media type are used only for rendering a document in the Oxygen XML Editor **Author** mode. For more information, see *oxygen Media Type* on page 1260.
- `oxygen-dark-theme` - The styles marked with this media type are used only for rendering a document in the Oxygen XML Editor **Author** mode when a dark theme is used (for example, *Graphite*).
- `oxygen-high-contrast-black` - The styles marked with this media type are used only for rendering a document in the Oxygen XML Editor **Author** mode on a Windows High Contrast Theme with a black background.
- `oxygen-high-contrast-white` - The styles marked with this media type are used only for rendering a document in the Oxygen XML Editor **Author** mode on a Windows High Contrast Theme with a white background.

Example: *@media* Rule

```
@media oxygen{
 b{
  text-decoration:underline;
 }
}
@media oxygen-high-contrast-white{
 b{
  font-weight:bold;
 }
}
```

Supported Properties

Oxygen XML Editor also supports a few properties to set specific style rules that depend upon the size of the visible area in **Author** mode. These supported properties are as follows:

- `min-width` - The styles selected in this property are applied if the visible area in **Author** mode is equal to or greater than the specified value.
- `max-width` - The styles selected in this property are applied if the visible area in **Author** mode is less than or equal to the specified value.

Example: *min-width* and *max-width* Properties

```
@media (min-width:500px){
 p{
    content:'XXX';
 }
}
@media (max-width:700px){
 p:after{
    content:'yyy';
 }
}
```

Related Information:

oxygen Media Type on page 1260

Standard W3C CSS Supported Features

Oxygen XML Editor supports most of the CSS Level 3 selectors and most of the CSS Level 2.1 properties

Supported CSS Selectors

The following table lists the CSS selectors that are supported in Oxygen XML Editor:

Expression	Name	CSS Level	Description / Example
`*`	Universal selector	CSS Level 2	Matches any element
`E`	Type selector	CSS Level 2	Matches any E element (i. e. an element with the local name E)
`E F`	Descendant selector	CSS Level 2	Matches any F element that is a descendant of an E element.
`E > F`	Child selectors	CSS Level 2	Matches any F element that is a child of an element E.
`E:lang(c)`	Language pseudo-class	CSS Level 2	Matches element of type E if it is in (human) language c (the document language specifies how language is determined).
`E + F`	Adjacent selector	CSS Level 2	Matches any F element immediately preceded by a sibling element E.
`E ~ F`	General sibling selector	CSS Level 3	Matches any F element preceded by a sibling element E.
`E[foo]`	Attribute selector	CSS Level 2	Matches any E element with the `"foo"` attribute set (whatever the value).
`E[foo="warning"]`	Attribute selector with value	CSS Level 2	Matches any E element whose `"foo"` attribute value is exactly equal to `"warning"`.
`E[foo~="warning"]`	Attribute selector containing value	CSS Level 2	Matches any E element whose `"foo"` attribute value is a list of space-separated values, one of which is exactly equal to `"warning"`.
`E[lang|="en"]`	Attribute selector containing hyphen separated values	CSS Level 2	Matches any E element whose `"lang"` attribute has a hyphen-separated list of values beginning (from the left) with `"en"`.
`E:before` and `E:after`	Pseudo elements	CSS Level 2	The `':before'` and `':after'` pseudo-elements can be used to

Expression	Name	CSS Level	Description / Example
			insert generated content before or after an element's content.
`E:before(n)` and `E:after(n)`	Pseudo elements	CSS Level 3	Multiple '`:before(n)`' and '`:after(n)`' pseudo-elements can be used to insert content before or after the content of an element (or other pseudo-element). For more information, see *the W3C CSS3 pseudo elements site*.
`E:first-child`	The first-child pseudo-class	CSS Level 2	Matches element E when E is the first child of its parent.
`E:not(s)`	Negation pseudo-class	CSS Level 2	An E element that does not match simple selector s.
`E:has`	Relational pseudo-class	CSS Level 4	The `:has()` relational pseudo-class is a functional pseudo-class that takes a relative selector as an argument. For more information, see *:has Relational Pseudo-Class* on page 1266.
`E:hover`	The hover pseudo-class	CSS Level 2	The `:hover` pseudo-class applies while the user designates an element with a pointing device, but does not necessarily activate it. When moving the pointing device over an element, all the parent elements up to the root are taken into account.
`E:focus`	The focus pseudo-class	CSS Level 2	The `:focus` pseudo-class applies while an element has the focus (accepts keyboard input).
`E:focus-within`	The generalized input focus pseudo-class	CSS Level 4	The `:focus-within` pseudo-class applies to elements for which the `:focus` pseudo-class applies. Additionally, the ancestors of an element that matches `:focus-within` also match.
`E#myid`	The ID selector	CSS Level 2	Matches any E element with ID equal to "myid". **Important:** Limitation: In Oxygen XML Editor the match is performed only taking into account the attributes with the exact name: *"id"*.
`E[att^="val"]`	Substring matching attribute selector	CSS Level 3	An E element whose `att` attribute value begins exactly with the string `val`.
`E[att$="val"]`	Substring matching attribute selector	CSS Level 3	An E element whose `att` attribute value ends exactly with the string `val`.

Expression	Name	CSS Level	Description / Example
`E[att*="val"]`	Substring matching attribute selector	CSS Level 3	An E element whose `att` attribute value contains the substring `val`.
`E:root`	Root pseudo-class	CSS Level 3	Matches the root element of the document. In HTML, the root element is always the HTML element.
`E:empty`	Empty pseudo-class	CSS Level 3	An E element that has no text or child elements.
`E:nth-child(n)`	The nth-child pseudo-class	CSS Level 3	An E element, the nth child of its parent.
`E:nth-last-child(n)`	The nth-last-child pseudo-class	CSS Level 3	An E element, the nth child of its parent, counting from the last one.
`E:nth-of-type(n)`	The nth-of-type pseudo-class	CSS Level 3	An E element, the nth sibling of its type.
`E:nth-last-of-type(n)`	The nth-last-of-type pseudo-class	CSS Level 3	An E element, the nth sibling of its type, counting from the last one.
`E:last-child`	The last-child pseudo-class	CSS Level 3	An E element, last child of its parent.
`E:first-of-type`	The first-of-type pseudo-class	CSS Level 3	An E element, first sibling of its type.
`E:last-of-type`	The last-of-type pseudo-class	CSS Level 3	An E element, last sibling of its type.
`E:only-child`	The only-child pseudo-class	CSS Level 3	An E element, only child of its parent.
`E:only-of-type`	The only-of-type pseudo-class	CSS Level 3	An E element, only sibling of its type.
`ns|E`	Element namespace selector	CSS Level 3	An element that has the local name E and the namespace given by the prefix `ns`. The namespace prefix can be bound to a URI by the at-rule: `@namespace ns "http://some_namespace_uri";` See *Namespace Selector* on page 1264.
`E!>F`	The subject selector	CSS Level 4 (experimental)	An element that has the local name E and has a child F. See *Subject Selector* on page 1266.

Namespace Selector

In the CSS 2.1 standard, the element selectors ignore the namespaces of the elements they are matching. Only the local name of the elements are considered in the selector matching process.

Oxygen XML Editor uses a different approach that is similar to the CSS Level 3 specification. If the element name from the CSS selector is not preceded by a namespace prefix it is considered to match an element with the same local name as the selector value and ANY namespace. Otherwise, the element must match both the local name and the namespace.

In CSS up to version 2.1 the name tokens from selectors are matching all elements from ANY namespace that have the same local name. Example:

```
<x:b xmlns:x="ns_x"/>
<y:b xmlns:y="ns_y"/>
```

Are both matched by the rule:

```
b {font-weight:bold}
```

Starting with CSS Level 3 you can create selectors that are namespace aware.

Example: Defining both prefixed namespaces and the default namespace

Given the namespace declarations:

```
@namespace sync "http://sync.example.org";
@namespace "http://example.com/foo";
```

In a context where the default namespace applies:

sync|A

represents the name A in the `http://sync.example.org` namespace.

|B

represents the name B that belongs to NO NAMESPACE.

***|C**

represents the name C in ANY namespace, including NO NAMESPACE.

D

represents the name D in the `http://example.com/foo` namespace.

Example: Defining only prefixed namespaces

Given the namespace declaration:

```
@namespace sync "http://sync.example.org";
```

Then:

sync|A

represents the name A in the `http://sync.example.org` namespace.

|B

represents the name B that belongs to NO NAMESPACE.

***|C**

represents the name C in ANY namespace, including NO NAMESPACE.

D

represents the name D in ANY namespace, including NO NAMESPACE.

Example: Defining prefixed namespaces combined with pseudo-elements

To match the `def` element its namespace will be declared, bind it to the *abs* prefix, and then write a CSS rule:

```
@namespace abs "http://www.oxygenxml.com/sample/documentation/abstracts";
```

Then:

abs|def

represents the name "def" in the `http://www.oxygenxml.com/sample/documentation/abstracts` namespace.

abs|def:before

represents the :before pseudo-element of the "def" element from the `http://www.oxygenxml.com/sample/documentation/abstracts` namespace.

Subject Selector

Oxygen XML Editor supports the subject selector described in CSS Level 4 (currently a working draft at W3C *http://www.w3.org/TR/selectors4/*). This selector matches a structure of the document, but unlike a compound selector, the styling properties are applied to the subject element (the one marked with "!") instead of the last element from the path.

The subject of the selector can be explicitly identified by appending an exclamation mark (!) to one of the compound selectors in a selector. Although the element structure that the selector represents is the same with or without the exclamation mark, indicating the subject in this way can change which compound selector represents the subject in that structure.

Example:

```
table! > caption {
  border: 1px solid red;
}
```

A border will be drawn to the table elements that contain a caption, as direct child.

This is different from:

```
table > caption {
  border: 1px solid red;
}
```

This draws a border around the caption.

Taking Processing Instructions into Account in CSS Subject Selectors

You can test for the existence of specific processing instructions (`PI`) in the child hierarchy of a subject selector.

For example:

```
@namespace oxy "http://www.oxygenxml.com/extensions/author";

chapter! > oxy|processing-instruction[important][level="high"]{
  color:red;
}
```

This would change the color of a DocBook chapter to red if it contains the `important` processing instruction:

```
<chapter>
   <title>A title</title>
   <?important level='high'?>
</chapter>
```

Descendant Selectors Limitation

Important: The current implementation has a known limitation. The general descendant selectors are taken into account as direct child selectors. For example, the following two CSS selectors are considered equivalent:

```
a! b c
```

and:

```
a! > b > c
```

Related Information:

:has Relational Pseudo-Class on page 1266

`:has` Relational Pseudo-Class

Oxygen XML Editor supports the CSS Level 4 subject selector (currently a working draft at W3C *http://www.w3.org/TR/selectors4/*), as described in *Subject Selector* on page 1266. Oxygen XML Editor also supports

the `:has` relational pseudo-class that has similar functionality and it can match an element by taking its child elements into account. For more information, see *https://drafts.csswg.org/selectors-4/#relational*.

You can create conditions that take into account the structure of the matching element.

Example: *has* Pseudo Class

```
table:has( tbody > thead){
 border: 1px solid red;
}
```

This example will result in a border being drawn for the table elements that contain at least a `thead` element in the `tbody` element.

Taking Processing Instructions into Account in CSS Subject Selectors

You can test for the existence of specific processing instructions (`PI`) in the child hierarchy of a subject selector.

For example:

```
@namespace oxy "http://www.oxygenxml.com/extensions/author";

chapter! > oxy|processing-instruction[important][level="high"]{
  color:red;
}
```

This would change the color of a DocBook chapter to red if it contains the `important` processing instruction:

```
<chapter>
   <title>A title</title>
   <?important level='high'?>
</chapter>
```

Descendant Selectors Limitation

Important: The current implementation has a known limitation. The general descendant selectors are taken into account as direct child selectors. For example, the following two CSS selectors are considered equivalent:

```
a! b c
```

and:

```
a! > b > c
```

Supported CSS Properties

Oxygen XML Editor validates all CSS 2.1 properties, but does not render *aural* and *paged* categories properties in **Author** mode, as well as some of the values of the *visual* category that are listed below under the **Ignored Values** column. For the Oxygen XML Editor-specific (extension) CSS properties, see *Oxygen XML Editor CSS Extensions* on page 1274.

Name	Rendered Values	Ignored Values
`'background-attachment'`	NONE	
`'background-color'`	`<color> | inherit`	`transparent`
`'background-image'`	`<uri> | none | inherit`	
`'background-position'`	`top | right | bottom | left | center`	`<percentage> | <length>`
`'background-repeat'`	`repeat | repeat-x | repeat-y | no-repeat | inherit`	
`'background'`	`background-color | background-image | background-position | background-repeat`	

Name	Rendered Values	Ignored Values
'border-collapse'	NONE	
'border-color'	<color> | inherit	transparent
'border-spacing'	NONE	
'border-style'	<border-style> | inherit	
'border-top' 'border-right' 'border-bottom' 'border-left'	[<border-width> || <border-style> || <border-color>] | inherit	
'border-top-color' 'border-right-color' 'border-bottom-color' 'border-left-color'	<color> | inherit	transparent
'border-top-style' 'border-right-style' 'border-bottom-style' 'border-left-style'	<border-style> | inherit	
'border-top-width' 'border-right-width' 'border-bottom-width' 'border-left-width'	<border-width> | inherit	
'border-width'	<border-width> | inherit	
'border'	[<border-width> || <border-style> || <border-color>] | inherit	
'bottom'	*<length>* | <percentage> | inherit	auto
'caption-side'	NONE	
'clear'	NONE	
'clip'	NONE	
'color'	<color> | inherit	
'content'	normal | none | [<string> | <URI> | <counter> | attr(<identifier>) | open-quote | close-quote]+ | inherit	no-open-quote | no-close-quote
'counter-increment'	[<identifier> <integer> ?]+ | none | inherit	
'counter-reset'	[<identifier> <integer> ?]+ | none | inherit	
'cursor'	NONE	
'direction'	ltr| rtl | inherit	
'display'	inline | block | list-item | table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell	run-in | inline-block | inline-table - considered block

Name	Rendered Values	Ignored Values
	| table-caption | none | inherit	
'empty-cells'	show | hide | inherit	
'float'	NONE	
'font-family'	[[<family-name> | <generic-family>] [, <family-name> | <generic-family>]*] | inherit	
'font-size'	<absolute-size> | <relative-size> | *<length>* | <percentage> | inherit	
'font-style'	normal | italic | oblique | inherit	
'font-variant'	NONE	
'font-weight'	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit	
'font'	[['font-style' || 'font-weight']? 'font-size' [/ 'line-height']? 'font-family'] | inherit	'font-variant' 'line-height' caption | icon | menu | message-box | small-caption | status-bar
'height'	NONE	
'left'	*<length>* | <percentage> | inherit	auto
'letter-spacing'	normal | *<length>* | inherit	
'line-height'	normal | <number> | *<length>* | <percentage> | inherit	
'list-style-image'	NONE	
'list-style-position'	NONE	
'list-style-type'	disc | circle | square | decimal | lower-roman | upper-roman | lower-latin | upper-latin | lower-alpha | upper-alpha | -oxy-lower-cyrillic-ru | -oxy-lower-cyrillic-uk | -oxy-upper-cyrillic-ru | -oxy-upper-cyrillic-uk | box | diamond | check | hyphen | none | inherit	lower-greek | armenian | georgian
'list-style'	['list-style-type'] | inherit	'list-style-position' || 'list-style-image'
'margin-right' 'margin-left'	<margin-width> | inherit | auto	

Name	Rendered Values	Ignored Values
'margin-top' 'margin-bottom'	<margin-width> | inherit	
'margin'	<margin-width> | inherit | auto	
'max-height'	NONE	
'max-width'	*<length>* | <percentage> | none | inherit - supported for *inline*, *block-level*, and replaced elements (such as images, tables, table cells)	
'min-height'	Absolute values, such as 230px, 1in, 7pt, 12em.	Values proportional to the parent element height, such as 30%
'min-width'	*<length>* | <percentage> | inherit - supported for *inline*, *block-level*, and replaced elements (such as images, tables, table cells)	
'outline-color'	[<color> | invert | inherit	
'outline-style'	[<border-style> | inherit	
'outline-width'	[<border-width> | inherit	
'outline'	[<outline-width> || <outline-style> || <outline-color>] | inherit	
'overflow'	NONE	
'padding-top' 'padding-right' 'padding-bottom' 'padding-left'	<padding-width> | inherit	
'padding'	<padding-width> | inherit	
'position'	absolute | fixed - supported for block display elements, relative - supported for *block* and *inline* display elements	absolute | fixed not supported for *inline* display elements
'quotes'	NONE	
'right'	*<length>* | <percentage> | inherit	auto
'table-layout'	auto	fixed | inherit
'text-align'	left | right | center | inherit	justify
'text-decoration'	none | [underline || overline || line-through] | inherit	blink
'text-decoration-style'	solid | double | dotted | dashed | wavy | inherit	

Name	Rendered Values	Ignored Values
'text-indent'	*<length>* | <percentage> | inherit	
'text-transform'	none | capitalize | uppercase | lowercase | inherit	
'top'	*<length>* | <percentage> | inherit	auto
'unicode-bidi'	bidi-override| normal| embed| inherit	
'vertical-align'	baseline | sub | super | top | text-top | middle | bottom | text-bottom | inherit	<percentage> | *<length>*
'visibility'	visible | hidden | inherit | -oxy-collapse-text	collapse
'white-space'	normal | pre | nowrap | pre-wrap | pre-line	
'width'	*<length>* | <percentage> | auto | inherit - supported for *inline*, *block-level*, and replaced elements (such as images, tables, table cells)	
'word-spacing'	NONE	
'z-index'	NONE	

<length> - Refers to distance measurements and is expressed in units such as mm, cm, in, em, rem, ex, pc, pt, px. For more information, see *the W3 CSS Level 3 length type specifications*.

Related Information:

Oxygen XML Editor CSS Extensions

Transparent Colors

CSS3 supports RGBA colors. The RGBA declaration allows you to set opacity (via the Alpha channel) as part of the color value. A value of 0 corresponds to a completely transparent color, while a value of 1 corresponds to a completely opaque color. To specify a value, you can use either a *real* number between 0 and 1, or a percent.

Example: RGBA Color

```
personnel:before {
    display:block;
    padding: 1em;
    font-size: 1.8em;
    content: "Employees";
    font-weight: bold;
    color:#EEEEEE;
    background-color: rgba(50, 50, 50, 0.6);
}
```

`attr()` Function: Properties Values Collected from the Edited Document

In CSS Level 2.1 you may collect attribute values and use them as content *only* for the pseudo-elements. For instance, the :before pseudo-element can be used to insert some content before an element. This is valid in CSS 2.1:

```
title:before{
  content: "[Audience Level: " attr(audience) "]";
```

```
}
```

If the `title` element from the XML document is:

```
<title audience="Expert">Changing the Timing Belt</title>
```

Then the title will be displayed as:

```
[Audience Level: Expert] Changing the Timimg Belt
```

In Oxygen XML Editor, the use of `attr()` function is available not only for the `content` property, but also for any other property. This is similar to the CSS Level 3 working draft: *http://www.w3.org/TR/2006/WD-css3-values-20060919/#functional*. The arguments of the function are:

`attr ( attribute_name , attribute_type , default_value )`

`attribute_name`

The attribute name. This argument is required.

`attribute_type`

The attribute type. This argument is optional. If it is missing, argument's type is considered `string`. This argument indicates what is the meaning of the attribute value and helps to perform conversions of this value. Oxygen XML Editor accepts one of the following types:

`color`

The value represents a color. The attribute may specify a color in various formats. Oxygen XML Editor supports colors specified either by name (`red`, `blue`, `green`, etc.) or as an RGB hexadecimal value `#FFEEFF`.

`url`

The value is a URL pointing to a media object. Oxygen XML Editor supports only images. The attribute value can be a complete URL, or a relative one to the XML document. Note that this URL is also resolved through the catalog resolver.

`integer`

The value must be interpreted as an integer.

`number`

The value must be interpreted as a float number.

`length`

The value must be interpreted as an integer.

`percentage`

The value must be interpreted relative to another value (length, size) expressed in percents.

`em`

The value must be interpreted as a size. 1 em is equal to the *font-size* of the relevant font.

`ex`

The value must be interpreted as a size. 1 ex is equal to the *height* of the **x** character of the relevant font.

`px`

The value must be interpreted as a size expressed in pixels relative to the viewing device.

`mm`

The value must be interpreted as a size expressed in millimeters.

`cm`

The value must be interpreted as a size expressed in centimeters.

`in`

The value must be interpreted as a size expressed in inches. 1 inch is equal to 2.54 centimeters.

`pt`

The value must be interpreted as a size expressed in points. The points used by CSS2 are equal to 1/72th of an inch.

`pc`

The value must be interpreted as a size expressed in picas. 1 pica is equal to 12 points.

`default_value`

This argument specifies a value that is used by default if the attribute value is missing. This argument is optional.

Example: *attr* Function

Consider the following XML instance:

```
<sample>
   <para bg_color="#AAAAFF">Blue paragraph.</para>
   <para bg_color="red">Red paragraph.</para>
   <para bg_color="red" font_size="2">Red paragraph with large font.</para>
   <para bg_color="#00AA00" font_size="0.8" space="4">
        Green paragraph with small font and margin.</para>
</sample>
```

The `para` elements have `bg_color` attributes with RGB color values (such as #AAAAFF). You can use the `attr()` function to change the elements appearance in the editor based on the value of this attribute:

```
background-color:attr(bg_color, color);
```

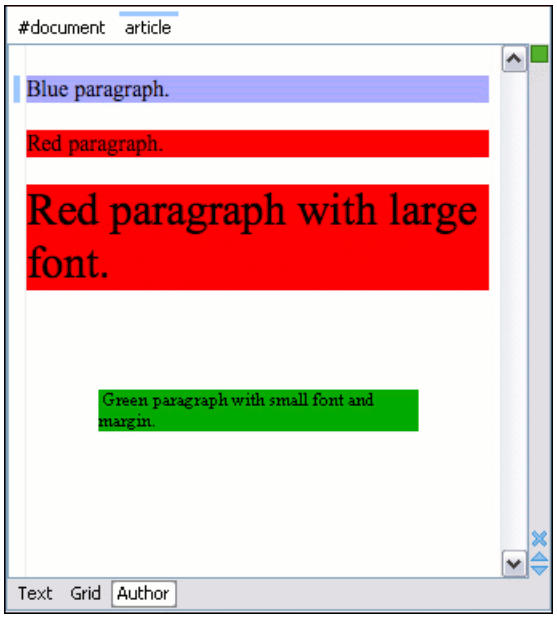
The attribute `font_size` represents the font size in *em* units. You can use this value to change the style of the element:

```
font-size:attr(font_size, em);
```

The complete CSS rule is:

```
para{
 display:block;
 background-color:attr(bg_color, color);
 font-size:attr(font_size, em);
 margin:attr(space, em);
}
```

The document is rendered as:



Oxygen XML Editor CSS Extensions

CSS stylesheets provide support for displaying documents. When editing non-standard documents, Oxygen XML Editor CSS extensions are useful.

Examples of how they can be used:

- Property for marking *foldable elements* in large files.
- Enforcing a display mode for the XML tags, regardless of the current mode selected by the user.
- Constructing a URL from a relative path location.
- String processing functions.

Built-in CSS Selectors

When Oxygen XML Editor renders content in the **Author** mode, it adds built-in CSS selectors (in addition to the CSS stylesheets linked in the XML or specified in the document type associated to the XML document). These built-in CSS selectors are processed before all other CSS content, but they can be overwritten if the CSS developer wants to modify a default behavior.

List of CSS Selector Contributed by Oxygen XML Editor

```
@namespace oxy "http://www.oxygenxml.com/extensions/author";
@namespace xi "http://www.w3.org/2001/XInclude";
@namespace xlink "http://www.w3.org/1999/xlink";
@namespace svg "http://www.w3.org/2000/svg";
@namespace mml "http://www.w3.org/1998/Math/MathML";

oxy|document {
    display:block !important;
}

oxy|cdata {
    display:-oxy-morph !important;
    white-space:pre-wrap !important;
    border-width:0px !important;
    margin:0px !important;
    padding: 0px !important;
}

oxy|processing-instruction {
    display:-oxy-morph !important;
    color: rgb(139, 38, 201) !important;
    white-space:pre-wrap !important;
    border-width:0px !important;
    margin:0px !important;
    padding: 0px !important;
}

oxy|processing-instruction[Pub],
oxy|processing-instruction[PubTbl],
oxy|processing-instruction[xm-replace_text],
```

```
oxy|processing-instruction[xm-deletion_mark],
oxy|processing-instruction[xm-insertion_mark_start],
oxy|processing-instruction[xm-insertion_mark_end],
oxy|processing-instruction[xml-model],
oxy|processing-instruction[xml-stylesheet]
{
    display:none !important;
}

oxy|comment {
    display:-oxy-morph !important;
    color: rgb(0, 100, 0) !important;
    background-color:rgb(255, 255, 210) !important;
    white-space:pre-wrap !important;
    border-width:0px !important;
    margin:0px !important;
    padding: 0px !important;
}

oxy|reference:before,
oxy|entity[href]:before{
  link: attr(href) !important;
  text-decoration: underline !important;
  color: navy !important;

  margin: 2px !important;
  padding: 0px !important;
  margin-right:0px !important;
  padding-right:2px !important;
}

oxy|reference:before {
  display: -oxy-morph !important;
  content: url(../images/editContent.gif) !important;
}

oxy|entity[href]:before{
  display: -oxy-morph !important;
  content: url(../images/editContent.gif) !important;
}

oxy|reference,
oxy|entity {
    -oxy-editable:false !important;
    background-color: rgb(240, 240, 240) !important;
    margin:0px !important;
    padding: 0px !important;

}

oxy|reference {
    display:-oxy-morph !important;
    /*EXM-28674 No need to present tags for these artificial references.*/
    -oxy-display-tags: none;
}

oxy|entity {
    display:-oxy-morph !important;
}

oxy|entity[name='amp'],
oxy|entity[name='lt'],
oxy|entity[name='gt'],
oxy|entity[name='quot'],
oxy|entity[name='apos']{
    /*EXM-32236, EXM-37026 Do not present tags for
simple character entity references.*/
    -oxy-display-tags: none;
}

oxy|entity[href] {
  border: 1px solid rgb(175, 175, 175) !important;
  padding: 0.2em !important;
}

xi|include {
    display:-oxy-morph !important;
    margin-bottom: 0.5em !important;
    padding: 2px !important;
}
xi|include:before,
xi|include:after{
    display:inline !important;
    background-color:inherit !important;
    color:#444444 !important;
    font-weight:bold !important;
}

xi|include:before {
    content:url(../images/link.png) attr(href)  !important;
    link: attr(href) !important;
}
xi|include[parse="text"]:before {
    content:url(../images/link.png) !important;
}
```

```
xi|include[xpointer]:before {
    content:url(../images/link.png) attr(href) " " attr(xpointer) !important;
    link: oxy_concat(attr(href), "#", attr(xpointer)) !important;
}

xi|fallback {
    display:-oxy-morph !important;
    margin: 2px !important;
    border: 1px solid #CB0039 !important;
}

xi|fallback:before {
    display:-oxy-morph !important;
    content:"XInclude fallback: " !important;
    color:#CB0039 !important;
}

oxy|doctype {
    display:block !important;
    background-color: transparent !important;
    color:blue !important;
    border-width:0px !important;
    margin:0px !important;
    padding: 2px !important;
}

@media oxygen-high-contrast-black, oxygen-dark-theme{
    oxy|doctype {
        color:#D0E2F4 !important;
    }
}

oxy|error {
    display:-oxy-morph !important;
    -oxy-editable:false !important;
    white-space:pre !important;
    font-weight:bold !important;
    color: rgb(178, 0, 0) !important;
    -oxy-display-tags: none;
}

oxy|error:before {
    content:url(../images/ReferenceError.png) "[" !important;
    color: rgb(178, 0, 0) !important;
}
oxy|error[level='warn']:before {
    content:url(../images/ReferenceWarn.png) "[" !important;
    color: rgb(200, 185, 0) !important;
}

oxy|error[level='warn'] {
    color: rgb(200, 185, 0) !important;
}

oxy|error:after {
    content:"]" !important;
}


*[xlink|href]:before {
    content:url(../images/link.png);
    link: attr(xlink|href) !important;
}


/*No direct display of the MathML and SVG images.*/
svg|svg{
 display:inline !important;
    white-space: -oxy-trim-when-ws-only !important;
}

svg|svg * {
    display:none !important;
    white-space:normal !important;
}

mml|math{
 display:inline !important;
    white-space: -oxy-trim-when-ws-only !important;
}
mml|math mml|*{
    display:none !important;
    white-space: normal !important;
}

/*Text direction attributes*/
*[dir='rtl'] { direction:rtl; unicode-bidi:embed; }
*[dir='rlo'] { direction:rtl; unicode-bidi:bidi-override; }

*[dir='ltr'] { direction:ltr; unicode-bidi:embed; }
*[dir='lro'] { direction:ltr; unicode-bidi:bidi-override; }

@media oxygen-high-contrast-black, oxygen-dark-theme{
    xi|include:before,
    xi|include:after{
```

```
        color:#808080 !important;
    }
}
```

Example:

To show all entities in the **Author** mode as transparent, without a gray background, first define in your CSS after all imports the namespace:

```
@namespace oxy "http://www.oxygenxml.com/extensions/author";
```

and then add the following selector:

```
oxy|entity {
    background-color: inherit !important;
}
```

Additional CSS Selectors

Oxygen XML Editor provides support for selecting additional types of nodes. These custom selectors apply to: *document*, *doctype sections*, *processing-instructions*, *comments*, *CDATA sections*, *reference sections*, and *entities*. *Processing-instructions* are not displayed by default. To display them, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Editor** > **Author**, and select **Show processing instructions**.

Note: The custom selectors are presented in the default CSS for **Author** mode and all of their properties are marked with an *!important* flag. For this reason, you have to set the *!important* flag on each property of the custom selectors from your CSS to be applicable.

For the custom selectors to work in your CSS stylesheets, declare the **Author** mode extensions namespace at the beginning of the stylesheet documents:

```
@namespace oxy url('http://www.oxygenxml.com/extensions/author');
```

- The `oxy|document` selector matches the entire document:

  ```
  oxy|document {
      display:block !important;
  }
  ```

- The following example changes the rendering of doctype sections:

  ```
  oxy|doctype {
      display:block !important;
      color:blue !important;
      background-color:transparent !important;
  }
  ```

- To match the processing instructions, you can use the `oxy|processing-instruction` selector:

  ```
  oxy|processing-instruction {
      display:block !important;
      color:purple !important;
      background-color:transparent !important;
  }
  ```

  A processing instruction usually has a target and one or more pseudo attributes:

  ```
  <?target_name data="b"?>
  ```

  You can match a processing instruction with a particular target from the CSS using the construct:

  ```
  oxy|processing-instruction[target_name]
  ```

  You can also match the processing instructions having a certain target and pseudo attribute value, such as:

  ```
  oxy|processing-instruction[target_name][data="b"]
  ```

- The XML comments display in **Author** mode can be changed using the `oxy|comment` selector:

  ```
  oxy|comment {
      display:block !important;
      color:green !important;
      background-color:transparent !important;
  ```

```
}
```

- The `oxy|cdata` selector matches CDATA sections:

```
oxy|cdata{
    display:block !important;
    color:gray !important;
    background-color:transparent !important;
}
```

- The `oxy|entity` selector matches the entities content:

```
oxy|entity {
    display:morph !important;
    editable:false !important;
    color:orange !important;
    background-color:transparent !important;
}
```

To match particular entities, use the `oxy|entity` selector in expressions such as:

```
oxy|entity[name='amp'],
oxy|entity[name='lt'],
oxy|entity[name='gt'],
oxy|entity[name='quot'],
oxy|entity[name='apos'],
oxy|entity[name^='#']{
    -oxy-display-tags: none;
}
```

- The *references to entities*, *XInclude*, and *DITA conrefs* and *conkeyrefs* are expanded by default in **Author** mode and the referenced content is displayed. The referenced resources are displayed inside the element or entity that refers to them.
  - You can use the `reference` property to customize the way these references are rendered in **Author** mode:

```
oxy|reference {
  border:1px solid gray !important;
}
```

In the **Author** mode, content is highlighted when text contains *comments* and changes (if *Track Changes* was active when the content was modified).

If this content is referenced, the **Author** mode does not display the highlighted areas in the new context. If you want to mark the existence of this comments and changes you can use the `oxy|reference[comments]`, `oxy|reference[changeTracking]`, and `oxy|reference[changeTracking][comments]` selectors.

Note: Two artificial attributes (`comments` and `changeTracking`) are set on the reference node, containing information about the number of comments and tracked changes in the content.

- The following example represents the customization of the reference fragments that contain comments:

```
oxy|reference[comments]:before {
  content: "Comments: " attr(comments) !important;
}
```

- To match reference fragments based on the fact that they contain tracked changes inside, use the `oxy|reference[changeTracking]` selector.

```
oxy|reference[changeTracking]:before {
  content: "Change tracking: " attr(changeTracking) !important;
}
```

- Here is an example of how you can set a custom color to the reference containing both tracked changes and comments:

```
oxy|reference[changeTracking][comments]:before {
  content: "Change tracking: " attr(changeTracking)
           " and comments: " attr(comments) !important;
}
```

Figure 588: Example: A Document Rendered Using these Rules

Additional CSS Properties

Oxygen XML Editor offers an extension of the standard CSS properties suited for content editing.

Folding Elements: -oxy-foldable

Oxygen XML Editor allows you to declare some elements to be *foldable*. This is especially useful when working with large documents organized in logical blocks, editing a large DocBook article or book, for instance. Oxygen XML Editor marks the *foldable* content with a small blue triangle. When you hover with your mouse pointer over this marker, a dotted line borders the collapsible content. The following actions are available in the **Folding** submenu of the contextual menu:

Toggle Fold

Toggles the state of the current fold.

Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on OS X))

Folds all the elements except the current element.

Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on OS X))

Folds the elements indented with one level inside the current element.

Expand Child Folds

Unfolds all child elements of the currently selected element.

Expand All (Ctrl + NumPad* (Command + NumPad* on OS X))

Unfolds all elements in the current document.

To define the element whose content can be *folded* by the user, you must use the property: `-oxy-foldable:true;`. To define the elements that are *folded* by default, use the `-oxy-folded:true` property.

Note: The `-oxy-folded` property works in conjunction with the `-oxy-foldable` property. Thus, the `folded` property is ignored if the `-oxy-foldable` property is not set on the same element.

When collapsing an element, it is useful to keep some of its content visible (for example, a short description of the collapsed region). The property `-oxy-not-foldable-child` is used to identify the child element that is kept visible. It accepts as value an element name or a list of comma separated element names. The first child element from the XML document that appears in the list of element names will be identified as the not *foldable* child and displayed. If the element is marked as *foldable* (`-oxy-foldable:true;`) but it doesn't have the property `-oxy-not-foldable-child` or none of the specified *non-foldable* children exists, then the element is still *foldable*. In this case the element kept visible when *folded* will be the `before` pseudo-element.

Note: Deprecated properties `foldable`, `not-foldable-child`, and `folded` are also supported.

Example: Folding DocBook Elements

All the elements below can have a `title` child element and are considered to be logical sections. You mark them as being *foldable* leaving the `title` element visible.

```
set,
book,
part,
reference,
chapter,
preface,
article,
sect1,
sect2,
sect3,
sect4,
section,
appendix,
figure,
example,
table {
    -oxy-foldable:true;
    -oxy-not-foldable-child: title;
}
```

Placeholders for Empty Elements: -oxy-placeholder-content

Oxygen XML Editor displays the element name as pseudo-content for empty elements if the ***Show placeholders for empty elements** option* is selected in the **Author** preferences page and there is no *before* or *after* content set in the CSS for this type of element. There are two CSS properties that can be used to control the placeholders (`-oxy-placeholder-content` and `-oxy-show-placeholder`).

`-oxy-placeholder-content` CSS Property

To control the displayed pseudo-content for empty elements, you can use the `-oxy-placeholder-content` CSS property.

The following example would change the `keyword` element to be displayed as `key`:

```
keyword{
    -oxy-placeholder-content:"key";
}
```

Note: This CSS property accepts the *${i18n(key)}* localization editor variable, as in the following example:

```
-oxy-placeholder-content:"${i18n(id)}";
```

`-oxy-show-placeholder` CSS Property

The `-oxy-show-placeholder` property allows you to decide whether or not the placeholder will be shown. The possible values are:

- `always` - Always display placeholders.
- `default` - Always display placeholders if *before* or *after* content are not set is CSS.
- `inherit` - The placeholders are displayed according to the ***Show placeholders for empty elements** option* (if `before` and `after` content is not declared).
- no - Never display placeholders.

Note: Deprecated properties `show-placeholder` and `placeholder-content` are also supported.

Read-only Elements: `-oxy-editable` property

If you want to inhibit editing a certain element content, you can set the `-oxy-editable` (deprecated property `editable` is also supported) CSS property to `false`.

Display Elements: `-oxy-morph` Value

Oxygen XML Editor allows you to specify that an element has an `-oxy-morph` display type (deprecated `morph` property is also supported), meaning that the element is *inline* if all its children are *inline*.

Example: *-oxy-morph* Property Value

Suppose you have a **wrapper** XML element that allows users to set a number of attributes on all sub-elements. This element should have an *inline* or *block* behavior, depending on the behavior of its child elements:

```
wrapper{
 display:-oxy-morph;
}
```

Whitespace Property: `-oxy-trim-when-ws-only` Value

Oxygen XML Editor allows you to set the `white-space` property to `-oxy-trim-when-ws-only`, meaning that the leading and trailing whitespaces are removed.

Visibility Property: -oxy-collapse-text Value

Oxygen XML Editor allows you to set the value of the `visibility` property to `-oxy-collapse-text`, meaning that the text content of that element is not rendered. If an element is marked as `-oxy-collapse-text` you are not able to position the cursor inside it and edit it. The purpose of `-oxy-collapse-text` is to make the text value of an element editable only through a form control.

Example: *visibility* Property

The text value of an XML element will be edited using a text field form control. In this case, we want the text content not to be directly present in the **Author** visual editing mode:

```
title{
 content: oxy_textfield(edit, '#text', columns, 40);
 visibility:-oxy-collapse-text;
}
```

Cyrillic Counters: -oxy-lower-cyrillic

Oxygen XML Editor allows you to set the value of the `list-style-type` property to Cyrillic counters. For example, `-oxy-lower-cyrillic-ru`, `-oxy-lower-cyrillic-uk`, `-oxy-upper-cyrillic-ru` or `-oxy-upper-cyrillic-uk`, meaning that you can have Russian and Ukrainian counters.

Example: Cyrillic Counters

Counting list items with Cyrillic symbols:

```
li{
  display:list-item;
  list-style-type:-oxy-lower-cyrillic-ru;
}
```

Link Property: -oxy-link

Oxygen XML Editor allows you to declare some elements to be *links*. This is especially useful when working with many documents that reference each other. The links allow for an easy way to get from one document to another. Clicking the link marker will open the referenced resource in an editor.

To define the element that should be considered a link, you must use the `link` property on the `before` or `after` pseudo element. The value of the property indicates the location of the linked resource. Since links are usually indicated by the value of an attribute in most cases it will have a value similar to `attr(href)`

Example: DocBook Link Elements

The following elements are defined to be links on the `before` pseudo element and their values are defined by the value of an attribute.

```
*[href]:before{
```

```
    -oxy-link:attr(href);
    content: "Click " attr(href) " for opening" ;
}

ulink[url]:before{
    -oxy-link:attr(url);
    content: "Click to open: " attr(url);
}

olink[targetdoc]:before{
    -oxy-link: attr(targetdoc);
    content: "Click to open: " attr(targetdoc);
}
```

Display Tag Markers: -oxy-display-tags

Oxygen XML Editor allows you to choose whether tag markers of an element should never be presented or the current display mode should be respected. This is especially useful when working with `:before` and `:after` pseudo-elements, in which case the element range is already visually defined so the tag markers are redundant.

The property is named `-oxy-display-tags`, with the following possible values:

- *none* - Tags markers must not be presented regardless of the current *display mode.*.
- *default* - The tag markers will be created depending on the current *display mode.*.
- *inherit* - The value of the property is inherited from an ancestor element.

```
-oxy-display-tags
    Value: none | default | inherit
    Initial: default
    Applies to: all nodes(comments, elements, CDATA, etc.)
    Inherited: false
    Media: all
```

Example: *-oxy-display-tags* Property

In this example, the **para** element from DocBook uses a `:before` and `:after` element and its tag markers will not be visible.

```
para:before{
    content: "{";
}

para:after{
    content: "}";
}

para{
    -oxy-display-tags: none;
    display:block;
    margin: 0.5em 0;
}
```

Append Content Properties: `-oxy-append-content` / `-oxy-prepend-content`

`-oxy-append-content` Property

This property appends the specified content to the content generated by other matching CSS rules of lesser specificity. Unlike the `content` property, where only the value from the rule with the greatest specificity is taken into account, the `-oxy-append-content` property adds content to that generated by the lesser specificity rules into a new compound content.

Example:

```
element:before{
    content: "Hello";
}
element:before{
    -oxy-append-content: " World!";
}
```

The content shown before the `element` will be `Hello World!`.

`-oxy-prepend-content` Property

Prepends the specified content to the content generated by other matching CSS rules of lesser specificity. Unlike the `content` property, where only the value from the rule with the greatest specificity is taken into account, the -

`oxy-prepend-content` prepends content to that generated by the lesser specificity rules into a new compound content.

Example:

```
element:before{
    content: "Hello!";
}
element:before{
    -oxy-prepend-content: "said: ";
}
element:before{
    -oxy-prepend-content: "I ";
}
```

The content shown before the `element` will be `I said: Hello!`.

Custom Colors for Element Tags: -oxy-tags-color

By default, Oxygen XML Editor does not display element tags. You can use the **Partial Tags** button from the **Author** toolbar to control the amount of *displayed markup.*

To configure the default background and foreground colors of the tags, *open the **Preferences** dialog box (Options > Preferences)*, go to **Editor** > **Edit modes** > **Author**, and set the desired colors in the ***Tags background color*** and ***Tags foreground color*** options.

If you want to be more specific and configure the colors using your CSS, the `-oxy-tags-background-color` and `-oxy-tags-color` properties allow you to control the background and foreground colors for any particular XML element.

Example:

```
para {
    -oxy-tags-color:white;
    -oxy-tags-background-color:green;
}
title {
    -oxy-tags-color:yellow;
    -oxy-tags-background-color:black;
}
```

Style Element Property: -oxy-style

Oxygen XML Editor allows you to specify the style for an XML element. This is helpful if you want to embed CSS styling to XML elements directly in the XML file you are editing without having to edit the CSS files that are normally attached to the XML files. The property should have an XPath function for the value.

Example: *-oxy-style* Property

The following code snippet should be added in the CSS file that renders the files for your framework customization:

```
*{
   -oxy-style:attr(style);
}
```

Suppose you want to display the `title` elements in your XML document in the color red. You could add the following snippet directly in the XML document:

```
<title style="color:red;">My Memoirs</title>
```

Tip: The `style` attribute is supported by default in HTML5 documents.

Custom CSS Functions

The visual **Author** editing mode supports also a wide range of custom CSS extension functions.

`oxy_local-name()` Function
This function evaluates the local name of the current node.

It does not have any arguments.

Example: *oxy_local-name* Function

To insert the local name as static text content before the element, use this CSS selector:

```
*:before{
  content: oxy_local-name() ": ";
}
```

`oxy_name()` Function

This function evaluates the qualified name of the current node.

It does not have any arguments.

Example: *oxy_name* Function

To insert a qualified name as static text content before the element, use this CSS selector:

```
*:before{
  content: oxy_name() ": ";
}
```

`oxy_url()` Function

This function extends the standard CSS **url()** function by allowing you to specify additional relative path components (parameters **loc_1** to **loc_n**).

Oxygen XML Editor uses all these parameters to construct an absolute location. Note that any of the parameters that are passed to the function can be either relative or absolute locations. These locations can be expressed as String objects, functions, or *editor variables* (built-in or custom).

`oxy_url ( base_location , loc_1 , loc_2 )`

base_location

String representing the base location. If not absolute, will be solved relative to the CSS file URL.

loc_1 ... loc_n (optional)

Strings representing relative location path components.

Examples: *oxy_url* Function

The following function receives String objects as input parameters:

```
oxy_url('http://www.oxygenxml.com/css/test.css', '../dir1/',
        'dir2/dir3/', '../../dir4/dir5/test.xml')
```

and returns:

```
'http://www.oxygenxml.com/dir1/dir4/dir5/test.xml'
```

The following function receives the result of the evaluation of two other functions as parameters (for instance, this is useful if you have image references and you want to see thumbnail images stored in the same folder):

```
image[href]{
 content:oxy_url(oxy_base-uri(), oxy_replace(attr(href),
    '.jpeg', 'Thumbnail.jpeg'));
}
```

The following function uses an *editor variable* as the first parameter to point to the Oxygen XML Editor installation location:

```
image[href] {
    content: oxy_url('${oxygenHome}', 'logo.png');
}
```

Related Information:

Editor Variables on page 169

`oxy_base-uri()` Function

This function evaluates the base URL in the context of the current node.

It does not have any arguments and takes into account the `xml:base` context of the current node. See the *XML Base specification* for more details.

Example: *oxy_base-uri* Function

Suppose you have some image references but you want to see other thumbnail images that reside in the same folder (in **Author** mode):

```
image[href]{
 content:oxy_url(oxy_base-uri(), oxy_replace(attr(href),
    '.jpeg', 'Thumbnail.jpeg'));
}
```

`oxy_parent-url()` Function

This function evaluates the parent URL of a URL received as string.

`oxy_parent-url` (`URL`)

URL

The URL as string.

`oxy_capitalize()` Function

This function capitalizes the first letter of the text received as argument.

`oxy_capitalize` (`text`)

text

The text in which the first letter will be capitalized.

Example: *oxy_capitalize* Function

```
*:before{
  content: oxy_capitalize(oxy_name()) ": ";
}
```

This would insert the capitalized qualified name as static text content before the element.

`oxy_uppercase()` Function

This function transforms to upper case the text received as argument.

`oxy_uppercase` (`text`)

text

The text to be capitalized.

Example: *oxy_uppercase* Function

To insert the upper-cased qualified name as static text content before the element, use this CSS selector:

```
*:before{
  content: oxy_uppercase(oxy_name()) ": ";
}
```

`oxy_lowercase()` Function

This function transforms to lower case the text received as argument.

`oxy_lowercase` (`text`)

text

The text to be lower cased.

Example: *oxy_lowercase* Function

To insert a lower-cased qualified name as static text content before the element, use this CSS selector:

```
*:before{
  content: oxy_lowercase(oxy_name()) ": ";
}
```

`oxy_concat()` Function

This function concatenates the received string arguments.

`oxy_concat` (`str_1` , `str_2`)

str_1 … str_n

The string arguments to be concatenated.

Example: *oxy_concat* Function

If an XML element has an attribute called **padding-left**:

```
<p padding-left="20">...
```

and you want to add a padding before it with that specific amount specified in the attribute value:

```
*[padding-left]{
 padding-left:oxy_concat(attr(padding-left), "px");
}
```

`oxy_replace()` Function

This function is used to replace a string of text.

The `oxy_replace()` function has two signatures:

- `oxy_replace ( text , target , replacement )`

  This function replaces each substring of the text that matches the literal target string with the specified literal replacement string.

  text

  The text in which the replace will occur.

  target

  The target string to be replaced.

  replacement

  The string replacement.

  EXAMPLE: Suppose that you have image references but you want to see other thumbnail images that reside in the same folder in the visual **Author** editing mode:

  ```
  image[href]{
   content:oxy_url(oxy_base-uri(), oxy_replace(attr(href),
      '.jpeg', 'Thumbnail.jpeg'));
  }
  ```

- `oxy_replace ( text , target , replacement , isRegExp )`

  This function replaces each substring of the text that matches the target string with the specified replacement string.

  text

  The text in which the replace will occur.

  target

  The target string to be replaced.

  replacement

  The string replacement.

  isRegExp

  If *true* the target and replacement arguments are considered regular expressions, if *false* they are considered literal strings.

  EXAMPLE: Suppose that you want to use a regular expression to replace all space sequences with an underscore:

  ```
  image[title]{
   content:oxy_replace(attr(title), "\\s+", "_", true)
  }
  ```

`oxy_unparsed-entity-uri()` Function

This function returns the URI value of an unparsed entity name.

`oxy_unparsed-entity-uri ( unparsedEntityName )`

unparsedEntityName

The name of an unparsed entity defined in the DTD.

This function can be useful to display images that are referenced with unparsed entity names.

Example: *oxy_unparsed-entity-uri* Function

CSS for displaying the image in Author for an *`imagedata`* with *`entityref`* to an unparsed entity:

```
imagedata[entityref]{
content: oxy_url(oxy_unparsed-entity-uri(attr(entityref)));
}
```

`oxy_attributes()` Function

This function concatenates the attributes for an element and returns the serialization.

`oxy_attributes ( )`

Example: *oxy_attributes* Function

```
element{
  content:oxy_attributes();
}
```

For instance, if you have the following XML fragment: `<element att1="x" xmlns:a="2" x="""/>`, the CSS function will display:

```
att1="x" xmlns:a="2" x="""
```

`oxy_substring()` Function

This function is used to return a string of text.

The `oxy_substring()` function has two signatures:

- `oxy_substring ( text , startOffset )`

  Returns a new string that is a substring of the original **text** string. It begins with the character at the specified index and extends to the end of **text** string.

  text
  : The original string.

  startOffset
  : The beginning index, inclusive

- `substring ( text , startOffset , endOffset )`

  Returns a new string that is a substring of the original **text** string. The substring begins at the specified **startOffset** and extends to the character at index **endOffset** - 1.

  text
  : The original string.

  startOffset
  : The beginning index, inclusive.

  endOffset
  : The ending index, exclusive.

Example: *oxy_substring* Function

`oxy_substring('abcd', 1)` returns the string `'bcd'`.

`oxy_substring('abcd', 4)` returns an empty string.

`oxy_substring('abcd', 1, 3)` returns the string `'bc'`.

If you only want to display part of an attribute value, for instance the part that comes before an **Appendix** string:

```
image[longdesc]{
 content: oxy_substring(attr(longdesc), 0,
          oxy_indexof(attr(longdesc), "Appendix"));
}
```

`oxy_getSomeText(text, length)` Function

This function allows you to truncate a long string and to set a maximum number of displayed characters.

The following properties are supported:

- **text** - Displays the actual text.
- **length** - Sets the maximum number of characters that are displayed.
- **endsWithPoints** - Specifies if the truncated text ends with ellipsis.

Example: *oxy_getSomeText* Function

If an attribute value is very large, you can trim its content before it is displayed as static content:

```
*[longdesc]:before{
  content: oxy_getSomeText(attr(longdesc), 200);
}
```

`oxy_indexof()` Function

This function is used to define searches.

The `oxy_indexof()` function has two signatures:

- `oxy_indexof ( text , toFind )`

  Returns the index within **text** string of the first occurrence of the **toFind** substring.

  text

  Text to search in.

  toFind

  The searched substring.

- `oxy_indexof ( text , toFind , fromOffset )`

  Returns the index within **text** string of the first occurrence of the **toFind** substring. The search starts from **fromOffset** index.

  text

  Text to search in.

  toFind

  The searched substring.

  fromOffset

  The index to start the search from.

Example: *oxy_indexof* Function

`oxy_indexof('abcd', 'bc')` returns 1.

`oxy_indexof('abcdbc', 'bc', 2)` returns 4.

If you only want to display part of an attribute value, for instance the part that comes before an **Appendix** string:

```
image[longdesc]{
 content: oxy_substring(attr(longdesc), 0,
          oxy_indexof(attr(longdesc), "Appendix"));
}
```

`oxy_lastindexof()` Function

This function is used to define last occurrence searches.

The `oxy_lastindexof()` function has two signatures:

- `oxy_lastindexof ( text , toFind )`

  Returns the index within **text** string of the rightmost occurrence of the **toFind** substring.

  text

  Text to search in.

toFind

The searched substring.

- `oxy_lastindexof ( text , toFind , fromOffset )`

  The search starts from **fromOffset** index. Returns the index within **text** string of the last occurrence of the **toFind** substring, searching backwards starting from the **fromOffset** index.

  text

  Text to search in.

  toFind

  The searched substring.

  fromOffset

  The index to start the search backwards from.

Example: *oxy_lastindexof* Function

`oxy_lastindexof('abcdbc', 'bc')` returns 4.

`oxy_lastindexof('abcdbccdbc', 'bc', 2)` returns 1.

If you only want to display part of an attribute value, for instance the part that comes before an **Appendix** string:

```
image[longdesc]{
 content: oxy_substring(attr(longdesc), 0,
          oxy_lastindexof(attr(longdesc), "Appendix"));
}
```

`oxy_xpath()` Function

This function is used to evaluate XPath expressions.

The `oxy_xpath()` function has the following signature:

- `oxy_xpath ( XPathExpression [, processChangeMarkers , value ] [, evaluate , value ])`

  It evaluates the given XPath 2.0 expression using Saxon 9 and returns the result. XPath expressions that depend on the cursor location can be successfully evaluated only when the cursor is located in the actual XML content. Evaluation fails when the current editing context is inside a referenced **xi:include** section or inside artificially referenced content (for example, DITA `conref` or `topicref` references).

  The parameters of the function are as follows:

  - A required expression parameter, which is the XPath expression to be evaluated.
  - An optional `processChangeMarkers` parameter, followed by its value, which can be either `true` or `false` (default value). When you set the parameter to `true`, the function returns the resulting text with all the change markers accepted (*delete* changes are removed and *insert* changes are preserved).
  - An optional `evaluate` parameter, followed by its value, which can be one of the following:
    - `dynamic` - Evaluates the XPath each time there are changes in the document.
    - `dynamic-once` - Separately evaluates the XPath for each node that matches the CSS selector. It will not re-evaluate the expression when changes are made to other nodes in the document. This will lead to improved performance, but the displayed content may not be updated to reflect the actual document content.
    - `static` - If the same XPath is evaluated on several nodes, the result for the first evaluation will be used for all other matches. Use this only if the XPath does not contain a relationship with the node on which the CSS property is evaluated. This will lead to improved performance, but the static displayed content may not be updated to reflect the actual document content.

Note: When XPath expressions are evaluated, the entities and `xi:include` elements are replaced with the actual content that is referenced. For example, consider the following code snippet:

```
<article>
  <xi:include href="section1.xml" xmlns:xi="http://www.w3.org/2001/XInclude"/>
</article>
```

where `section1.xml` contains the following content:

```
<section>
  <p>Referenced content</p>
</section>
```

The latter will be the actual content in which the XPath expression is executed.

Example: *oxy_xpath* Function

The following example counts the number of words from a paragraph (including *tracked changes*) and displays the result in front of it:

```
para:before{
  content:
   concat("|Number of words:",
    oxy_xpath(
        "count(tokenize(normalize-space(string-join(text(), '')), ' '))",
        processChangeMarkers,
       true),
    "| ");
}
```

Note: The `oxy_xpath()` function supports *editor variables*, as in the following example:

```
* {
    content:
    oxy_concat("Result: ",
        oxy_xpath('count(collection("${cfdu}/?select=*.xml"))')
    );
}
```

`oxy_action()` Function

This function allows you to define actions directly in the CSS, rather than referencing them from the associated framework.

The `oxy_action()` function is frequently used from *the `oxy_button()` function*.

The arguments received by the `oxy_action()` function are a list of properties that define an action. The following properties are supported:

- `name` - The name of the action. It will be displayed as the label for the button or menu item.
- `description` (optional) - A short description with details about the result of the action.
- `icon` (optional) - A path relative to the CSS pointing to an image (the icon for the action). The path can point to resources that are packed in Oxygen XML Editor (`oxygen.jar`) by starting its value with `/` (for example, `/images/Remove16.png`). It can also be expressed as an *editor variable*.
- `operation` - The name of the Java class implementing the *`ro.sync.ecss.extensions.api.AuthorOperation`* interface. There is also a variety of *predefined operations* that can be used.

  Note: If the name of the operation specified in the CSS is not qualified (has no Java package name), then it is considered to be one of the built-in Oxygen XML Editor operations from `ro.sync.ecss.extensions.commons.operations` package. If the class is not found in this package, then it will be loaded using the specified name.
- `arg-<string>` - All arguments with the `arg-` prefix are passed to the operation (the string that follows the `arg-` prefix is passed).
- `ID` - (optional) - The ID of the action from the *framework*. If this is specified, all others parameters are disregarded.

Example: *oxy_action* function inside an *oxy_button* form control:

```
oxy_button(
   action, oxy_action(
            name, 'Insert',
            description, 'Insert an element after the current one',
            icon, url('insert.png'),
            operation,
             'InsertFragmentOperation',
            arg-fragment, '<element>${caret}</element>',
            arg-insertLocation, '.',
            arg-insertPosition, 'After'),
   showIcon, true)
```

Example: *oxy_action* Function

You can also create a button form control directly from an `oxy_action` function:

```
oxy_action(
            name, 'Insert',
            description, 'Insert an element after the current one',
            operation, 'InsertFragmentOperation',
            arg-fragment, '<element>${caret}</element>',
            arg-insertLocation, '.',
            arg-insertPosition, 'After')
```

Tip: A code template is available to make it easy to add the `oxy_action` function with the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_action` code template.

Related Information:

Button Form Control on page 1296
The `oxy_button` built-in form control is used for graphical user interface objects that invoke a custom **Author** mode action (defined in the associated Document Type) referencing it by its ID, or directly in the CSS.

`oxy_action_list()` Function

This function allows you to define a list of actions directly in the CSS, rather than referencing them from the associated framework.

The `oxy_action_list()` function is used from *the `oxy_buttonGroup()` function*.

The arguments received by the `oxy_action_list()` function are a list of actions that are defined with *the `oxy_action()` function*. The following properties are supported in the `oxy_action_list()` function:

- `name` - The name of the action. It will be displayed as the label for the button or menu item.
- `description` (optional) - A short description with details about the result of the action.
- `icon` (optional) - A path relative to the CSS pointing to an image (the icon for the action). The path can point to resources that are packed in Oxygen XML Editor (`oxygen.jar`) by starting its value with `/` (for example, `/images/Remove16.png`). It can also be expressed as an *editor variable*.
- `operation` - The name of the Java class implementing the *`ro.sync.ecss.extensions.api.AuthorOperation`* interface. There is also a variety of *predefined operations* that can be used.

  Note: If the name of the operation specified in the CSS is not qualified (has no Java package name), then it is considered to be one of the built-in Oxygen XML Editor operations from `ro.sync.ecss.extensions.commons.operations` package. If the class is not found in this package, then it will be loaded using the specified name.
- `arg-<string>` - All arguments with the `arg-` prefix are passed to the operation (the string that follows the `arg-` prefix is passed).
- `ID` - (optional) - The ID of the action from the *framework*. If this is specified, all others parameters are disregarded.

Example: *oxy_action_list* Function

```
oxy_action_list(
  oxy_action(
    name, 'Insert',
    description,
        'Insert an element after the current one',
    operation, InsertFragmentOperation,
    arg-fragment, '<element></element>',
    arg-insertLocation, '.',
    arg-insertPosition, 'After'
  ),
  oxy_action(
    name, 'Delete',
    description, 'Deletes the current element',
    operation, DeleteElementOperation
  )
)
```

Tip: A code template is available to make it easy to add the `oxy_action_list` function with the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_action_list` code template.

Related Information:

oxy_action() Function on page 1290
This function allows you to define actions directly in the CSS, rather than referencing them from the associated framework.

Button Group Form Control on page 1297
The `oxy_buttonGroup` built-in form control is used for a graphical user interface group of buttons that invokes one of several custom **Author** mode actions (defined in the associated Document Type) referencing it by its ID, or directly in the CSS.

`oxy_label()` Function

This function can be used in conjunction with the CSS `content` property to change the style of generated text.

The arguments of the function are *property name* - *property value* pairs. The following properties are supported:

- `text` - This property specifies the built-in form control you are using.
- `width` - Specifies the width of the content area using relative (`em`, `ex`), absolute (`in`, `cm`, `mm`, `pt`, `pc`, `px`), and percentage (followed by the `%` character) length units. The `width` property takes precedence over the `columns` property (if the two are used together).
- `color` - Specifies the foreground color of the form control. If the value of the `color` property is `inherit`, the form control has the same color as the element in which it is inserted.
- `background-color` - Specifies the background color of the form control. If the value of the `background-color` property is `inherit`, the form control has the same color as the element in which it is inserted.
- `styles` - Specifies styles for the form control. The values of this property are a set of CSS properties:
  - `font-weight`, `font-size`, `font-style`, `font`
  - `text-align`, `text-decoration`
  - `width`
  - `color`, `background-color`
  - `link` - For more information about this property see *the `link` property section*.

```
element{
    content: oxy_label(text, "Label Text", styles,
      "font-size:2em;color:red;link:attr(href);");
}
```

Instead of using the values of the `styles` property individually, you can define them in a CSS file as in the following example:

```
* {
    width: 40%;
    text-align:center;
}
```

Then refer that file with an `@import` directive, as follows:

```
elem {
  content: oxy_label(text, 'my_label', styles, "@import 'labels.css';")
}
```

CAUTION: Extensive use of the `styles` property may lead to performance issues.

If the text from an `oxy_label()` function contains new lines, for example `oxy_label(text, 'LINE1\A LINE2', width, 100px)`, the text is split in two. Each of the two new lines has the specified width of 100 pixels.

Note: The text is split after `\A`, which represents a new line character.

You can use the `oxy_label()` function together with a *built-in form control* function to create a form control based layouts.

Example: *oxy_label* Function

An example of a use case is if you have multiple attributes on a single element and you want use form controls on separate lines and style them differently. Consider the following CSS rule:

```
person:before {
```

```
  content: "Name:*" oxy_textfield(edit, '@name', columns, 20)
    "\A Address:" oxy_textfield(edit, '@address', columns, 20)
}
```

Suppose you only want the **Name** label to be set to **bold**, while you want both labels aligned to look like a table (the first column with labels and the second with a text field). To achieve this, you can use the `oxy_label()` to style each label differently.

```
person:before {
 content: oxy_label(text, "Name:*", styles, "font-weight:bold;width:200px")
          oxy_textfield(edit, '@name', columns, 20) "\A "
          oxy_label(text, "Address:", styles, "width:200px")
          oxy_textfield(edit, '@address', columns, 20)
}
```

Tip: A code template is available to make it easy to add the `oxy_label` function with the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_label` code template..

`oxy_link-text()` Function

You can use this function on the CSS `content` property to obtain a text description from the source of a reference.

By default, the `oxy_link-text()` function resolves DITA and DocBook references. For further details about how you can also extend this functionality to other *frameworks*, go to *Configuring an Extensions Bundle*.

DITA Support

For DITA, the `oxy_link-text()` function resolves the `xref` element and the elements that have a `keyref` attribute. The text description is the same as the one presented in the final output for those elements. If you use this function for a `topicref` element that has the `navtitle` and `locktitle` attributes set, the function returns the value of the `navtitle` attribute.

DocBook Support

For DocBook, the `oxy_link-text()` function resolves the `xref` element that defines a link in the same document. The text description is the same as the one presented in the final output for those elements.

Example: *oxy_link-text* Function

For the following XML and associated CSS fragments the `oxy_link-text()` function is resolved to the value of the `xreflabel` attribute.

```
<para><code id="para.id" xreflabel="The reference label">my code</code>
      </para>
<para><xref linkend="para.id"/></para>
```

```
xref {
    content: oxy_link-text();
}
```

If the text from the target cannot extracted (for instance, if the `href` is not valid), you can use an optional argument to display fallback text.

```
*[class~="map/topicref"]:before{
    content: oxy_link-text("Cannot find the topic reference");
    link:attr(href);
}
```

`oxy_unescapeURLValue(string)` Function

This function returns the unescaped value of a URL-like string given as a parameter.

For example, if the value contains `%20` it will be converted to a simple space character.

Example: *oxy_unescapeURLValue* Function

`oxy_unescapeURLValue("http://www.example.com/a%20simple%20example.html")` returns the following value:

```
http://www.example.com/a simple example.html
```

Arithmetic Functions
Arithmetic Functions that are supported.

You can use any of the arithmetic functions implemented in the *java.lang.Math* class (*http://download.oracle.com/javase/6/docs/api/java/lang/Math.html*).

In addition, the following functions are available:

Syntax	Details
oxy_add (*param1*, ... , *paramN*, *'returnType'*)	Adds the values of all parameters from *param1* to *paramN*.
oxy_subtract (*param1*, *param2*, ... , *paramN*, *'returnType'*)	Subtracts the values of parameters *param2* to *paramN* from *param1*.
oxy_multiply (*param1*, ... , *paramN*, *'returnType'*)	Multiplies the values of parameters from *param1* to *paramN*.
oxy_divide (*param1*, *param2*, *'returnType'*)	Performs the division of *param1* to *param2*.
oxy_modulo (*param1*, *param2*, *'returnType'*)	Returns the reminder of the division of *param1* to *param2*.

Note: The *returnType* can be *'integer'*, *'number'*, or any of the supported CSS measuring types.

Example: *oxy_multiply* Function

If you have an image with **width** and **height** specified on it, this will compute the number of pixels on it:

```
image:before{
 content: "Number of pixels: " oxy_multiply(attr(width), attr(height), "px");
}
```

Form Controls

Oxygen XML Editor provides a variety of built-in form controls that allow users to interact with documents with familiar user interface objects. For customization purposes, Oxygen XML Editor also supports *custom form controls in Java*.

To watch our video demonstration in regards to form controls, go to *https://www.oxygenxml.com/demo/Form_Controls.html*.

Related Information:
Dynamically Add Form Controls Using a Styles Filter on page 1325

Audio File Player Form Control
The `oxy_audio` built-in form control is used for providing a mechanism to play audio clips.

The `oxy_audio` form control supports the following properties:

- `href` - The absolute or relative location of a resource. This property is mandatory. Relative values are resolved relative to the CSS. If you have media resources relative to the XML document, you can specify their paths like this:

  ```
  oxy_audio(href, oxy_url(oxy_base-uri(), 'ex.mp3')), width, 400px)
  ```

- `width` - Specifies the width of the content area using relative (`em`, `ex`), absolute (`in`, `cm`, `mm`, `pt`, `pc`, `px`), and percentage (followed by the % character) length units.

Example: *oxy_audio* Form Control

```
object {
    content:
        oxy_audio(
            href, 'resources/audio.mp3',
            width, 200px),
}
```

Tip: To insert a sample of the `oxy_audio` form control in a CSS file (or LESS file), invoke the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_audio` code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: ***[OXYGEN_INSTALL_DIR]*/samples/form-controls**.

Browser Form Control
The `oxy_browser` built-in form control is used for providing a mechanism to integrate HTML frames or interact with SVG documents directly in the **Author** mode editor. It can also be used to load HTML that executes JavaScript and from that JavaScript you can access the Oxygen XML Editor workspace. For example, you could use this method to change the value of an attribute, insert XML fragments, or open a new editor.

The `oxy_browser` form control supports the following properties:

- `href` - The absolute or relative location of a resource. This property is mandatory. Relative values are resolved relative to the CSS. If you have media resources relative to the XML document, you can specify their paths like this:

  ```
  oxy_browser(href, oxy_url(oxy_base-uri(), 'ex.svg')), width, 50%, height, 50%)
  ```

- `width` - Specifies the width of the content area using relative (`em`, `ex`), absolute (`in`, `cm`, `mm`, `pt`, `pc`, `px`), and percentage (followed by the % character) length units.
- `height` - Specifies the height of the form control area using relative (`em`, `ex`), absolute (`in`, `cm`, `mm`, `pt`, `pc`, `px`), and percentage (followed by the % character) length units.

Example: *oxy_browser* Form Control

```
object {
    content:
        oxy_browser(
            href, 'http://example.page',
            width, 600px,
            height, 400px),
}
```

Tip: To insert a sample of the `oxy_browser` form control in a CSS file (or LESS file), invoke the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_browser` code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: ***[OXYGEN_INSTALL_DIR]*/samples/form-controls**.

Interacting with the Oxygen XML Editor Workspace

The `oxy_browser` form control also provides the possibility of creating custom form control without having to use the Java-based API. You can use the `oxy_browser` form control to load HTML that executes JavaScript. In the JavaScript, you can use some predefined global variables that provide a gateway between the JavaScript and the Oxygen XML Editor Java API. This allows you to perform changes in the document, open resources, and more, solely from the JavaScript.

Important: This will only work if the loaded HTML is located inside a *framework or plugin directory*, such as: *[OXYGEN_INSTALL_DIR]*`/frameworks/` or *[OXYGEN_INSTALL_DIR]*`/plugins/`.

The following global variables can be used:

- **authorAccess** - This object is an instance of *ro.sync.ecss.extensions.api.AuthorAccess*.
- **contextElement** - An instance of *ro.sync.ecss.extensions.api.node.AuthorNode*. The form control is added over this node.
- **pluginWorkspace** - An instance of *ro.sync.exml.workspace.api.standalone.StandalonePluginWorkspace*.
- **fcArguments** - A `java.util.Map` implementation with the properties (name and value pairs) passed on the form control function.
- **apiHelper** - A helper object for creating Java objects. It allows you to create Java objects from within the JavaScript code. These objects can then be passed to the Java methods as in the following example:

  ```
  var newAttrValue = apiHelper.newInstance(
              "ro.sync.ecss.extensions.api.node.AttrValue",
              ["normalizedValue", "rawValue", true]);
  authorAccess.getDocumentController().setAttribute(
          "counter", newAttrValue, contextElement);
  ```

For more information, open the `form-controls.xml` file in the ***[OXYGEN_INSTALL_DIR]*/samples/form-controls** directory and go to section **11.1 - Interacting with the Oxygen Workspace**. For debugging information, continue reading below.

Debugging JavaScript Used for Custom Form Controls

If you encounter unexpected results when using the *method described above*, you can debug the script by using the following guidelines:

- Calls to `alert("message.to.present")` or `console.log("message.to.present")` will be presented in the *Results panel*.
- You can install the ***Firebug** extension* by executing the following script:

```
{code:javascript}
function installFB() {
  if (!document.getElementById('FirebugLite')) {
    E = document['createElement' + 'NS'] && document.documentElement.namespaceURI;
    E = E ? document['createElement' + 'NS'](E, 'script')
          : document['createElement']('script');
    E['setAttribute']('id', 'FirebugLite');
    E['setAttribute']('src',
                      'https://getfirebug.com/' + 'firebug-lite.js' + '#startOpened');
    E['setAttribute']('FirebugLite', '4');
    (document['getElementsByTagName']('head')[0]
          || document['getElementsByTagName']('body')[0]).appendChild(E);
    E = new Image;
    E['setAttribute']('src', 'https://getfirebug.com/' + '#startOpened');
  }
}
{code}
```

Note: To force the *Browser Form Control* to reload after making changes to the JavaScript file, you need to use the **Reload page** action from the form control's contextual menu.

Button Form Control
The `oxy_button` built-in form control is used for graphical user interface objects that invoke a custom **Author** mode action (defined in the associated Document Type) referencing it by its ID, or directly in the CSS.

The `oxy_button` form control supports the following properties:

- `actionContext` - Specifies the context in which the action associated with the form control is executed. Its possible values are `element` (default value) and `caret`. If you select the `element` value, the context is the element that holds the form control. If you select the `caret` value, the action is invoked at the cursor location. If the cursor is not inside the element that holds the form control, the `element` value is selected automatically.
- `fontInherit` - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be `true` or `false` (default value). To make the form control inherit its font from its parent element, set the `fontInherit` property to `true`.
- `color` - Specifies the foreground color of the form control. If the value of the `color` property is `inherit`, the form control has the same color as the element in which it is inserted.
- `actionID` - The ID of the action, specified in the *document type association*, that is invoked when you click the button.

  Note: The element that contains the form control represents the context where the action is invoked.
- `action` - Defines an action directly, rather than using the `actionID` parameter to reference an action from the *document type association*. This property is defined using the *`oxy_action` function*.

```
oxy_button(action, oxy_action(
          name, 'Insert',
          description, 'Insert an element after the current one',
          icon, url('insert.png'),
          operation, 'InsertFragmentOperation',
          arg-fragment, '<element>${caret}</element>',
          arg-insertLocation, '.',
          arg-insertPosition, 'After'
))
```

Tip: You can also create a button form control *directly from an `oxy_action` function*.

- `visible` - Specifies whether or not the form control is visible. The possible values of this property are `true` (default value) and `false`.

- `transparent` - Flattens the aspect of the button form control, removing its border and background. The values of this property can be `true` or `false` (default value).
- `showText` - Specifies if the action text should be displayed on the button form control. If this property is missing then the button displays the icon only if it is available, or the text if the icon is not available. The values of this property can be `true` or `false`.

```
element {
  content: oxy_button(actionID, 'remove.attribute', showText, true);
}
```

- `showIcon` - Specifies if the action icon should be displayed on the button form control. If this property is missing then the button displays the icon only if it is available, or the text if the icon is not available. The values of this property can be `true` or `false`.

```
element {
  content: oxy_button(actionID, 'remove.attribute', showIcon, true);
}
```

- `enableInReadOnlyContext` - To enable *button form controls* or *groups of buttons form controls* this property needs to be set to `true`. This property can be used to specify areas as *read-only* (by setting the `-oxy-editable` property to `false`). This is useful when you want to use an action that does not modify the context.
- `hoverPseudoclassName` - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```
p:before {
  content: oxy_button(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
  border: 1px solid red;
}
```

Example: *oxy_button* Form Control

```
button:before {
  content: "Label:"
     oxy_button(
         /* This action is declared in the document type
            associated with the XML document. */
         actionID, "insert.popupWithMultipleSelection");
}
```

Tip: To insert a sample of the `oxy_button` form control in a CSS file (or LESS file), invoke the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_button` code template. Also, an `oxy_button_in_place_action` code template is available that inserts an `oxy_button` function that includes an `action` parameter.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: ***[OXYGEN_INSTALL_DIR]*/samples/form-controls**.

Button Group Form Control
The `oxy_buttonGroup` built-in form control is used for a graphical user interface group of buttons that invokes one of several custom **Author** mode actions (defined in the associated Document Type) referencing it by its ID, or directly in the CSS.

The `oxy_buttonGroup` form control supports the following properties:

- `actionIDs` - The IDs of the actions that will be presented in the group of buttons.
- `actionID` - The ID of the action, specified in the *document type association*, that is invoked when you click the button.

  Note: The element that contains the form control represents the context where the action is invoked.
- `action_list` - Defines a list of actions directly, rather than using the `actionID` parameter to reference actions from the *document type association*. This property is defined using the *`oxy_action_list` function*.

```
oxy_buttonGroup(
    label, 'A group of actions',
    icon, url('http://www.oxygenxml.com/img/icn_oxy20.png'),
    actions,
      oxy_action_list(
```

```
            oxy_action(
                  name, 'Insert',
                  description, 'Insert an element after the current one',
                  operation, 'InsertFragmentOperation',
                  arg-fragment, '<element></element>',
                  arg-insertLocation, '.',
                  arg-insertPosition, 'After'
            ),
            oxy_action(
                  name, 'Delete',
                  description, 'Deletes the current element',
                  operation, 'DeleteElementOperation'
            )
      )
  )
```

Tip: A code template is available to make it easy to add the `oxy_action_list` function.

- `label` - Specifies the label to be displayed on the button. This label can be translated using the *`${i18n()}` editor variable*.
- `icon` - The path to the icon to be displayed on the button.
- `actionContext` - Specifies the context in which the action associated with the form control is executed. Its possible values are `element` (default value) and `caret`. If you select the `element` value, the context is the element that holds the form control. If you select the `caret` value, the action is invoked at the cursor location. If the cursor is not inside the element that holds the form control, the `element` value is selected automatically.
- `visible` - Specifies whether or not the form control is visible. The possible values of this property are `true` (default value) and `false`.
- `actionStyle` - Specifies what to display for an action in the form control. The values of this property can be `text` (default value), `icon`, or both.
- `tooltip` - Specifies a tooltip to be displayed when you hover over the form control.
- `transparent` - Makes the button transparent without any borders or background colors. The values of this property can be `true` or `false`.
- `fontInherit` - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be `true` or `false` (default value). To make the form control inherit its font from its parent element, set the `fontInherit` property to `true`.
- `enableInReadOnlyContext` - To enable *button form controls* or *groups of buttons form controls* this property needs to be set to `true`. This property can be used to specify areas as *read-only* (by setting the `-oxy-editable` property to `false`). This is useful when you want to use an action that does not modify the context.
- `hoverPseudoclassName` - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```
p:before {
  content: oxy_buttonGroup(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
  border: 1px solid red;
}
```

Example: *oxy_buttonGroup* Form Control

```
buttongroup:before {
  content:
    oxy_label(text, "Button Group:", width, 150px, text-align, left)
    oxy_buttonGroup(
       label, 'A group of actions',
       /* The action IDs are declared in the document type
          associated with the XML document. */
       actionIDs,
          "insert.popupWithMultipleSelection,insert.popupWithSingleSelection",
       actionStyle, "both");
}
```

Tip: To insert a sample of the `oxy_buttonGroup` form control in a CSS file (or LESS file),, invoke the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_buttonGroup` code template. Also, an `oxy_buttonGroup_in_place_action` code template is available that inserts an `oxy_buttonGroup` function that includes an `oxy_action_list` function.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: ***[OXYGEN_INSTALL_DIR]*/samples/form-controls**.

Checkbox Form Control
The `oxy_checkbox` built-in form control is used for a graphical user interface box that you can click to enable or disable an option. A single checkbox or multiple checkboxes can be used to present and edit the value on an attribute or element.

The `oxy_checkbox` form control supports the following properties:

- `edit` - Lets you edit the value of an attribute, the text content of an element, or Processing Instructions (PI). This property can have the following values:
  - **@attribute_name** - The name of the attribute whose value is being edited. If the attribute is in a namespace, the value of the property must be a *QName* and the CSS must have a namespace declaration for the prefix.
  - **#text** - Specifies that the presented/edited value is the simple text value of an element.

    Note: You can set the value of the `visibility` property to `-oxy-collapse-text` to render the text only in the form control that the `oxy_editor` function specifies.
- `resultSeparator` - If multiple check-boxes are used, the separator is used to compose the final result. If not specified, the *space* character is used.
- `tooltips` - Associates tooltips to each value in the `values` property. The value of this property is a list of tooltip messages separated by commas. If you want the tooltip to display a comma, use the `${comma}` variable.
- `visible` - Specifies whether or not the form control is visible. The possible values of this property are `true` (default value) and `false`.
- `values` - Specifies the values that are committed when the check-boxes are selected. If these values are not specified in the CSS, they are collected from the associated XML Schema.

  Note: Typically, when you use a comma in the values of a form control, the content that follows a comma is considered a new value. If you want to include a comma in the values, precede the comma with two backslashes. For example, `(values, '1\\, 2\\, 3, 4, edit, false)` will display a form control that has `1, 2, 3` for the first value and 4 for the second value.
- `fontInherit` - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be `true` or `false` (default value). To make the form control inherit its font from its parent element, set the `fontInherit` property to `true`..
- `uncheckedValues` - Specifies the values that are committed when check-boxes are not selected.
- `labels` - This property must have the same number of items as the `values` property. Each item provides a literal description of the items listed in the `values` property. These labels can be translated using the *`${i18n()}` editor variable*.. If this property is not specified, the `values` property is used as the label.
- `columns` - Controls the width of the form control. The unit size is the width of the **w** character.
- `color` - Specifies the foreground color of the form control. If the value of the `color` property is `inherit`, the form control has the same color as the element in which it is inserted.
- `hoverPseudoclassName` - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```
p:before {
  content: oxy_checkbox(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
  border: 1px solid red;
}
```

Example: Single *oxy_checkbox* Form Control

```
checkBox[attribute]:before {
    content: "A check box editor that edits a two valued attribute (On/Off)
              The values are specified in the CSS:"
        oxy_checkbox(
        edit, "@attribute",
        values, "On",
        uncheckedValues, "Off",
        labels, "On/Off");
}
```

Example: Multiple *oxy_checkbox* Form Controls

```
multipleCheckBox[attribute]:before {
    content: "Multiple checkboxes editor that edits an attribute value.
              Depending whether the check-box is selected,
              a different value is committed:"
        oxy_checkbox(
        edit, "@attribute",
        values, "true, yes, on",
        uncheckedValues, "false, no, off",
        resultSeparator, ","
        labels, "Present, Working, Started");
}
```

Tip: To insert a sample of the `oxy_checkbox` form control in a CSS file (or LESS file), invoke the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_checkbox` code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: ***[OXYGEN_INSTALL_DIR]*/samples/form-controls**.

Combo Box Form Control
The `oxy_combobox` built-in form control is used for providing a graphical user interface object that is a drop-down menu of proposed values. This form control can also be used for a combination of a drop-down menu and an editable single-line text field.

The `oxy_combobox` form control supports the following properties:

- `edit` - Lets you edit the value of an attribute, the text content of an element, or Processing Instructions (PI). This property can have the following values:
  - **@attribute_name** - The name of the attribute whose value is being edited. If the attribute is in a namespace, the value of the property must be a *QName* and the CSS must have a namespace declaration for the prefix.
  - **#text** - Specifies that the presented/edited value is the simple text value of an element.

    Note: You can set the value of the `visibility` property to `-oxy-collapse-text` to render the text only in the form control that the `oxy_editor` function specifies.
- `columns` - Controls the width of the form control. The unit size is the width of the **w** character.
- `width` - Specifies the width of the content area using relative (`em`, `ex`), absolute (`in`, `cm`, `mm`, `pt`, `pc`, `px`), and percentage (followed by the % character) length units. The `width` property takes precedence over the `columns` property (if the two are used together).
- `visible` - Specifies whether or not the form control is visible. The possible values of this property are `true` (default value) and `false`.
- `editable` - This property accepts the **true** and **false** values. In addition to a drop-down menu, the **true** value also generates an editable text field box that allows you to insert other values than the proposed ones. The **false** value generates a drop-down menu that only accepts the proposed values.
- `tooltips` - Associates tooltips to each value in the `values` property. The value of this property is a list of tooltip messages separated by commas. If you want the tooltip to display a comma, use the `${comma}` variable.
- `values` - Specifies the values that populate the list of proposals. If these values are not specified in the CSS, they are collected from the associated XML Schema..

  Note: Typically, when you use a comma in the values of a form control, the content that follows a comma is considered a new value. If you want to include a comma in the values, precede the comma with two backslashes. For example, `(values, '1\\, 2\\, 3, 4, edit, false)` will display a form control that has `1, 2, 3` for the first value and 4 for the second value.
- `fontInherit` - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be `true` or `false` (default value). To make the form control inherit its font from its parent element, set the `fontInherit` property to `true`.
- `labels` - This property must have the same number of items as the `values` property. Each item provides a literal description of the items listed in the `values` property. These labels can be translated using the *`${i18n()}` editor variable*.

  Note: This property is only available for read-only combo boxes (the `editable` property is set to `false`).
- `color` - Specifies the foreground color of the form control. If the value of the `color` property is `inherit`, the form control has the same color as the element in which it is inserted.

- `hoverPseudoclassName` - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```
p:before {
  content: oxy_combobox(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
  border: 1px solid red;
}
```

- `canRemoveValue` - If the value is set to `true` and the combo box is not editable, then a new `<Empty>` value is added in that combo box. This clears or removes the value being edited, depending on if it edits an element or attribute.

Example: *oxy_combobox* Form Control

```
comboBox:before {
    content: "A combo box that edits an attribute value.
                The possible values are provided from CSS:"
        oxy_combobox(
            edit, "@attribute",
            editable, false,
            values, "value1, value2, value3",
            labels, "Value no1, Value no2, Value no3");
}
```

Tip: To insert a sample of the `oxy_combobox` form control in a CSS file (or LESS file), invoke the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_combobox` code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: ***[OXYGEN_INSTALL_DIR]*/samples/form-controls**.

Date Picker Form Control
The `oxy_datePicker` built-in form control is used for offering a text field with a calendar browser that allows the user to choose a certain date in a specified format.

The `oxy_datePicker` form control supports the following properties:

- `edit` - Lets you edit the value of an attribute, the text content of an element, or Processing Instructions (PI). This property can have the following values:
  - **@attribute_name** - The name of the attribute whose value is being edited. If the attribute is in a namespace, the value of the property must be a *QName* and the CSS must have a namespace declaration for the prefix.
  - **#text** - Specifies that the presented/edited value is the simple text value of an element.

    Note: You can set the value of the `visibility` property to `-oxy-collapse-text` to render the text only in the form control that the `oxy_editor` function specifies.
- `columns` - Controls the width of the form control. The unit size is the width of the **w** character.
- `width` - Specifies the width of the content area using relative (`em`, `ex`), absolute (`in`, `cm`, `mm`, `pt`, `pc`, `px`), and percentage (followed by the % character) length units. The `width` property takes precedence over the `columns` property (if the two are used together).
- `color` - Specifies the foreground color of the form control. If the value of the `color` property is `inherit`, the form control has the same color as the element in which it is inserted.
- `format` - This property specifies the format of the inserted date. The pattern value must be a valid Java date (or date-time) format. If missing, the type of the date is determined from the associated schema.
- `visible` - Specifies whether or not the form control is visible. The possible values of this property are `true` (default value) and `false`.
- `validateInput` - Specifies if the form control is validated. If you introduce a date that does not respect the format, the `datePicker` form control is rendered with a red foreground. By default, the input is validated. To disable the validation, set this property to `false`.
- `hoverPseudoclassName` - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```
p:before {
  content: oxy_datePicker(hoverPseudoclassName, 'showBorder')
}
```

```
p:showBorder {
  border: 1px solid red;
}
```

Example: *oxy_datePicker* Form Control

```
date {
    content:
        oxy_label(text, "Date time attribute with
                    format defined in CSS: ", width, 300px)
        oxy_datePicker(
            columns, 16,
            edit, "@attribute",
            format, "yyyy-MM-dd");
}
```

Tip: To insert a sample of the `oxy_datePicker` form control in a CSS file (or LESS file), invoke the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_datePicker` code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: ***[OXYGEN_INSTALL_DIR]*/samples/form-controls**.

HTML Content Form Control
The `oxy_htmlContent` built-in form control is used for rendering HTML content. This HTML content is displayed as a graphical element shaped as a box. The shape of the box is determined by a given width and the height is computed based upon the length of the text.

The `oxy_htmlContent` form control supports the following properties:

- `href` - The absolute or relative location of a resource. The resource needs to be a well-formed HTML file.
- `id` - The unique identifier of an item. This is a `div` element that has a unique `id` and is a child of the `body` element. The `div` element is the container of the HTML content to be rendered by the form control.
- `content` - An alternative to the `href` and `id` pair of elements. It provides the HTML content that will be displayed in the form control.
- `width` - Specifies the width of the content area using relative (`em`, `ex`), absolute (`in`, `cm`, `mm`, `pt`, `pc`, `px`), and percentage (followed by the % character) length units. The `width` property takes precedence over the `columns` property (if the two are used together).
- `hoverPseudoclassName` - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```
p:before {
  content: oxy_htmlContent(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
  border: 1px solid red;
}
```

You can customize the style of the content using CSS that is either referenced by the file identified by the `href` property or is defined inline. If you change the HTML content or CSS and you want your changes to be reflected in the XML that renders the form control, then you need to refresh the XML file. If the HTML does not have an associated style, then a default text and background color will be applied.

Example: *oxy_htmlContent* Form Control

In the following example, the form control collects the content from the `p_description` `div` element found in the `descriptions.html` file. The box is 400 pixels wide and is displayed before a paragraph identified by the `intro_id` attribute value.

```
p#intro_id:before {
    content:
        oxy_htmlContent(
            href, "descriptions.html",
            id, "p_description",
            width, 400px);
}
```

An alternative example, using the `content` property:

```
p#intro_id:before {
    content:
```

```
        oxy_htmlContent(
            content, "<div style='font-weight:bold;'>My content</div>",
            width, 400px);
}
```

Tip: To insert a sample of the `oxy_htmlContent` form control in a CSS file (or LESS file), invoke the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_htmlContent` code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: ***[OXYGEN_INSTALL_DIR]*/samples/form-controls**.

Pop-up Form Control
The `oxy_popup` built-in form control is used to offer a contextual menu that provides quick access to various actions. A pop-up form control can display single or multiple selections.

The `oxy_popup` form control supports the following properties:

- `edit` - Lets you edit the value of an attribute, the text content of an element, or Processing Instructions (PI). This property can have the following values:
  - **@attribute_name** - The name of the attribute whose value is being edited. If the attribute is in a namespace, the value of the property must be a *QName* and the CSS must have a namespace declaration for the prefix.
  - **#text** - Specifies that the presented/edited value is the simple text value of an element.

    Note: You can set the value of the `visibility` property to `-oxy-collapse-text` to render the text only in the form control that the `oxy_editor` function specifies.
- `rows` - This property specifies the number of rows that the form control presents.

  Note: If the value of the **rows** property is not specified, the default value of *12* is used.
- `color` - Specifies the foreground color of the form control. If the value of the `color` property is `inherit`, the form control has the same color as the element in which it is inserted.

  Note: This property is used for rendering in the **Author** mode.
- `visible` - Specifies whether or not the form control is visible. The possible values of this property are `true` (default value) and `false`.
- `tooltips` - Associates tooltips to each value in the `values` property. The value of this property is a list of tooltip messages separated by commas. If you want the tooltip to display a comma, use the `${comma}` variable.

  Example:

  ```
  link:before{
   content: oxy_popup(
           edit, '@href',
           values, "Spring, Summer, Autumn, Winter",
           tooltips, "Iris${comma}Snowdrop, Gardenia${comma}Liliac,
               Chrysanthemum${comma}Salvia, Gerbera",
          selectionMode, single);
  }
  ```
- `values` - Specifies the values that populate the list of proposals. If these values are not specified in the CSS, they are collected from the associated XML Schema.

  Note: Typically, when you use a comma in the values of a form control, the content that follows a comma is considered a new value. If you want to include a comma in the values, precede the comma with two backslashes. For example, `(values, '1\\, 2\\, 3, 4, edit, false)` will display a form control that has `1, 2, 3` for the first value and 4 for the second value.
- `resultSeparator` - If multiple check-boxes are used, the separator is used to compose the final result. If not specified, the *space* character is used.

  Note: The value of the `resultSeparator` property cannot exceed one character.
- `selectionMode` - Specifies whether the form control allows the selection of a single value or multiple values. The predefined values of this property are `single` (default value) and `multiple`.
- `labels` - Specifies the label associated with each entry used for presentation. If this property is not specified, the **values** property is used instead.
- `columns` - Controls the width of the form control. The unit size is the width of the **w** character. This property is used for the visual representation of the form control.

- `width` - Specifies the width of the content area using relative (`em`, `ex`), absolute (`in`, `cm`, `mm`, `pt`, `pc`, `px`), and percentage (followed by the `%` character) length units. The `width` property takes precedence over the `columns` property (if the two are used together).
- `rendererSort` - Allows you to sort the values rendered on the form control label. The possible values of this property are `ascending` and `descending`.
- `editorSort` - Allows you to sort the values rendered on the form control. The possible values of this property are `ascending` and `descending`.
- `rendererSeparator` - Defines a separator used when multiple values are rendered. If not specified, the value of the `resultSeparator` property is used.
- `fontInherit` - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be `true` or `false` (default value). To make the form control inherit its font from its parent element, set the `fontInherit` property to `true`.
- `hoverPseudoclassName` - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```
p:before {
  content: oxy_popup(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
  border: 1px solid red;
}
```

Example: *oxy_popup* Form Control

```
popupWithMultipleSelection:before {
    content: " This editor edits an attribute value.
               The possible values are specified
    inside the CSS: "
        oxy_popup(
            edit, "@attribute",
            values, "value1, value2, value3, value4, value5",
            labels, "Value no1, Value no2, Value no3, Value no4, Value no5",
            resultSeparator, "|",
            columns, 10,
            selectionMode, "multiple",
            color, "blue",
            fontInherit, true);
    font-size:30px;
}
```

Tip: To insert a sample of the `oxy_popup` form control in a CSS file (or LESS file), invoke the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_popup` code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: ***[OXYGEN_INSTALL_DIR]*/samples/form-controls**.

Text Area Form Control
The `oxy_textArea` built-in form control is used for entering multiple lines of text in a graphical user interface box. A text area may include optional syntax highlight capabilities to present the form control.

The `oxy_textArea` form control supports the following properties:

- `edit` - Lets you edit the value of an attribute, the text content of an element, or Processing Instructions (PI). This property can have the following values:
  - **@attribute_name** - The name of the attribute whose value is being edited. If the attribute is in a namespace, the value of the property must be a *QName* and the CSS must have a namespace declaration for the prefix.
  - **#text** - Specifies that the presented/edited value is the simple text value of an element.

    Note: You can set the value of the `visibility` property to `-oxy-collapse-text` to render the text only in the form control that the `oxy_editor` function specifies.
  - **#content** - This parameter is useful when an element has mixed or element-only content and you want to edit its content inside a text area form control.

    For example, if you have the following XML content:

```
<codeblock outputclass="language-xml">START_TEXT<ph>phase</ph>
          <apiname><text>API</text></apiname></codeblock>
```

and your CSS includes the following snippet:

```
codeblock:before{
content:
    oxy_textArea(
      edit, '#content',
      contentType, 'text/xml');
}
```

then the text area form control will edit the following fragment:

```
START_TEXT<ph>phase</ph><apiname><text>API</text></apiname>
```

Note: When the value of the `edit` property is `#content`, the text area form control will also offer content completion proposals.

- **#content** - This parameter is useful when an element has mixed or element-only content and you want to edit its content inside a text area form control.

  For example, if you have the following XML content:

  ```
  <codeblock outputclass="language-xml">START_TEXT<ph>phase</ph>
             <apiname><text>API</text></apiname></codeblock>
  ```

  and your CSS includes the following snippet:

  ```
  codeblock:before{
  content:
      oxy_textArea(
        edit, '#content',
        contentType, 'text/xml');
  }
  ```

  then the text area form control will edit the following fragment:

  ```
  START_TEXT<ph>phase</ph><apiname><text>API</text></apiname>
  ```

  Note: When the value of the `edit` property is `#content`, the text area form control will also offer content completion proposals.
- `columns` - Controls the width of the form control. The unit size is the width of the **w** character.
- `width` - Specifies the width of the content area using relative (`em`, `ex`), absolute (`in`, `cm`, `mm`, `pt`, `pc`, `px`), and percentage (followed by the % character) length units. The `width` property takes precedence over the `columns` property (if the two are used together).
- `fontInherit` - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be `true` or `false` (default value). To make the form control inherit its font from its parent element, set the `fontInherit` property to `true`.
- `visible` - Specifies whether or not the form control is visible. The possible values of this property are `true` (default value) and `false`.
- `rows` - This property specifies the number of rows that the form control presents. If the form control has more lines, you can scroll and see them all.
- `contentType` - Specifies the type of content for which the form control offers syntax highlighting. The following values are supported: `text/css; text/shell; text/cc; text/xquery; text/xml; text/python; text/xsd; text/c; text/xpath; text/javascript; text/xsl; text/wsdl; text/html; text/xproc; text/properties; text/sql; text/rng; text/sch; text/json; text/perl; text/php; text/java; text/batch; text/rnc; text/dtd; text/nvdl; text/plain.`
- `indentOnTab` - Specifies the behavior of the **Tab** key. If the value of this property is set to `true` (default value), the **Tab** key inserts characters. If it is set to `false`, **Tab** is used for navigation, jumping to the next editable position in the document.
- The `white-space` CSS property influences the value that you edit, as well as the from control size:
  - `pre` - The whitespaces and new lines of the value are preserved and edited. If the `rows` and `columns` properties are not specifies, the form control calculates its size on its own so that all the text is visible.
  - `pre-wrap` - The long lines are wrapped to avoid horizontal scrolling.

Note: The `rows` and `columns` properties must be specified. If these are not specified, the form control considers the value to be `pre`.

- `normal` - The white spaces and new lines are normalized.

- `hoverPseudoclassName` - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```
p:before {
  content: oxy_textArea(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
  border: 1px solid red;
}
```

Example: *oxy_textArea* Form Control

The following example presents a text area with CSS syntax highlighting that calculates its own dimension, and a second one with XML syntax highlighting with defined dimension.

```
textArea {
    visibility: -oxy-collapse-text;
    white-space: pre;
}

textArea[language="CSS"]:before {
    content: oxy_textArea(
      edit, '#text',
      contentType, 'text/css');
}

textArea[language="XML"]:before {
    content: oxy_textArea(
      edit, '#text',
      contentType, 'text/xml',
      rows, 10,
      columns, 30);
}
```

Tip: To insert a sample of the `oxy_textArea` form control in a CSS file (or LESS file), invoke the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_textArea` code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: ***[OXYGEN_INSTALL_DIR]*/samples/form-controls**.

Text Field Form Control
The `oxy_textfield` built-in form control is used for entering a single line of text in a graphical user interface box. A text field may include optional content completion capabilities, used to present and edit the value of an attribute or an element.

The `oxy_textfield` form control supports the following properties:

- `edit` - Lets you edit the value of an attribute, the text content of an element, or Processing Instructions (PI). This property can have the following values:
  - **@attribute_name** - The name of the attribute whose value is being edited. If the attribute is in a namespace, the value of the property must be a *QName* and the CSS must have a namespace declaration for the prefix.
  - **#text** - Specifies that the presented/edited value is the simple text value of an element.

    Note: You can set the value of the `visibility` property to `-oxy-collapse-text` to render the text only in the form control that the `oxy_editor` function specifies.
- `columns` - Controls the width of the form control. The unit size is the width of the **w** character.
- `width` - Specifies the width of the content area using relative (`em`, `ex`), absolute (`in`, `cm`, `mm`, `pt`, `pc`, `px`), and percentage (followed by the % character) length units. The `width` property takes precedence over the `columns` property (if the two are used together).
- `fontInherit` - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be `true` or `false` (default value). To make the form control inherit its font from its parent element, set the `fontInherit` property to `true`.
- `visible` - Specifies whether or not the form control is visible. The possible values of this property are `true` (default value) and `false`.

- `values` - Specifies the values that populate the list of proposals. If these values are not specified in the CSS, they are collected from the associated XML Schema.
- `tooltips` - Associates tooltips to each value in the `values` property. The value of this property is a list of tooltip messages separated by commas. If you want the tooltip to display a comma, use the `${comma}` variable.
- `tooltip` - Specifies a tooltip to be displayed when you hover over the form control.
- `color` - Specifies the foreground color of the form control. If the value of the `color` property is `inherit`, the form control has the same color as the element in which it is inserted.
- `hasMultipleValues` - Specifies if the text field allows multiple values separated by spaces or just a single value.

  Note: If the value is `false`, the *Content Completion Assistant* considers the entire text as the prefix for its proposals. If the value is `true` (the default value), the space is the delimiter for the values and thus it is not included in the prefix (the prefix will be whatever comes after the space).

  For example, suppose the possible values for your text field are: `value a`, `value b`, and `other values`. If the `hasMultipleValues` property is set to `true` and the user enters `"value "` (notice the space character after 'value') in the text field, the *Content Completion Assistant* will suggest all three values because the prefix is whatever comes after the space, and in this case the user did not enter anything after the space. If the `hasMultipleValues` property was set to `false` for our example, the *Content Completion Assistant* would only suggest `value a` and `value b` because the space is considered part of the prefix.
- `hoverPseudoclassName` - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```
p:before {
  content: oxy_textfield(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
  border: 1px solid red;
}
```

Example: *oxy_textfield* Form Control

```
element {
    content: "Label: "
        oxy_textfield(
            edit, "@my_attr",
            values, "value1, value2",
            color, "red",
            columns, 40);
}
```

Tip: To insert a sample of the `oxy_textfield` form control in a CSS file (or LESS file), invoke the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_textfield` code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: ***[OXYGEN_INSTALL_DIR]*/samples/form-controls**.

URL Chooser Form Control
The `oxy_urlChooser` built-in form control is used for a dialog box that allows you to select the location of local or remote resources. The inserted reference is made relative to the URL of the currently opened editor.

The `oxy_urlChooser` editor supports the following properties:

- `edit` - Lets you edit the value of an attribute, the text content of an element, or Processing Instructions (PI). This property can have the following values:
  - **@attribute_name** - The name of the attribute whose value is being edited. If the attribute is in a namespace, the value of the property must be a *QName* and the CSS must have a namespace declaration for the prefix.
  - **#text** - Specifies that the presented/edited value is the simple text value of an element.

    Note: You can set the value of the `visibility` property to `-oxy-collapse-text` to render the text only in the form control that the `oxy_editor` function specifies.
- `columns` - Controls the width of the form control. The unit size is the width of the **w** character.

- `width` - Specifies the width of the content area using relative (`em`, `ex`), absolute (`in`, `cm`, `mm`, `pt`, `pc`, `px`), and percentage (followed by the % character) length units. The `width` property takes precedence over the `columns` property (if the two are used together).
- `color` - Specifies the foreground color of the form control. If the value of the `color` property is `inherit`, the form control has the same color as the element in which it is inserted.
- `visible` - Specifies whether or not the form control is visible. The possible values of this property are `true` (default value) and `false`.
- `fontInherit` - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be `true` or `false` (default value). To make the form control inherit its font from its parent element, set the `fontInherit` property to `true`.
- `fileFilter` - string value that holds comma-separated file extensions. The URL chooser uses these extensions to filter the displayed files. A value such as `"jpg,png,gif"` is mapped to a single filter that will display all `jpg`, `png`, and `gif` files.
- `hoverPseudoclassName` - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```
p:before {
  content: oxy_urlChooser(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
  border: 1px solid red;
}
```

Example: *oxy_urlChooser* Form Control

```
urlChooser[file]:before {
    content: "A URL chooser editor that allows browsing for a URL.
          The selected URL is made relative to the currently edited file:"
        oxy_urlChooser(
        edit, "@file",
        columns 25);
}
```

Tip: To insert a sample of the `oxy_urlChooser` form control in a CSS file (or LESS file), invoke the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_urlChooser` code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: ***[OXYGEN_INSTALL_DIR]*/samples/form-controls**.

Video Player Form Control
The `oxy_video` built-in form control is used for providing a mechanism to play videos.

The `oxy_video` form control supports the following properties:

- `href` - The absolute or relative location of a resource. This property is mandatory. Relative values are resolved relative to the CSS. If you have media resources relative to the XML document, you can specify their paths like this:

```
oxy_video(href, oxy_url(oxy_base-uri(), 'ex.mp4')), width, 400px, height, 300px)
```

- `width` - Specifies the width of the content area using relative (`em`, `ex`), absolute (`in`, `cm`, `mm`, `pt`, `pc`, `px`), and percentage (followed by the % character) length units.
- `height` - Specifies the height of the form control area using relative (`em`, `ex`), absolute (`in`, `cm`, `mm`, `pt`, `pc`, `px`), and percentage (followed by the % character) length units.

Example: *oxy_video* Form Control

```
object {
    content:
        oxy_video(
            href, 'resources/video.mp4',
            width, 400px,
            height, 300px),
}
```

Tip: To insert a sample of the `oxy_video` form control in a CSS file (or LESS file), invoke the *Content Completion Assistant* by pressing **Ctrl + Space (Command + Space on OS X)** and select the `oxy_video` code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: ***[OXYGEN_INSTALL_DIR]*/samples/form-controls**.

Implementing Custom Form Controls
If the built-in form controls are not sufficient for your needs, you can implement custom form controls in Java.

Custom Form Controls Implementation

You can specify custom form controls using the following properties:

- **rendererClassName** - The name of the class that draws the edited value. It must be an implementation of *`ro.sync.ecss.extensions.api.editor.InplaceRenderer`*. The renderer has to be a **SWING** implementation and can be used both in the standalone and Eclipse distributions.
- **swingEditorClassName** - You can use this property for the standalone (**Swing**-based) distribution to specify the name of the class used for editing. It is a **Swing** implementation of *`ro.sync.ecss.extensions.api.editor.InplaceEditor`*.
- **swtEditorClassName** - You can use this property for the Eclipse plugin distribution to specify the name of the class used for editing. It is a **SWT** implementation of the *`ro.sync.ecss.extensions.api.editor.InplaceEditor`*.

  Note: If the custom form control is intended to work in the Oxygen XML Editor standalone distribution, the declaration of **swtEditorClassName** is not required. The **renderer** (the class that draws the value) has different properties from the **editor** (the class that edits the value) because you can present a value in one way and edit it in another.
- **classpath** - You can use this property to specify the location of the classes used for a custom form control. The value of the **classpath** property is an enumeration of URLs separated by comma.
- **edit** - If your form control edits the value of an attribute or the text value of an element, you can use the `@attribute_name` and `#text` predefined values and Oxygen XML Editor will perform the commit logic by itself. You can use the `custom` value to perform the commit logic yourself.
- **saHeavyFormControlClassName** - This type of form control is effectively present at all times at its allocated bounds. This is useful if you need a form controls that renders dynamic or interactive SVG documents (for example, if you have an SVG document that displays tooltips when hovering over certain areas). It is also helpful if you want to use JavaFX, since JavaFX-based form controls are not compatible with the classic form control architecture.

  The value of this property is a class name that must implement the *`ro.sync.ecss.extensions.api.editor.InplaceHeavyEditor`* method. The *JAR* that contains this implementation can either be added in the ***Classpath** tab in the **Document Type Configuration** dialog box* for your particular *framework* or specified with the *classpath property*.

Example: Java Code

The following is a sample Java code for implementing a custom combo box form control that inserts an XML element in the content when the editing stops:

```
public class ComboBoxEditor  extends AbstractInplaceEditor {
  /**
   * @see ro.sync.ecss.extensions.api.editor.InplaceEditor#stopEditing()
   */
  @Override
   public void stopEditing() {
     Runnable customCommit =  new Runnable() {
      @Override
       public void run() {
        AuthorDocumentController documentController =
            context.getAuthorAccess().getDocumentController();
        documentController.insertXMLFragment( "<custom/>", offset);
      }
    };
    EditingEvent event =  new EditingEvent(customCommit,  true);
    fireEditingStopped(event);
  }
```

The custom form controls can use any of the predefined properties of the *built-in form controls*, as well as specified custom properties.

Example: CSS

The following is an example of how to specify a custom form control in the CSS:

```
myElement {
    content: oxy_editor(
        rendererClassName, "com.custom.editors.CustomRenderer",
        swingEditorClassName, "com.custom.editors.SwingCustomEditor",
        swtEditorClassName, "com.custom.editors.SwtCustomEditor",
        edit, "@my_attr",
        customProperty1, "customValue1",
        customProperty2, "customValue2"
    )
}
```

How to Implement Custom Form Controls

To implement a custom form control, follow these steps:

1. Download the Oxygen XML Editor SDK at *https://www.oxygenxml.com/oxygen_sdk_maven.html*.
2. Implement the custom form control by extending `ro.sync.ecss.extensions.api.editor.InplaceEditorRendererAdapter`. You could also use `ro.sync.ecss.extensions.api.editor.AbstractInplaceEditor`, which offers some default implementations and listeners management.
3. Pack the previous implementation in a Java *JAR* library.
4. Copy the *JAR* library to the `[OXYGEN_INSTALL_DIR]/frameworks/[FRAMEWORK_DIR]` directory.
5. In Oxygen XML Editor, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, edit the appropriate *framework*, and add the *JAR* library in the **Classpath** tab.
6. Specify the custom form control in your CSS, as described above.

Tip: To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: ***[OXYGEN_INSTALL_DIR]*/samples/form-controls**.

Editing Processing Instructions Using Form Controls
Oxygen XML Editor allows you to edit *processing instructions*, *comments*, and *CDATA* by using the built-in editors.

Oxygen XML Editor allows you to edit *processing instructions*, *comments*, and *CDATA* by using the built-in editors.

Note: You can edit both the content and the attribute value from a *processing instruction*.

Example: Editing an Attribute from a Processing Instruction

PI content:

```
<?pi_target attr="val"?>
```

CSS:

```
oxy|processing-instruction:before {
    display:inline;
    content:
        "EDIT attribute: " oxy_textfield(edit, '@attr', columns, 15);
    visibility:visible;
}
oxy|processing-instruction{
    visibility:-oxy-collapse-text;
}
```

Custom CSS Pseudo-classes

You can set your custom CSS pseudo-classes on the nodes from the `AuthorDocument` model. These are similar to the normal XML attributes, with the important difference that they are not serialized, and by changing them the document does not create undo and redo edits - the document is considered unmodified. You can use custom pseudo-classes for changing the style of an element (and its children) without altering the document.

In Oxygen XML Editor they are used to hide/show the `colspec` elements from CALS tables. To take a look at the implementation, see:

1. `[OXYGEN_INSTALL_DIR]/frameworks/docbook/css/cals_table.css` (Search for `-oxy-visible-colspecs`)

2. The definition of action `table.toggle.colspec` from the DocBook *framework* makes use of the pre-defined *`TogglePseudoClassOperation`* **Author** mode operation.

Here are some examples:

Example: Controlling the visibility of a section using a pseudo-class

You can use a non standard (custom) pseudo-class to impose a style change on a specific element. For instance, you can have CSS styles matching the custom pseudo-class `access-control-user`, like the one below:

```
section {
  display:none;
}

section:access-control-user {
  display:block;
}
```

By setting the pseudo-class `access-control-user`, the element section will become visible by matching the second CSS selector.

Example: Coloring the elements at the current cursor location

You could create an *`AuthorCaretListener`* that sets the `caret-visited` pseudo-class to the element at the cursor location. The effect will be that all the elements traversed by the cursor become red.

```
*:caret-visited {
  color:red;
}
```

The API that you can use from the `CaretListener`:

`ro.sync.ecss.extensions.api.AuthorDocumentController#setPseudoClass(java.lang.String, ro.sync.ecss.extensions.api.node.AuthorElement)`

`ro.sync.ecss.extensions.api.AuthorDocumentController#removePseudoClass(java.lang.String, ro.sync.ecss.extensions.api.node.AuthorElement)`

Predefined Pseudo-Class Author Mode Operations

Pre-defined ***Author*** *mode operations* can be used directly in your *framework* to work with custom pseudo-classes:

1. *`TogglePseudoClassOperation`*
2. *`SetPseudoClassOperation`*
3. *`RemovePseudoClassOperation`*

Debugging CSS Stylesheets

To assist you with debugging and customizing CSS stylesheets the **Author** mode includes a ***CSS Inspector*** *view* to examine the CSS rules that match the currently selected element.

This tool is similar to the Inspect Element development tool that is found in most browsers. The **CSS Inspector** view allows you to see how the CSS rules are applied and the properties defined. Each rule that is displayed in this view includes a link to the line in the CSS file that defines the styles for the element that matches the rule. You can use the link to open the appropriate CSS file and edit the style rules. Once you have found the rule you want to edit, you can click the link in the top-right corner of that rule to open the CSS file in the editor.

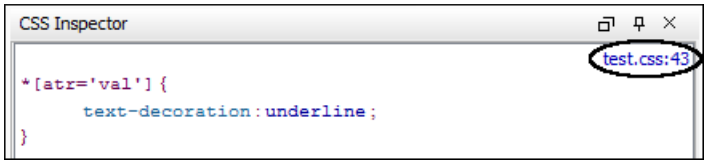


Figure 589: CSS Inspector View

There are two ways to open the CSS Inspector view:

1. Select **CSS Inspector** from the **Window** > **Show View** menu.
2. Select the **Inspect Styles** action from the contextual menu in **Author** mode.

Related Information:
CSS Inspector View on page 232

Creating and Running Automated Tests

If you have developed complex custom *plugin* or *framework* (document types), the best way to test your implementation and ensure that further changes will not interfere with the current behavior is to make automated tests for your customization.

An Oxygen XML Editor standalone installation includes a main `oxygen.jar` library located in the *`[OXYGEN_INSTALL_DIR]`*. That *JAR* library contains a base class for testing developer customizations named: `ro.sync.exml.workspace.api.PluginWorkspaceTCBase`.

To develop `JUnit` tests for your customizations using the **Eclipse** workbench, follow these steps:

1. Create a new Eclipse Java project and copy to it the entire contents of the *`[OXYGEN_INSTALL_DIR]`*.
2. Add all *JAR* libraries present in the *`[OXYGEN_INSTALL_DIR]`*`/lib` directory to the **Java Build Path**->**Libraries** tab. Make sure that the main *JAR* library `oxygen.jar` or `oxygenAuthor.jar` is the first one in the Java classpath by moving it up in the **Order and Export** tab.
3. Click **Add Library** and add the *`JUnit`* and *`JFCUnit`* libraries.
4. Create a new Java class that extends `ro.sync.exml.workspace.api.PluginWorkspaceTCBase`.
5. Pass the following parameters on to the constructor of the super class:
   - `File installationFolder` - The file path to the main application installation directory. If not specified, it defaults to the folder where the test is started.
   - `File frameworksFolder` - The file path to the `frameworks` directory. It can point to a custom *framework* directory where it resides.
   - `File pluginsFolder` - The file path to the `plugins` directory. It can point to a custom *plugin* directory where it resides.
   - `File optionsFolder` - The folder that contains the application options. If not specified, the application will auto-detect the location based on the started product ID.
   - `String licenseKey` - The license key used to license the test class.
   - `int productID` - The ID of the product and should be one of the following: `PluginWorkspaceTCBase.XML_AUTHOR_PRODUCT`, `PluginWorkspaceTCBase.XML_EDITOR_PRODUCT`, or `PluginWorkspaceTCBase.XML_DEVELOPER_PRODUCT`.
6. Create test methods that use the API in the base class to open XML files and perform various actions on them. Your test class could look something like this:

```
public class MyTestClass extends PluginWorkspaceTCBase {

/**
 * Constructor.
 */
public MyTestClass() throws Exception {
 super(null, new File("frameworks"), new File("plugins"), null,
"------START-LICENSE-KEY------\n" +
 "\n" +
 "Registration_Name=Developer\n" +
 "\n" +
 "Company=\n" +
 "\n" +
 "Category=Enterprise\n" +
 "\n" +
 "Component=XML-Editor, XSLT-Debugger, Saxon-SA\n" +
 "\n" +
 "Version=14\n" +
 "\n" +
 "Number_of_Licenses=1\n" +
 "\n" +
 "Date=09-04-2012\n" +
 "\n" +
 "Trial=31\n" +
 "\n" +
 "SGN=MCwCFGNoEGJSeiC3XCYIyalvjzHhGhhqAhRNRDpEu8RIWb8icCJO7HqfVP4++A\\=\\=\n" +
 "\n" +
"-------END-LICENSE-KEY-------",
 PluginWorkspaceTCBase.XML_AUTHOR_PRODUCT);
 }
```

```
 /**
   * <p><b>Description:</b> TC for opening a file and using a bold operation</p>
   * <p><b>Bug ID:</b> EXM-20417</p>
   *
   * @author radu_coravu
   *
   * @throws Exception
   */
public void testOpenFileAndBoldEXM_20417() throws Exception {
      WSEditor ed = open(new File
("D:/projects/eXml/test/authorExtensions/dita/sampleSmall.xml").toURL());
     //Move caret
     moveCaretRelativeTo("Context", 1, false);

     //Insert <b>
     invokeAuthorExtensionActionForID("bold");
     assertEquals("<?xml version=\"1.0\" encoding=\"utf-8\"?>\n" +
       "<!DOCTYPE task PUBLIC \"-//OASIS//DTD DITA Task//EN\"
 \"http://docs.oasis-open.org/dita/v1.1/OS/dtd/task.dtd\">\n" +
       "<task id=\"taskId\">\n" +
       "    <title>Task <b>title</b></title>\n" +
       "    <prolog/>\n" +
       "    <taskbody>\n" +
       "        <context>\n" +
       "            <p>Context for the current task</p>\n" +
       "        </context>\n" +
       "        <steps>\n" +
       "            <step>\n" +
       "                <cmd>Task step.</cmd>\n" +
       "            </step>\n" +
       "        </steps>\n" +
       "    </taskbody>\n" +
       "</task>\n" +
       "", getCurrentEditorXMLContent());
   }
}
```

API Frequently Asked Questions (API FAQ)

This section contains answers to common questions regarding the Oxygen XML Editor customizations using the *Oxygen SDK*, *Author Component*, or *Plugins*.

For additional questions, *contact us*. The preferred approach is via email because API questions must be analyzed thoroughly. We also provide code snippets, if they are required.

To stay up-to-date with the latest API changes, discuss issues and ask for solutions from other developers working with the *Oxygen SDK*, register on the *Oxygen-SDK mailing list*.

Add Custom Actions to the Contextual Menu?

Question

How do I add my own custom actions to the contextual menu using an API?

Answer

The API methods `WSAuthorEditorPageBase.addPopUpMenuCustomizer` and `WSTextEditorPage.addPopUpMenuCustomizer` allow you to customize the contextual menu shown either in the **Author** or **Text** modes. The API is available both in the standalone application and in the Eclipse plugin.

Here is an elegant way to add actions to the **Author** page from your Eclipse plugin extension:

1. Create a pop-up menu customizer implementation:

```
import org.eclipse.jface.action.ContributionManager;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.menus.IMenuService;
import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.structure.AuthorPopupMenuCustomizer;
/**
* This class is used to create the possibility to attach certain
* menuContributions to the {@link ContributionManager}, which is used for the
* popup menu in the Author Page of the Oxygen Editor.<br />
* You just need to use the org.eclipse.ui.menus extension and add a
* menuContribution with the locationURI: <b>menu:oxygen.authorpage</b>
*/
public class OxygenAuthorPagePopupMenuCustomizer implements
        AuthorPopupMenuCustomizer {
```

```
    @Override
    public void customizePopUpMenu(Object menuManagerObj,
            AuthorAccess authoraccess) {
        if (menuManagerObj instanceof ContributionManager) {
 ContributionManager contributionManager = (ContributionManager) menuManagerObj;
            IMenuService menuService = (IMenuService) PlatformUI.getWorkbench()
                    .getActiveWorkbenchWindow().getService(IMenuService.class);

            menuService.populateContributionManager(contributionManager,
                    "menu:oxygen.authorpage");
            contributionManager.update(true);
        }
    }
}
```

2. Add a workbench listener and add the pop-up customizer when an editor is opened in the **Author** page:

```
Workbench.getInstance().getActiveWorkbenchWindow().getPartService()
.addPartListener(
    new IPartListener() {
      @Override
      public void partOpened(IWorkbenchPart part) {
        if(part instanceof ro.sync.exml.workspace.api.editor.WSEditor) {
          WSEditorPage currentPage = ((WSEditor)part).getCurrentPage();
          if(currentPage instanceof WSAuthorEditorPage) {
            ((WSAuthorEditorPage)currentPage).addPopUpMenuCustomizer
(new OxygenAuthorPagePopupMenuCustomizer());
          }
        }
      }
      ........
    });
```

3. Implement the extension point in your `plugin.xml` file:

```
<extension
       point="org.eclipse.ui.menus">
    <menuContribution
          allPopups="false"
          locationURI="menu:oxygen.authorpage">
       <command
             commandId="eu.doccenter.kgu.client.tagging.removeTaggingFromOxygen"
             style="push">
       </command>
    </menuContribution>
</extension>
```

Add Custom Callouts

Question

I want to highlight validation errors, instead of underlining them (for example, changing the text background color to light red or yellow). Also, I want to let Oxygen XML Editor write a note about the error type into the **Author** mode directly at the error position (for example, " [value "text" not allowed for attribute "type"] "). Is this possible using the API?

Answer

The Plugins API allows you to set a `ValidationProblemsFilter` that gets notified when automatic validation errors are available. Then you can map each of the problems to an offset range in the **Author** mode using the API `WSTextBasedEditorPage.getStartEndOffsets(DocumentPositionedInfo)`. For each of those offsets, you can add either persistent or non-persistent highlights. If you add persistent highlights, you can also customize callouts to appear for each of them. The downside is that they need to be removed before the document gets saved. The result would look something like this:

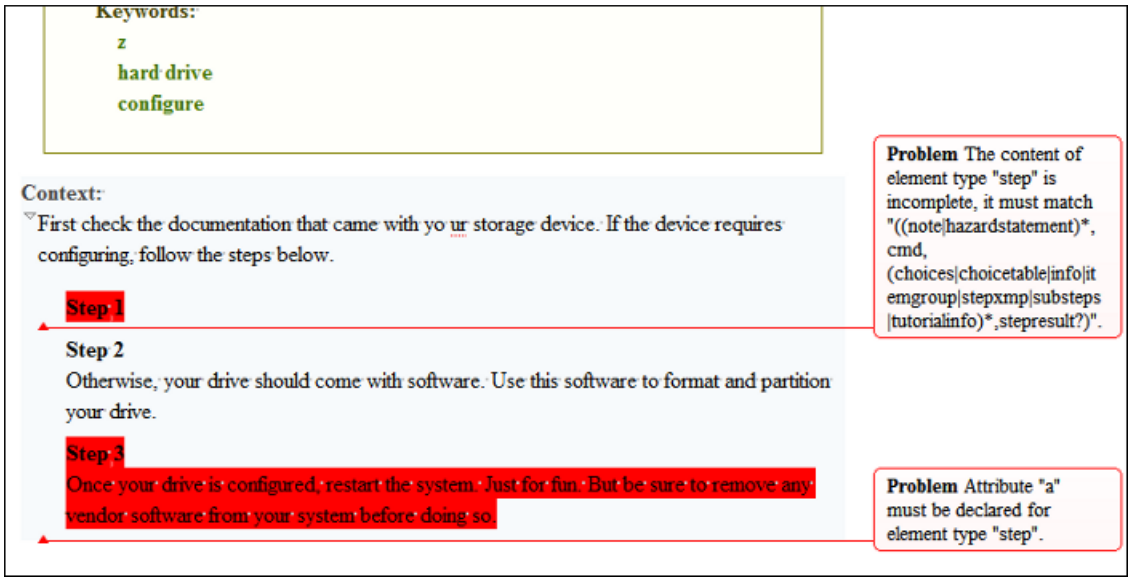


Figure 590: Custom Callouts with Persistent Highlights

Here is a working example:

```
/**
* Plugin extension - workspace access extension.
*/
public class CustomWorkspaceAccessPluginExtension
                     implements WorkspaceAccessPluginExtension {

/**
* @see ro.sync.exml.plugin.workspace.WorkspaceAccessPluginExtension
   #applicationStarted(
ro.sync.exml.workspace.api.standalone.StandalonePluginWorkspace)
*/
    public void applicationStarted
(final StandalonePluginWorkspace pluginWorkspaceAccess) {
        pluginWorkspaceAccess.addEditorChangeListener
(new WSEditorChangeListener() {
          /**
           * @see WSEditorChangeListener#editorOpened(java.net.URL)
           */
          @Override
          public void editorOpened(URL editorLocation) {
            final WSEditor currentEditor = pluginWorkspaceAccess.getEditorAccess
(editorLocation, StandalonePluginWorkspace.MAIN_EDITING_AREA);
            WSEditorPage currentPage = currentEditor.getCurrentPage();
            if(currentPage instanceof WSAuthorEditorPage) {
              final WSAuthorEditorPage currentAuthorPage =
(WSAuthorEditorPage)currentPage;
              currentAuthorPage.getPersistentHighlighter().setHighlightRenderer
(new PersistentHighlightRenderer() {
                @Override
                public String getTooltip(AuthorPersistentHighlight highlight) {
                  return highlight.getClonedProperties().get("message");
                }
                @Override
                public HighlightPainter getHighlightPainter
(AuthorPersistentHighlight highlight) {
                  //Depending on severity could have different color.
                  ColorHighlightPainter painter = new ColorHighlightPainter
(Color.COLOR_RED, -1, -1);
                  painter.setBgColor(Color.COLOR_RED);
                  return painter;
                }
              });
         currentAuthorPage.getReviewController()
         .getAuthorCalloutsController().setCalloutsRenderingInformationProvider(
               new CalloutsRenderingInformationProvider() {
               @Override
     public boolean shouldRenderAsCallout(AuthorPersistentHighlight highlight) {
                  //All custom highlights are ours
                  return true;
                }
                @Override
     public AuthorCalloutRenderingInformation getCalloutRenderingInformation(
                    final AuthorPersistentHighlight highlight) {
                  return new AuthorCalloutRenderingInformation() {
                    @Override
                    public long getTimestamp() {
                      //Not interesting
                      return -1;
                    }
```

```
                    @Override
                    public String getContentFromTarget(int limit) {
                      return "";
                    }
                    @Override
                    public String getComment(int limit) {
                      return highlight.getClonedProperties().get("message");
                    }
                    @Override
                    public Color getColor() {
                      return Color.COLOR_RED;
                    }
                    @Override
                    public String getCalloutType() {
                      return "Problem";
                    }
                    @Override
                    public String getAuthor() {
                      return "";
                    }
                    @Override
                    public Map<String, String> getAdditionalData() {
                      return null;
                    }
                  };
                }
              });
 currentEditor.addValidationProblemsFilter(new ValidationProblemsFilter() {
    List<int[]> lastStartEndOffsets = new ArrayList<int[]>();
    /**
    * @see ro.sync.exml.workspace.api.editor.validation.ValidationProblemsFilter
    #filterValidationProblems
(ro.sync.exml.workspace.api.editor.validation.ValidationProblems)
      */
   @Override
   public void filterValidationProblems(ValidationProblems validationProblems) {
     List<int[]> startEndOffsets = new ArrayList<int[]>();
     List<DocumentPositionedInfo> problemsList =
validationProblems.getProblemsList();
     if(problemsList != null) {
       for (int i = 0; i < problemsList.size(); i++) {
         try {
 startEndOffsets.add(currentAuthorPage.getStartEndOffsets(problemsList.get(i)));
         } catch (BadLocationException e) {
           e.printStackTrace();
         }
       }
     }
       if(lastStartEndOffsets.size() != startEndOffsets.size()) {
         //Continue
       } else {
         boolean equal = true;
         for (int i = 0; i < startEndOffsets.size(); i++) {
           int[] o1 = startEndOffsets.get(i);
           int[] o2 = lastStartEndOffsets.get(i);
           if(o1 == null && o2 == null) {
             //Continue
           } else  if(o1 != null && o2 != null
               && o1[0] == o2[0] && o1[1] == o2[1]){
             //Continue
           } else {
             equal = false;
             break;
           }
         }
         if(equal) {
           //Same list of problems already displayed.
           return;
         }
       }
       //Keep last used offsets.
       lastStartEndOffsets = startEndOffsets;
     try {
       if(! SwingUtilities.isEventDispatchThread()) {
         SwingUtilities.invokeAndWait(new Runnable() {
           @Override
         public void run() {
             //First remove all custom highlights.
     currentAuthorPage.getPersistentHighlighter().removeAllHighlights();
           }
         });
     }
     } catch (InterruptedException e1) {
     e1.printStackTrace();
     } catch (InvocationTargetException e1) {
       e1.printStackTrace();
     }
     if(problemsList != null) {
       for (int i = 0; i < problemsList.size(); i++) {
         //A reported problem (could be warning, could be error).
         DocumentPositionedInfo dpi = problemsList.get(i);
         try {
           final int[] currentOffsets = startEndOffsets.get(i);
           if(currentOffsets != null) {
             //These are offsets in the Author content.
```

```
             final LinkedHashMap<String, String> highlightProps =
new LinkedHashMap<String, String>();
             highlightProps.put("message", dpi.getMessage());
             highlightProps.put("severity", dpi.getSeverityAsString());
             if(! SwingUtilities.isEventDispatchThread()) {
               SwingUtilities.invokeAndWait(new Runnable() {
                 @Override
                 public void run() {
                  currentAuthorPage.getPersistentHighlighter().addHighlight(
                   currentOffsets[0], currentOffsets[1] - 1, highlightProps);
            }
               });
             }
          }
         } catch (InterruptedException e) {
           e.printStackTrace();
         } catch (InvocationTargetException e) {
           e.printStackTrace();
         }
      }
     }
   }
});
 currentEditor.addEditorListener(new WSEditorListener() {
  /**
  * @see WSEditorListener#editorAboutToBeSavedVeto(int)
  */
  @Override
  public boolean editorAboutToBeSavedVeto(int operationType) {
      try {
       if(! SwingUtilities.isEventDispatchThread()) {
        SwingUtilities.invokeAndWait(new Runnable() {
          @Override
          public void run() {
            //Remove all persistent highlights before saving
             currentAuthorPage.getPersistentHighlighter().removeAllHighlights();
           }
        });
       }
     } catch (InterruptedException e) {
     e.printStackTrace();
    } catch (InvocationTargetException e) {
      e.printStackTrace();
    }
    return true;
    }
   });
  }
 }
}, StandalonePluginWorkspace.MAIN_EDITING_AREA);
      }


/**
* @see WorkspaceAccessPluginExtension#applicationClosing()
       */
      public boolean applicationClosing() {
        return true;
      }
    }
```

Add Custom Highlights to Content

Question

How can we add custom highlights to the document content in **Author** mode?

Answer

There are two types of highlights you can add:

1. **Non-Persistent Highlights** - Such highlights are removed when the document is closed and then re-opened.

   You can use the following API method:

   `ro.sync.exml.workspace.api.editor.page.author.WSAuthorEditorPageBase.getHighlighter()`

   to obtain an *AuthorHighlighter* that allows you to add a highlight between certain offsets with a specified painter.

   For example, you can use this support to implement your own spell checker with a custom highlight for the unrecognized words.
2. **Persistent Highlights** - Such highlights are saved in the XML content as processing instructions.

You can use the following API method:

`ro.sync.exml.workspace.api.editor.page.author.WSAuthorEditorPageBase.getPersistentHighlig`

to obtain an *AuthorPersistentHighlighter* class that allows you to add a persistent highlight between certain offsets, set new properties for a specific highlight, and render it with a specified painter.

For example, you can use this support to implement your own way of adding review comments.

Related Information:

Adding Custom Persistent Highlights on page 1237

Auto-Generate an ID When a Document is Opened or Created

Question

Is it possible to configure how the application generates ids? For project compliance we need ids having a certain format for each created topic.

Answer

This could be done implementing a *plugin* for Oxygen XML Editor using the Plugins SDK:

https://www.oxygenxml.com/oxygen_sdk.html#Developer_Plugins

There is a type of *plugin* called "Workspace Access" that can be used to add a listener to be notified when an editor is opened.

The implemented *plugin* would intercept the opened editor and editor page change events (which occur when a new editor is created) and generate a new ID attribute value on the root element.

The Java code would look like this:

```
pluginWorkspaceAccess.addEditorChangeListener(new WSEditorChangeListener() {
/**
* @see WSEditorChangeListener#editorOpened(java.net.URL)
*/
@Override
public void editorOpened(URL editorLocation) {
    WSEditor ed = pluginWorkspaceAccess.getEditorAccess
(editorLocation, PluginWorkspace.MAIN_EDITING_AREA);
        generateID(ed);
}
/**
* @see WSEditorChangeListener#editorPageChanged(java.net.URL)
*/
@Override
public void editorPageChanged(URL editorLocation) {
    WSEditor ed = pluginWorkspaceAccess.getEditorAccess
(editorLocation, PluginWorkspace.MAIN_EDITING_AREA);
        generateID(ed);
}

private void generateID(WSEditor ed) {
 if(ed.getCurrentPage() instanceof WSAuthorEditorPage) {
  WSAuthorEditorPage authorEditPage = (WSAuthorEditorPage) ed.getCurrentPage();
  AuthorDocumentController ctrl = authorEditPage.getDocumentController();
  AuthorElement root = ctrl.getAuthorDocumentNode().getRootElement();
   if(root.getAttribute("id") == null ||
!root.getAttribute("id").getValue().startsWith("generated_")) {
    ctrl.setAttribute("id", new AttrValue("generated_" + Math.random()), root);
         }
       }
      }

    }, PluginWorkspace.MAIN_EDITING_AREA);
```

Change the Default Track Changes (Review) Author Name

Question

How can we change the default author name used for *Tracked Changes* in the *Author Component*?

Answer

The *Track Changes* (Review) author name is determined in the following order:

1. **API** - The review user name can be imposed through the following API:

```
ro.sync.ecss.extensions.api.AuthorReviewController.setReviewerAuthorName(String)
```

2. **Options** - If the author name was not imposed from the API, it is determined from the ***Author*** *option set in the* ***Review*** *preferences page*.
3. **System properties** - If the author name was not imposed from the API or from the application options then the following system property is used:

```
System.getProperty("user.name")
```

So, to impose the *Track Changes* author, use one of the following approaches:

1. Use the API to impose the reviewer author name. Here is the online Javadoc of this method: *https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/ecss/extensions/api/AuthorReviewController.html#setReviewerAuthorName(java.lang.String)*
2. Customize the default options and set a specific value for the ***Author*** *name option set in the* ***Review*** *preferences page*.
3. Set the value of `user.name` system property when the applet is initializing and before any document is loaded.

Change the DOCTYPE of an Opened XML Document

Question

How to change the DOCTYPE of a document opened in the **Author** mode?

Answer

The following API:

`ro.sync.ecss.extensions.api.AuthorDocumentController.getDoctype()`

allows you to get the DOCTYPE of the current XML file opened in the **Author** mode.

There is also an API method available that would allow you to set the DOCTYPE back to the XML:

`ro.sync.ecss.extensions.api.AuthorDocumentController.setDoctype(AuthorDocumentType)`

Here is an example of how this solution would work:

```
AuthorDocumentType
dt = new AuthorDocumentType("article", "testSystemID", "testPublicID",
            "<!DOCTYPE article PUBLIC \"testPublicID\" \"testSystemID\">");
docController.setDoctype(dt);
```

Basically, you could take the entire content from the existing DOCTYPE,

`ro.sync.ecss.extensions.api.AuthorDocumentType.getContent()`

modify it to your needs, and create another `AuthorDocumentType` object with the new content and with the same public, system IDs.

For example, you could use this API is you want to add unparsed entities in the XML DOCTYPE.

Control XML Serialization in the Oxygen XML Author Component

Question

How can I force the Oxygen XML Author Component to save the XML with zero indent size and not to break the line inside *block elements*?

Answer

Usually, in a standalone version of Oxygen XML Editor, the **Editor** > **Format** and **Editor** > **Format** > **XML** preferences pages allow you to control the way the XML is saved on the disk after you edit it in the **Author** mode.

In the editor application (Standalone or Eclipse-based), you can either bundle a *default set of options* or use the `PluginWorkspace.setGlobalObjectProperty(String, Object)` API:

```
//For not breaking the line
//Long line
pluginWorkspace.setObjectProperty("editor.line.width", new Integer(100000));
//Do not break before inline elements
pluginWorkspace.setObjectProperty("editor.format.indent.inline.elements", false)
;

//For forcing zero indent
//Force indent settings to be controlled by us
pluginWorkspace.setObjectProperty("editor.detect.indent.on.open", false);
//Zero indent size
pluginWorkspace.setObjectProperty("editor.indent.size.v9.2", 0);
```

In the Oxygen XML Author Component, you can either bundle a *fixed set of options*, or use our Java API to set properties that overwrite the default options:

```
    //For not breaking the line
    //Long line
    AuthorComponentFactory.getInstance().setObjectProperty
("editor.line.width", new Integer(100000));
    //Do not break before inline elements
    AuthorComponentFactory.getInstance().setObjectProperty
("editor.format.indent.inline.elements", false);

    //For forcing zero indent
    //Force indent settings to be controlled by us
    AuthorComponentFactory.getInstance().setObjectProperty
("editor.detect.indent.on.open", false);
    //Zero indent size
    AuthorComponentFactory.getInstance().setObjectProperty
("editor.indent.size.v9.2", 0);
```

Customize the Default Icons for Toolbars/Menus

Question

How can we change the default icons used for the application built-in actions?

Answer

If you look inside the main *JAR* library *[OXYGEN_INSTALL_DIR]*`\lib\oxygen.jar` or *[OXYGEN_INSTALL_DIR]*`\lib\author.jar`, it contains an `images` folder that contains all the images that we use for our buttons, menus, and toolbars.

To overwrite them with your own creations, follow these steps:

1. In the *[OXYGEN_INSTALL_DIR]*`\lib` directory create a folder called `endorsed`.
2. In the `endorsed` folder create another folder called `images`.
3. Add your own images in the `images` folder.

You can use this mechanism to overwrite any kind of resource located in the main *Oxygen JAR library*. The folder structure in the endorsed directory and in the main *Oxygen JAR* must be identical.

Customize the Outline View in Text Mode?

Question

How do I customize the ***Outline*** *view* in **Text** mode?

Answer

Suppose that you have the following XML document:

```
<doc startnumber="15">
```

```
    <sec counter="no">
        <info/>
        <title>Introduction</title>
      </sec>
    <sec>
        <title>Section title</title>
      <para>Content</para>
        <sec>
            <title>Section title</title>
                <para>Content</para>
        </sec>
    </sec>
    <sec>
            <title>Section title</title>
        <para>Content</para>
      </sec>
</doc>
```

and you want to display the XML content in a simplified Outline view like this:

```
doc "15"
sec Introduction
sec 15 Section title
sec 15.1 Section title
sec 16 Section title
```

Usually an Outline should have the following characteristics:

1. Double clicking in the Outline the corresponding XML content would get selected.
2. When the cursor moves in the opened XML document the Outline would select the proper entry.
3. When modifications occur in the document, the Outline would refresh.

A simple implementation using a Workspace Access plugin type could be something like this:

```
/**
 * Simple Outline for Text mode based on executing XPaths over the text content.
 */
 public class CustomWorkspaceAccessPluginExtension implements
WorkspaceAccessPluginExtension {
  /**
   * The custom outline list.
   */
  private JList customOutlineList;

  /**
   * Maps outline nodes to ranges in document
   */
  private WSXMLTextNodeRange[] currentOutlineRanges;

  /**
   * The current text page
   */
  private WSXMLTextEditorPage currentTextPage;

  /**
   * Disable CaretListener when we select from the CaretListener.
   */
  private boolean enableCaretListener = true;

  /**
   * @see WorkspaceAccessPluginExtension#applicationStarted
(ro.sync.exml.workspace.api.standalone.StandalonePluginWorkspace)
   */
  @Override
  public void applicationStarted
(final StandalonePluginWorkspace pluginWorkspaceAccess) {
    pluginWorkspaceAccess.addViewComponentCustomizer
(new ViewComponentCustomizer() {
      /**
       * @see ViewComponentCustomizer#customizeView
(ro.sync.exml.workspace.api.standalone.ViewInfo)
       */
      @Override
      public void customizeView(ViewInfo viewInfo) {
        if(
            //The view ID defined in the "plugin.xml"
            "SampleWorkspaceAccessID".equals(viewInfo.getViewID())) {
          customOutlineList = new JList();
          //Render the content in the Outline.
          customOutlineList.setCellRenderer(new DefaultListCellRenderer() {
       /**
        * @see javax.swing.DefaultListCellRenderer#getListCellRendererComponent
(javax.swing.JList, java.lang.Object, int, boolean, boolean)
        */
      @Override
      public Component getListCellRendererComponent
(JList<?> list, Object value, int index,
          boolean isSelected, boolean cellHasFocus) {
        JLabel label = (JLabel) super.getListCellRendererComponent
```

```
(list, value, index, isSelected, cellHasFocus);
        String val = null;
        if(value instanceof Element) {
          Element element = ((Element)value);
          val = element.getNodeName();
          if(!"".equals(element.getAttribute("startnumber"))) {
            val += " " + "'" + element.getAttribute("startnumber") + "'";
          }
          NodeList titles = element.getElementsByTagName("title");
          if(titles.getLength() > 0) {
            val += " \"" + titles.item(0).getTextContent() + "\"";
         }
       }
        label.setText(val);
        return label;
      }
    });
    //When we click a node, select it in the text page.
    customOutlineList.addMouseListener(new MouseAdapter() {
     @Override
     public void mouseClicked(MouseEvent e) {
       if(SwingUtilities.isLeftMouseButton(e) && e.getClickCount() == 2) {
          int sel = customOutlineList.getSelectedIndex();
          enableCaretListener = false;
         try {
          currentTextPage.select(currentTextPage.getOffsetOfLineStart
(currentOutlineRanges[sel].getStartLine()) +
currentOutlineRanges[sel].getStartColumn() - 1,
            currentTextPage.getOffsetOfLineStart
(currentOutlineRanges[sel].getEndLine()) +
currentOutlineRanges[sel].getEndColumn());
          } catch (BadLocationException e1) {
           e1.printStackTrace();
         }
          enableCaretListener = true;
        }
      }
    });
     viewInfo.setComponent(new JScrollPane(customOutlineList));
     viewInfo.setTitle("Custom Outline");
  }
 }
});

pluginWorkspaceAccess.addEditorChangeListener(new WSEditorChangeListener() {
  /**
  * @see WSEditorChangeListener#editorOpened(java.net.URL)
  */
  @Override
  public void editorOpened(URL editorLocation) {
   //An editor was opened
   WSEditor editorAccess = pluginWorkspaceAccess.getEditorAccess
(editorLocation, StandalonePluginWorkspace.MAIN_EDITING_AREA);
        if(editorAccess != null) {
          WSEditorPage currentPage = editorAccess.getCurrentPage();
          if(currentPage instanceof WSXMLTextEditorPage) {
            //User editing in Text mode an opened XML document.
            final WSXMLTextEditorPage xmlTP = (WSXMLTextEditorPage) currentPage;
            //Reconfigure outline on each change.
            xmlTP.getDocument().addDocumentListener(new DocumentListener() {
              @Override
              public void removeUpdate(DocumentEvent e) {
                reconfigureOutline(xmlTP);
              }
              @Override
              public void insertUpdate(DocumentEvent e) {
                reconfigureOutline(xmlTP);
              }
              @Override
              public void changedUpdate(DocumentEvent e) {
                reconfigureOutline(xmlTP);
              }
            });
            JTextArea textComponent = (JTextArea) xmlTP.getTextComponent();
            textComponent.addCaretListener(new CaretListener() {
              @Override
              public void caretUpdate(CaretEvent e) {
              if(currentOutlineRanges != null && currentTextPage != null &&
enableCaretListener) {
                  enableCaretListener = false;
                  //Find the node to select in the outline.
                  try {
                    int line = xmlTP.getLineOfOffset(e.getDot());
                    for (int i = currentOutlineRanges.length - 1; i >= 0; i--) {
              if(line > currentOutlineRanges[i].getStartLine() &&
line < currentOutlineRanges[i].getEndLine()) {
                        customOutlineList.setSelectedIndex(i);
                        break;
                      }
                    }
                  } catch (BadLocationException e1) {
                    e1.printStackTrace();
                  }
                  enableCaretListener = true;
                }
```

```
              }
            });
          }
        }
      }
      /**
       * @see WSEditorChangeListener#editorActivated(java.net.URL)
       */
      @Override
      public void editorActivated(URL editorLocation) {
        //An editor was selected, reconfigure the common outline
        WSEditor editorAccess = pluginWorkspaceAccess.getEditorAccess
(editorLocation, StandalonePluginWorkspace.MAIN_EDITING_AREA);
        if(editorAccess != null) {
          WSEditorPage currentPage = editorAccess.getCurrentPage();
          if(currentPage instanceof WSXMLTextEditorPage) {
            //User editing in Text mode an opened XML document.
            WSXMLTextEditorPage xmlTP = (WSXMLTextEditorPage) currentPage;
            reconfigureOutline(xmlTP);
          }
        }
      }
    }, StandalonePluginWorkspace.MAIN_EDITING_AREA);
  }

  /**
   * Reconfigure the outline
   *
   * @param xmlTP The XML Text page.
   */
  protected void reconfigureOutline(final WSXMLTextEditorPage xmlTP) {
    try {
      //These are DOM nodes.
      Object[] evaluateXPath = xmlTP.evaluateXPath("//doc | //sec");
      //These are the ranges each node takes in the document.
      currentOutlineRanges = xmlTP.findElementsByXPath("//doc | //sec");
      currentTextPage = xmlTP;
      DefaultListModel listModel = new DefaultListModel();
      if(evaluateXPath != null) {
        for (int i = 0; i < evaluateXPath.length; i++) {
          listModel.addElement(evaluateXPath[i]);
        }
      }
      customOutlineList.setModel(listModel);
    } catch(XPathException ex) {
      ex.printStackTrace();
    }
  }

  /**
   * @see WorkspaceAccessPluginExtension#applicationClosing()
   */
  @Override
  public boolean applicationClosing() {
    return true;
  }
}
```

Difference Between a Framework (Document Type) and a Plugin Extension

Question

What is the difference between a *Framework* and a *Plugin* Extension?

Answer

Two ways of customizing the application are possible:

1. Implement a *plugin*.

   A *plugin* serves a general purpose and influences any type of XML file that you open in Oxygen XML Editor.

   For the Oxygen XML Editor Plugins API, Javadoc, samples, and documentation, go to *https://www.oxygenxml.com/oxygen_sdk.html#Developer_Plugins*
2. *Create or modify the document type* that is associated to your specific XML vocabulary.

   This document type can be used, for instance, to provide custom actions for your type of XML files and to mount them on the toolbar, menus, and contextual menus.

   For example, if the end users are editing DITA documents, all the toolbar actions that are specific for DITA are provided by the DITA *framework*. If you look in the ***Document Type Association*** *preferences page* there is a *DITA* document type. If you edit that document type you will see that it has an **Author** tab in the ***Document***

Type Configuration *dialog box*. The subtabs in this tab can be used to define custom DITA actions and add them to the toolbars, main menus, or contextual menus.

For information about developing your own document types (*frameworks*), see the *Advanced Framework Customization* on page 1169 section.

If you look on disk in the `[OXYGEN_INSTALL_DIR]\frameworks\dita` folder, there is a file called `dita.framework`. That file gets updated when you edit a document type from the ***Document Type Association*** *preferences page*. Then you can share that updated file with all users.

The same folder contains some *JAR* libraries. These libraries contain custom Java operations that are called when the user presses certain toolbar actions.

We have an *Oxygen SDK* that contains the Java sources from all the DITA Java customizations:

https://www.oxygenxml.com/oxygen_sdk.html#XML_Editor_Authoring_SDK

Important: It is possible for a *plugin* to share the same classes with a *framework*. For further details, go to *How to Share the Classloader Between a Framework and a Plugin*.

Related Information:

Add a Custom Operation to an Existing Framework on page 1195

Disable Context-Sensitive Menu Items for Custom Author Actions

Question

Is there a way to disable menu items for custom **Author** mode actions depending on the cursor context?

Answer

By default, Oxygen XML Editor does not toggle the enabled/disabled states for actions based on whether or not the activation XPath expressions for that certain **Author** mode action are fulfilled. This is done because the actions can be many and evaluating XPath expression on each cursor move can lead to performance problems. However, if you have your own `ro.sync.ecss.extensions.api.ExtensionsBundle` implementation you can overwrite the method:

`ro.sync.ecss.extensions.api.ExtensionsBundle.createAuthorExtensionStateListener()`

and when the extension state listener gets activated, you can use the API like this:

```
    /**
     * @see ro.sync.ecss.extensions.api.AuthorExtensionStateListener#activated
(ro.sync.ecss.extensions.api.AuthorAccess)
     */
    public void activated(final AuthorAccess authorAccess) {

      //Add a caret listener to enable/disable extension actions:
authorAccess.getEditorAccess().addAuthorCaretListener(new AuthorCaretListener()
{
    @Override
    public void caretMoved(AuthorCaretEvent caretEvent) {
      try {
        Map<String, Object> authorExtensionActions =
authorAccess.getEditorAccess().getActionsProvider().getAuthorExtensionActions();
        //Get the action used to insert a paragraph. It's ID is "paragraph"
        AbstractAction insertParagraph = (
AbstractAction) authorExtensionActions.get("paragraph");
        //Evaluate an XPath expression in context of the current node
     Object[] evaluateXPath = authorAccess.getDocumentController().evaluateXPath
(".[ancestor-or-self::p]", false, false, false, false);
        if(evaluateXPath != null && evaluateXPath.length > 0 &&
evaluateXPath[0] != null) {
          //We are inside a paragraph, disable the action.
          insertParagraph.setEnabled(false);
        } else {
          //Enable the action
          insertParagraph.setEnabled(true);
        }
      } catch (AuthorOperationException e) {
        e.printStackTrace();
      }
    }
  });
```

When the extension is deactivated, you should remove the `CaretListener` to avoid adding multiple listeners that perform the same functionality.

Dynamically Add Form Controls Using a Styles Filter

Question

How do I add form controls using an API?

Answer

Usually, a form control is added from the CSS using one of the *built-in form controls*. However, in some cases you do not have all the information you need to properly initialize the form control at CSS level. In these cases you can add the form controls by using the API, more specifically *`ro.sync.ecss.extensions.api.StylesFilter`*.

For instance, if you want a combo box form control and the values to populate the combo are specified inside a file (or they come from a database). Here is how to add the form control from the API:

```
public class SDFStylesFilter implements StylesFilter {

 public Styles filter(Styles styles, AuthorNode authorNode) {
  if(authorNode.getType() == AuthorNode.NODE_TYPE_PSEUDO_ELEMENT
      && "before".equals(authorNode.getName())) {
    authorNode = authorNode.getParent();
   if ("country".equals(authorNode.getName())) {
    // This is the BEFORE pseudo element of the "country" element.
    // Read the supported countries from the configuration file.
    Map<String, Object> formControlArgs = new HashMap<String, Object>();
    formControlArgs.put(InplaceEditorArgumentKeys.PROPERTY_EDIT, "#text");
    formControlArgs.put(InplaceEditorArgumentKeys.PROPERTY_TYPE,
          InplaceEditorArgumentKeys.TYPE_COMBOBOX);
    // This will be a comma separated enumeration: France, Spain, Great Britain
    String countries = readCountriesFromFile();
    formControlArgs.put(InplaceEditorArgumentKeys.PROPERTY_VALUES, countries);
    formControlArgs.put(InplaceEditorArgumentKeys.PROPERTY_EDITABLE, "false");

    // We also add a label in form of the form control.
    Map<String, Object> labelProps = new HashMap<String, Object>();
    labelProps.put("text", "Country: ");
    labelProps.put("styles", "* {width: 100px; color: gray;}");
    StaticContent[] mixedContent = new StaticContent[]
           {new LabelContent(labelProps),new EditorContent(formControlArgs)};
    styles.setProperty(Styles.KEY_MIXED_CONTENT, mixedContent);
      }
    }

    // The added form control is the only way the element can be edited.
    if ("country".equals(authorNode.getName())) {
      styles.setProperty(Styles.KEY_VISIBITY, "-oxy-collapse-text");
    }

    return styles;
  }
}
```

If the execution of the `formControlArgs.put(InplaceEditorArgumentKeys.PROPERTY_VALUES, countries);` line consumes too much execution time (for example, if it connects to a database or if it needs to extract data from a very large file), you can choose to delay it until the values are actually needed by the form control. This approach is called *lazy evaluation* and can be implemented as follows:

```
formControlArgs.put(InplaceEditorArgumentKeys.PROPERTY_VALUES,
           new LazyValue<List<CIValue>>() {
  public java.util.List<CIValue> get() {
    // We avoid reading the possible values until they are actually requested.
    // This will be a List with CIValues created over countries:
           France, Spain, Great Britain
    return readCountriesFromFile();
  }
});
```

The lazy evaluation approach can be used for the following form controls properties:

- `InplaceEditorArgumentKeys.PROPERTY_VALUES`
- `InplaceEditorArgumentKeys.PROPERTY_LABELS`
- `InplaceEditorArgumentKeys.PROPERTY_TOOLTIPS`

The full source code for this example is available inside the *Oxygen SDK*.

Dynamically Modify the Content Inserted by the Author

Question

Is there a way to insert typographic quotation marks instead of double quotes?

Answer

By using the API you can set a document filter to change the text that is inserted in the document in **Author** mode. You can use this method to change the insertion of double quotes with the typographic quotes.

Here is some sample code:

```
    authorAccess.getDocumentController().setDocumentFilter
(new AuthorDocumentFilter() {
    /**
     * @see ro.sync.ecss.extensions.api.AuthorDocumentFilter#insertText
(ro.sync.ecss.extensions.api.AuthorDocumentFilterBypass, int, java.lang.String)
     */
    @Override
    public void insertText(AuthorDocumentFilterBypass filterBypass,
int offset, String toInsert) {
      if(toInsert.length() == 1 && "\"".equals(toInsert)) {
        //User typed a quote but he actually needs a smart quote.
        //So we either have to add \u201E (start smart quote)
        //Or we add \u201C (end smart quote)
        //Depending on whether there's already a start smart quote inserted
in the current paragraph.

      try {
        AuthorNode currentNode =
authorAccess.getDocumentController().getNodeAtOffset(offset);
        int startofTextInCurrentNode = currentNode.getStartOffset();
        if(offset > startofTextInCurrentNode) {
          Segment seg = new Segment();
        authorAccess.getDocumentController().getChars(startofTextInCurrentNode,
offset - startofTextInCurrentNode, seg);
          String previosTextInNode = seg.toString();
          boolean insertStartQuote = true;
          for (int i = previosTextInNode.length() - 1; i >= 0; i--) {
            char ch = previosTextInNode.charAt(i);
            if('\u201C' == ch) {
              //Found end of smart quote, so yes, we should insert a start one
              break;
            } else if('\u201E' == ch) {
              //Found start quote, so we should insert an end one.
              insertStartQuote = false;
              break;
            }
          }

          if(insertStartQuote) {
            toInsert = "\u201E";
          } else {
            toInsert = "\u201C";
          }
        }
      } catch (BadLocationException e) {
        e.printStackTrace();
      }
    }
    System.err.println("INSERT TEXT |" + toInsert + "|");
    super.insertText(filterBypass, offset, toInsert);
  }
});
```

You can find the online Javadoc for `AuthorDocumentFilter` API here: *https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/ecss/extensions/api/AuthorDocumentFilter.html*

An alternative to using a document filtering is the use of a *`ro.sync.ecss.extensions.api.AuthorSchemaAwareEditingHandlerAdapter`*, which has clear callbacks indicating the source from where the API is called (Paste, Drag and Drop, Typing).

Dynamically Open File in Oxygen XML Editor Distributed via JavaWebStart

Question

How can we dynamically open a file in an Oxygen XML Editor distributed via JWS?

Answer

The JWS packager Ant build file that is included with Oxygen XML Editor signs by default the JNLP file (this means that a copy of it is included in the main *JAR* library) in this step:

```
<copy file="${outputDir}/${packageName}/${productName}.jnlp" tofile="${home}/JNLP-INF/APPLICATION.JNLP"/>
```

Signing the JNLP file is required by newer Java versions and means that it is impossible to automatically generate a JNLP file containing some dynamic arguments. The solution is to use the signed JNLP template feature of Java 7, bundle inside the *JAR* library a signed `APPLICATION_TEMPLATE.JNLP` instead of an `APPLICATION.JNLP` with a wildcard command line argument:

```
<application-desc main-class="ro.sync.jws.JwsDeployer">
   <argument>*</argument>
</application-desc>
```

Then you can replace the wildcard in the external placed JNLP to the actual, dynamic command line arguments value.

A different, more complicated approach would be to have the JNLP file signed and always referenced as a URL argument a location like this:

```
http://path/to/server/redirectEditedURL.php
```

When the URL gets clicked on the client side you would also call a PHP script on the server side that would update the redirect location for `redirectEditedURL.php` to point to the clicked XML resource. Then the opened Oxygen XML Editor would try to connect to the redirect PHP and be redirected to open the XML.

Extend the Java Functionality of an Existing Framework (Document Type)

Question

How can I change the way a DocBook 4 `xref` displays in **Author** mode based on what element is at the `linkend`?

Answer

Follow these steps:

1. Create a Maven Java project and add a dependency on the Oxygen XML Editor classes:

   ```
   <dependency>
       <groupId>com.oxygenxml</groupId>
       <artifactId>oxygen-sdk</artifactId>
       <version>${oxygen.version}</version>
   </dependency>
   ```

   where `${oxygen.version}` is the version of Oxygen XML Editor.

   Alternatively, if the project does not use Maven, all the transitive dependencies of the above Maven artifact need to be added to the classpath of the project.
2. Also add the *[OXYGEN_INSTALL_DIR]*`\frameworks\docbook\docbook.jar` to the class path of the project.
3. Create a class that extends *`ro.sync.ecss.extensions.docbook.DocBook4ExtensionsBundle`* and overwrites the method: *`ro.sync.ecss.extensions.api.ExtensionsBundle#createLinkTextResolver()`*
4. For your custom resolver implementation you can start from the Java sources of the *`ro.sync.ecss.extensions.docbook.link.DocbookLinkTextResolver`* (the Java code for the entire DocBook customization is present in a subfolder in the *Oxygen SDK*).
5. Pack your extension classes in a *JAR* file. Copy the *JAR* to: *[OXYGEN_INSTALL_DIR]*`\frameworks\docbook\custom.jar`.
6. Start Oxygen XML Editor.

7. *Open the **Preferences** dialog box **(Options > Preferences)*** and go to **Document Type Association**. Edit the DocBook 4 document type. In the **Classpath** list add the path to the new *JAR*. In the extensions list select your custom extension instead of the regular DocBook one.
8. You can rename the document type and the `docbook` *framework* folder to something else (such as `custom_docbook`) and share it with others. A document type can also be installed using our *add-on support*.

Impose Custom Options for Authors

Question

How to enable *Track Changes* at startup?

Answer

There are two ways to enable *Track Changes* for every document that you open:

1. You could *customize the default options* that are used by your authors and set the ***Track Changes - Initial State** option* to **Always On**.
2. Use an API to toggle the *Track Changes* state after a document is opened in **Author** mode:

```
// Check the current state of Track Changes
boolean trackChangesOn = authorAccess.getReviewController().isTrackingChanges();
if (!trackChangesOn) {
  // Set Track Changes state to On
  authorAccess.getReviewController().toggleTrackChanges();
}
```

Insert an Element with all the Required Content

Question

I am inserting a DITA *image* XML element using the API that points to a certain resource and has required content. Can the required content be automatically inserted by the application?

Answer

The API `ro.sync.ecss.extensions.api.AuthorSchemaManager` can propose valid elements that can be inserted at the specific offset. Using the method `AuthorSchemaManager.createAuthorDocumentFragment(CIElement)`, you can convert the proposed elements to *document fragments* (which have all the required content filled in) that can then be inserted in the document.

```
    AuthorSchemaManager schemaManager =
this.authorAccess.getDocumentController().getAuthorSchemaManager();

    WhatElementsCanGoHereContext context =
schemaManager.createWhatElementsCanGoHereContext
(this.authorAccess.getEditorAccess().getCaretOffset());

    List<CIElement> possibleElementsAtCaretPosition =
schemaManager.whatElementsCanGoHere(context);
    loop: for (int i = 0; i < possibleElementsAtCaretPosition.size(); i++) {
      CIElement possibleElement = possibleElementsAtCaretPosition.get(i);
      List<CIAttribute> attrs = possibleElement.getAttributes();
      if(attrs != null) {
        for (int j = 0; j < attrs.size(); j++) {
          CIAttribute ciAttribute = attrs.get(j);
          if (ciAttribute.getName().equals("class")) {
            if (ciAttribute.getDefaultValue() != null
                && ciAttribute.getDefaultValue().contains("  topic/image ")) {
              //Found a CIElement for image
   //Create a fragment that contains all required child elements already built.
              AuthorDocumentFragment frag =
schemaManager.createAuthorDocumentFragment(possibleElement);
              //Now set the @href to it.
              //Ask the user and obtain a value for the @href
              //Then:

          String href = "test.png";
          List<AuthorNode> nodes = frag.getContentNodes();
          if(!nodes.isEmpty()) {
            AuthorElement imageEl = (AuthorElement) nodes.get(0);
            imageEl.setAttribute("href", new AttrValue(href));
          }
```

```
          //And insert the fragment.
          this.authorAccess.getDocumentController().insertFragment
(this.authorAccess.getEditorAccess().getCaretOffset(), frag);
          break loop;
        }
      }
    }
  }
}
```

Related Information:

AuthorDocumentFragment Class

Modify the XML Content on Open

Question

I have a bunch of DITA documents that have a fixed path the image `src` attributes. These paths are not valid and I am trying to move away from this practice by converting it in to relative paths. When an XML document is opened, can I trigger the Java API to change the fixed path to a relative path?

Answer

The Plugins SDK: *https://www.oxygenxml.com/oxygen_sdk.html#Developer_Plugins* contains a sample plugin type called *WorkspaceAccess*. Such a plugin is notified when the application starts and it can do what you want in a couple of ways:

1. Add a listener that notifies you when the user opens an XML document. Then if the XML document is opened in the **Author** visual editing mode you can use our *Author API* to change attributes:

```
   pluginWorkspaceAccess.addEditorChangeListener(new  WSEditorChangeListener() {
      /**
       * @see WSEditorChangeListener#editorOpened(java.net.URL)
       */
      @Override
      public void editorOpened(URL editorLocation) {
        WSEditor openedEditor = pluginWorkspaceAccess.getCurrentEditorAccess
(StandalonePluginWorkspace.MAIN_EDITING_AREA);
        if(openedEditor.getCurrentPage() instanceof WSAuthorEditorPage) {
        WSAuthorEditorPage authPage = (WSAuthorEditorPage)
openedEditor.getCurrentPage();
        AuthorDocumentController docController =
authPage.getDocumentController();
        try {
         //All changes will be undone by pressing Undo once.
         docController.beginCompoundEdit();
         fixupImageRefs(docController,
          docController.getAuthorDocumentNode());
        } finally {
           docController.endCompoundEdit();
          }
        }
      }

      private void fixupImageRefs
(AuthorDocumentController docController, AuthorNode authorNode) {
          if(authorNode instanceof AuthorParentNode) {
            //Recurse
            List<AuthorNode> contentNodes =
((AuthorParentNode)authorNode).getContentNodes();
            if(contentNodes != null) {
               for (int i = 0; i < contentNodes.size(); i++) {
                 fixupImageRefs(docController, contentNodes.get(i));
               }
             }
          }
          if(authorNode.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
            AuthorElement elem = (AuthorElement) authorNode;
            if("image".equals(elem.getLocalName())) {
               if(elem.getAttribute("href") != null) {
                 String originalHref = elem.getAttribute("href").getValue();
    URL currentLocation = docController.getAuthorDocumentNode().getXMLBaseURL();
               //TODO here you compute the new href.
               String newHref = null;
              docController.setAttribute("href", new AttrValue(newHref), elem);
             }
          }
        }
      }
    },
  StandalonePluginWorkspace.MAIN_EDITING_AREA);
```

2. An API to open XML documents in the application:

```
ro.sync.exml.workspace.api.Workspace.open(URL)
```

So you can create up a plugin that automatically opens one by one XML documents from a certain folder in the application, makes modifications to them, saves the content by calling:

```
ro.sync.exml.workspace.api.editor.WSEditorBase.save()
```

and then closes the editor:

```
ro.sync.exml.workspace.api.Workspace.close(URL)
```

Modify the XML Content on Save

Question

Is it possible to get Oxygen XML Editor to update the revised date on a DITA document when it's saved?

Answer

The Plugins SDK: *https://www.oxygenxml.com/oxygen_sdk.html#Developer_Plugins* contains a sample plugin type called *WorkspaceAccess*.Such a plugin is notified when the application starts.

You can add a listener that notifies you before the user saves an XML document. Then if the XML document is opened in the **Author** visual editing mode you can use our *Author API* to change attributes before the save takes place:

```
      @Override
      public void applicationStarted
(final StandalonePluginWorkspace pluginWorkspaceAccess) {
       pluginWorkspaceAccess.addEditorChangeListener
(new WSEditorChangeListener(){
         //An editor was opened
        @Override
        public void editorOpened(URL editorLocation) {
          final WSEditor editorAccess = pluginWorkspaceAccess.getEditorAccess
(editorLocation, PluginWorkspace.MAIN_EDITING_AREA);
          if(editorAccess != null){
            editorAccess.addEditorListener
(new ro.sync.exml.workspace.api.listeners.WSEditorListener(){
              //Editor is about to be saved
              @Override
              public boolean editorAboutToBeSavedVeto(int operationType) {
                if(EditorPageConstants.PAGE_AUTHOR.equals
(editorAccess.getCurrentPageID())){
                  WSAuthorEditorPage authorPage =
(WSAuthorEditorPage) editorAccess.getCurrentPage();
                  AuthorDocumentController controller =
authorPage.getDocumentController();
                  try {
                    //Find the revised element
                    AuthorNode[] nodes = controller.findNodesByXPath
("//revised", true, true, true);
                    if(nodes != null && nodes.length > 0){
                      AuthorElement revised = (AuthorElement) nodes[0];
                      //Set the modified attribute to it...
                      controller.setAttribute("modified",
new AttrValue(new Date().toString()), revised);
                    }
                  } catch (AuthorOperationException e) {
                    e.printStackTrace();
                  }
                }
                //And let the save continue..
                return true;
              }
            });
          }
        }
       }, PluginWorkspace.MAIN_EDITING_AREA);
      }
```

Multiple Rendering Modes for the Same Document in Author Mode

Question

How can we add multiple buttons, each showing a different visualization mode of the same document in **Author** mode?

Answer

In the toolbar of the **Author** mode there is a **Styles** drop-down menu that contains *alternate CSS styles* for the same document. To add an *alternate* CSS stylesheet, *open the **Preferences** dialog box **(Options** > **Preferences)***, go to **Document Type Association**, select the document type associated with your documents and press **Edit**. In the ***Document Type** configuration dialog box* that appears, go to the **Author** tab, and in the **CSS** subtab add references to *alternate* CSS stylesheets.

For example, one of the alternate CSS stylesheets that we offer for the DITA document type is located here:

```
[OXYGEN_INSTALL_DIR]/frameworks/dita/css_classed/hideColspec.css
```

If you open it, you will see that it imports the *main CSS* and then adds selectors of its own.

Obtain a DOM Element from `AuthorNode` or `AuthorElement`

Question

Can a DOM Element be obtained from an `AuthorNode` or an `AuthorElement`?

Answer

No, a DOM Element cannot be obtained from an `AuthorNode` or an `AuthorElement`. The `AuthorNode` structure is also hierarchical but the difference is that all the text content is kept in a single text buffer instead of having individual text nodes.

We have an image in the Javadoc documentation that explains this situation: *https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/ecss/extensions/api/node/AuthorDocumentFragment.html*

Obtain the Currently Selected Element Using the Author API

Question

In **Author** mode, if an element is fully selected, I want to perform an action on it. If not, I want to perform an action on the node that is located at the cursor position. Is this possible via the API?

Answer

When an element is fully selected by the user the selection start and end offsets are actually outside of the node's offset bounds. So using `AuthorDocumentController.getNodeAtOffset` will actually return the parent of the selected node. We have some special API that makes it easier for you to determine this situation: `WSAuthorEditorPageBase.getFullySelectedNode()`.

```
AuthorDocumentController controller = authorPageAccess.getDocumentController();
AuthorAccess authorAccess = authorPageAccess.getAuthorAccess();
int caretOffset = authorAccess.getEditorAccess().getCaretOffset();

    AuthorElement nodeAtCaret =
(AuthorElement) authorAccess.getEditorAccess().getFullySelectedNode();
    if (nodeAtCaret == null) {
     //We have no fully selected node. We can look at the cursor offset.
     nodeAtCaret = (AuthorElement)
authorAccess.getDocumentController().getNodeAtOffset(caretOffset);
    //Or we could look at the selection start and end, see which node is
the parent of each offset and get the closest common ancestor.
}
```

Run XSLT or XQuery Transformations

Question

Can I run XSL 2.0 / 3.0 transformation with Saxon EE using the *Oxygen SDK*?

Answer

The API class `ro.sync.exml.workspace.api.util.XMLUtilAccess` allows you to create an XSLT Transformer that implements the JAXP interface `javax.xml.transform.Transformer`. Then this type of transformer can be used to transform XML. Here's just an example of transforming when you have an `AuthorAccess` API available:

```
     InputSource is = new org.xml.sax.InputSource
(URLUtil.correct(new File("test/personal.xsl")).toString());
     xslSrc = new SAXSource(is);
     javax.xml.transform.Transformer transformer =
authorAccess.getXMLUtilAccess().createXSLTTransformer
(xslSrc, null, AuthorXMLUtilAccess.TRANSFORMER_SAXON_ENTERPRISE_EDITION);
     transformer.transform(new StreamSource(new File("test/personal.xml")),
new StreamResult(new File("test/personal.html")));
```

If you want to create the transformer from the *plugin* side, you can use this method instead: `ro.sync.exml.workspace.api.PluginWorkspace.getXMLUtilAccess()`.

Save a New Document with a Predefined File Name Pattern

Question

Is it possible to get Oxygen XML Editor to automatically generate a file name comprising a UUID plus file extension using the SDK?

Answer

This could be done implementing a *plugin* for Oxygen XML Editor using the Plugins SDK:

https://www.oxygenxml.com/oxygen_sdk.html#Developer_Plugins

There is a type of *plugin* called *Workspace Access* that can be used to add a listener to be notified before an opened editor is saved. The implemented *plugin* would intercept the save events when a newly created document is untitled and display an alternative chooser dialog box, then save the topic with the proper name.

The Java code would look like this:

```
  private static class CustomEdListener extends WSEditorListener{
    private final WSEditor editor;
    private final StandalonePluginWorkspace
        pluginWorkspaceAccess;
    private boolean saving = false;
    public CustomEdListener
(StandalonePluginWorkspace pluginWorkspaceAccess, WSEditor editor) {
            this.pluginWorkspaceAccess = pluginWorkspaceAccess;
            this.editor = editor;
    }
    @Override
    public boolean editorAboutToBeSavedVeto(int operationType) {
      if(! saving &&
        editor.getEditorLocation().toString().contains("Untitled")) {
        File chosenDir = pluginWorkspaceAccess.chooseDirectory();
        if(chosenDir != null) {
          final File chosenFile =
new File(chosenDir, UUID.randomUUID().toString() + ".dita");
        SwingUtilities.invokeLater(new Runnable() {
            @Override
              public void run() {
              try {
                saving = true;
                editor.saveAs(new URL(chosenFile.toURI().toASCIIString()));
              } catch (MalformedURLException e) {
                e.printStackTrace();
              } finally {
                saving = false;
              }
            }
          });
        }
```

```
        //Reject the original save request.
        return false;
      }
      return true;
    }
  }

    @Override
    public void applicationStarted
(final StandalonePluginWorkspace pluginWorkspaceAccess) {
    pluginWorkspaceAccess.addEditorChangeListener(new WSEditorChangeListener() {
      @Override
      public void editorOpened(URL editorLocation) {
        final WSEditor editor = pluginWorkspaceAccess.getEditorAccess
(editorLocation, PluginWorkspace.MAIN_EDITING_AREA);
      if(editor !=
null && editor.getEditorLocation().toString().contains("Untitled")) {

        //Untitled editor
 editor.addEditorListener(new CustomEdListener(pluginWorkspaceAccess, editor));
        }
      }
     },
  PluginWorkspace.MAIN_EDITING_AREA);
...............................................
```

Split Paragraph on Enter (Instead of Showing Content Completion List)

Question

How to split the paragraph on Enter instead of showing the content completion list?

Answer

To obtain this behavior, edit your document type and in the ***Document Type*** *configuration dialog box* go to the **Author** tab, then **Actions** subtab, and add your own split action. This action must have the **Enter** shortcut key associated and must trigger your own custom operation that handles the split.

So, when you press **Enter**, your Java operation is invoked and it will be your responsibility to split the paragraph using the current API (probably creating a *document fragment* from the cursor offset to the end of the paragraph, removing the content and then inserting the created fragment after the paragraph).

This solution has as a drawback. Oxygen XML Editor hides the *Content Completion Assistant* when you press **Enter**. If you want to show allowed child elements at that certain offset, implement your own content proposals window using the `ro.sync.ecss.extensions.api.AuthorSchemaManager` API to use information from the associated schema.

Use Custom Rendering Styles for Entity References, Comments, or PIs

Problem

Is there a way to display entity references in the **Author** mode without the distinct gray background and tag markers?

Solution

There is a built-in CSS stylesheet in the Oxygen XML Editor libraries that is used when styling content in the **Author** mode, no matter what CSS you use. This CSS has the following content:

```
@namespace oxy url('http://www.oxygenxml.com/extensions/author');
@namespace xi "http://www.w3.org/2001/XInclude";
@namespace xlink "http://www.w3.org/1999/xlink";
@namespace svg "http://www.w3.org/2000/svg";
@namespace mml "http://www.w3.org/1998/Math/MathML";

oxy|document {
    display:block !important;
}

oxy|cdata {
    display:morph !important;
    white-space:pre-wrap !important;
    border-width:0px !important;
    margin:0px !important;
    padding: 0px !important;
}
```

```
oxy|processing-instruction {
    display:block !important;
    color: rgb(139, 38, 201) !important;
    white-space:pre-wrap !important;
    border-width:0px !important;
    margin:0px !important;
    padding: 0px !important;
}

oxy|comment {
    display:morph !important;
    color: rgb(0, 100, 0) !important;
    background-color:rgb(255, 255, 210) !important;
    white-space:pre-wrap !important;
    border-width:0px !important;
    margin:0px !important;
    padding: 0px !important;
}

oxy|reference:before,
oxy|entity[href]:before{
  link: attr(href) !important;
  text-decoration: underline !important;
  color: navy !important;

  margin: 2px !important;
  padding: 0px !important;
}

oxy|reference:before {
  display: morph !important;
  content: url(../images/editContent.gif) !important;
}

oxy|entity[href]:before{
  display: morph !important;
  content: url(../images/editContent.gif) !important;
}

oxy|reference,
oxy|entity {
    editable:false !important;
    background-color: rgb(240, 240, 240) !important;
    margin:0px !important;
    padding: 0px !important;
}

oxy|reference {
    display:morph !important;
}

oxy|entity {
    display:morph !important;
}

oxy|entity[href] {
  border: 1px solid rgb(175, 175, 175) !important;
  padding: 0.2em !important;
}

xi|include {
    display:block !important;
    margin-bottom: 0.5em !important;
    padding: 2px !important;
}
xi|include:before,
xi|include:after{
    display:inline !important;
    background-color:inherit !important;
    color:#444444 !important;
    font-weight:bold !important;
}

xi|include:before {
    content:url(../images/link.gif) attr(href)  !important;
    link: attr(href) !important;
}
xi|include[xpointer]:before {
    content:url(../images/link.gif) attr(href) " " attr(xpointer) !important;
    link: oxy_concat(attr(href), "#", attr(xpointer)) !important;
}

xi|fallback {
    display:morph !important;
    margin: 2px !important;
    border: 1px solid #CB0039 !important;
}

xi|fallback:before {
    display:morph !important;
    content:"XInclude fallback: " !important;
    color:#CB0039 !important;
}
```

```
oxy|doctype {
    display:block !important;
    background-color: transparent !important;
    color:blue !important;
    border-width:0px !important;
    margin:0px !important;
    padding: 2px !important;
}

oxy|error {
    display:morph !important;
    editable:false !important;
    white-space:pre !important;
    color: rgb(178, 0, 0) !important;
    font-weight:bold !important;
}

*[xlink|href]:before {
    content:url(../images/link.gif);
    link: attr(xlink|href) !important;
}


/*No direct display of the MathML and SVG images.*/
svg|svg{
 display:inline !important;
    white-space: trim-when-ws-only;
}
svg|svg svg|*{
    display:none !important;
    white-space:normal;
}

mml|math{
 display:inline !important;
    white-space: trim-when-ws-only;
}
mml|math mml|*{
    display:none !important;
    white-space: normal;
}
```

In the CSS used for rendering the XML in **Author** mode, do the following:

1. Import the special **Author** mode namespace.
2. Use a special selector to customize the `entity` node.

Example:

```
@namespace oxy url('http://www.oxygenxml.com/extensions/author');
oxy|entity {
   background-color: inherit !important;
   margin:0px !important;
   padding: 0px !important;
    -oxy-display-tags:none;
}
```

You can overwrite styles in the predefined CSS to customize style comments, processing instructions, and *CData* sections. You can also customize the way `xi:include` elements are rendered.

Using the Oxygen XML SDK

16

Topics:

- *Extending Oxygen XML Editor with Plugins*
- *Oxygen XML Author Component*
- *Oxygen XML Web Author Component Component*

Oxygen XML Editor has an SDK that can be used as a base to develop *frameworks* and *plugins*. It can be also used to create projects that use the Oxygen XML Author Component or Oxygen XML Web Author Component. The SDK is a Java library available under the *Oxygen XML SDK licensing terms* and is delivered with a set of examples that demonstrate how to extend Oxygen XML functionality through API calls. The SDK is available on our website at *https://www.oxygenxml.com/oxygen_sdk.html*.

Extending Oxygen XML Editor with Plugins

A *plugin* is a software component that adds extra functionality to the standalone version of the application using a series of application-provided extension points.

This chapter explains how to write and install a *plugin* for the standalone version of Oxygen XML Editor. The *Plugins Development Kit* contains sample *plugins* (source and compiled Java code) and the Javadoc API necessary for developing custom *plugins*.

If you want to customize the Oxygen XML Editor Eclipse plugin you can look at the *Eclipse IDE Integration Sample Project* to see how an Eclipse plugin can interact with the Oxygen XML Editor APIs.

General Configuration of an Oxygen XML Editor Plugin

The Oxygen XML Editor functionality can be extended with *plugins* that implement a clearly specified API. On the Oxygen XML Editor website, there is an *SDK* with sample *plugins* (source and compiled Java code) and the Javadoc API necessary for developing custom *plugins*.

The minimal implementation of a *plugin* must provide:

- A Java class that extends the `ro.sync.exml.plugin.Plugin` class.
- A Java class that implements the `ro.sync.exml.plugin.PluginExtension` interface.
- A *plugin* descriptor file called `plugin.xml`.

A `ro.sync.exml.plugin.PluginDescriptor` object is passed to the constructor of the subclass of the `ro.sync.exml.plugin.Plugin` class. It contains the following data items about the *plugin*:

- basedir (*File* object) - The base directory of the *plugin*.
- description (*String* object) - The description of the *plugin*.
- name (*String* object) - The name of the *plugin*.
- vendor (*String* object) - The vendor name of the *plugin*.
- version (*String* object) - The *plugin* version.
- id (*String* object) - A unique identifier.
- classLoaderType - You can choose between `preferOxygenResources` (default value) and `preferReferencedResources`. When choosing `preferOxygenResources`, the libraries that are referenced in the Oxygen XML Editor `lib` directory will have precedence over those referenced in the `plugin.xml` configuration file, if they have the same package names. When choosing `preferReferencedResources`, the libraries that are referenced in the `plugin.xml` configuration file will

have precedence over those found in the Oxygen XML Editor `lib` directory, if they have the same package names.

The *plugin* descriptor is an XML file that defines how the *plugin* is integrated in Oxygen XML Editor and what libraries are loaded. The structure of the *plugin* descriptor file is fully described in a DTD grammar located in *[OXYGEN_INSTALL_DIR]*`/plugins/plugin.dtd`. Here is a sample *plugin* descriptor used by the *Capitalize Lines* sample *plugin*:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plugin SYSTEM "../plugin.dtd">
<plugin
    name="Capitalize Lines"
    description="Capitalize the first character on each line"
    version="1.0.0"
    vendor="SyncRO"
    class="ro.sync.sample.plugin.caplines.CapLinesPlugin">
    <runtime>
        <library name="lib/caplines.jar"/>
    </runtime>
    <extension type="selectionProcessor"
    class="ro.sync.sample.plugin.caplines.CapLinesPluginExtension"
            keyboardShortcut="ctrl shift EQUALS"/>
</plugin>
```

If your *plugin* is of the **Selection**, **Document** or **General** types, and thus contributes an action either to the contextual menu or to the main menu of the **Text** editing mode, then you can assign a keyboard shortcut for it. You can use the `keyboardShortcut` attribute for each `extension` element to specify the desired shortcut.

Tip: To compose string representations of the desired shortcut keys you can go to the Oxygen XML Editor **Menu Shortcut Keys** preferences page, press **Edit** on any action, press the desired key sequence and use the representation that appears in the **Edit** dialog box.

Referencing Libraries

To reference libraries, use either of the following elements:

- `<library name="libraryName" scope="global"/>` - To point to specific libraries.

  Note: You can use the *${oxygenInstallDir} editor variable* as part of the value of the `name` attribute. You can also use a system variable (`${system(var.name)}`) or environment variable (`${env(VAR_NAME)}`).
- `<librariesFolder name="libraryFolderPath" scope="global"/>` - To point to multiple libraries located in the specified folder.

Both elements support the `scope` attribute that defines the loading priority. It can have one of the following three values:

- *local* - The library is loaded in the *plugin's* own class loader. This is the default behavior.
- *global* - The library is loaded in the main application class loader as the last library in the list (as if it would be present in the application `lib` directory).
- *globalHighPriority* - The library is loaded in the main application class loader as the first library in the list (useful to patch certain resources located in other *JARs* of the application).

Installing an Oxygen XML Editor Plugin

Choose one of the following methods to install a *plugin* in Oxygen XML Editor:

Automatic Method

To install a new add-on, follow these steps:

1. Go to **Help** > **Install new add-ons**.
2. In the displayed dialog box, fill-in the **Show add-ons from** with the *update site* that hosts add-ons. If you want to see all Oxygen XML Editor default add-ons, choose the **ALL AVAILABLE SITES** option from the **Show add-ons from** drop down. The add-ons list contains the name, status, update version, Oxygen XML Editor version, and the type of the add-on (either **framework**, or **plugin**). A short description of each add-on is presented under the add-ons list.

Note: To see all the add-ons from the remote update site, deselect **Show only compatible add-ons** and **Show only the latest version of the add-ons**. Incompatible add-ons are shown only to acknowledge their presence on the remote update site. You cannot install an incompatible add-on.

3. By default, only the latest versions of the add-ons that are compatible with the current version of Oxygen XML Editor are displayed.
4. Choose the add-ons you want to install, press the **Next** button, then follow the on-screen instructions.

   Note: Accepting the license agreement of the add-on is a mandatory step in the installation process.

   Note: All add-ons are installed in the `extensions` directory inside the Oxygen XML Editor *preferences directory*.

Manual Method

To manually install a *plugin* in Oxygen XML Editor, follow these steps:

1. Go to the Oxygen XML Editor installation directory and locate the `plugins` directory.

   Note: The `plugins` directory contains all the *plugins* available to Oxygen XML Editor.
2. In the `plugins` directory, create a subfolder to store the *plugin* files.
3. In the new folder, place the *plugin* descriptor file (`plugin.xml`), the Java classes of the *plugin*, and the other files that are referenced in the descriptor file.
4. Restart Oxygen XML Editor.

Types of Plugin Extensions Available with the SDK

A *plugin* can have one or more defined *plugin extensions* that provide functionality to the application. This section presents the *plugin extensions* that are available.

Author Stylesheet Plugin Extension

This type of *plugin* allows you to add a stylesheet (CSS or LESS) that renders elements in **Author** mode.

To specify additional stylesheets, edit the *plugin* descriptor and add `extension` elements that point to them, as in the following example:

```
<extension type="AuthorStylesheet" href="showTables.css"/>
<extension type="AuthorStylesheet" href="hideButtons.css"/>
```

Using this mechanism, you can add one or more CSS stylesheets to merge with the existing ones. Whenever you add a new stylesheet using this *plugin*, it will have priority over all other stylesheets applied on the file edited in **Author** mode.

If your implementation requires more flexibility (such as a dynamic change of the stylesheet), you should consider using the *`StylesFilter` plugin extension*.

Additional Framework Plugin Extension

This type of *plugin* allows you to add a new framework straight from the *plugin*.

To specify additional frameworks, edit the *plugin* descriptor and add `extension` elements that point to them, as in the following example:

```
<extension type="AdditionalFrameworks" path="framework_directory"/>
```

The path attribute should be a sub-directory of the plugin. If the *plugin* is *installed as an add-on*, the new framework will be set as read-only and editing it will only be possible if you *duplicate it*. If the *plugin* is installed in the *`[OXYGEN_INSTALL_DIR]`*`/plugins` directory, the new frameworks will be editable.

Components Validation Plugin Extension

This type of *plugin* allows you to customize the menus, toolbars, and other components by enabling or filtering them from the user interface.

This *plugin* provides the following API:

- The interface `ComponentsValidatorPluginExtension` - There is one method that must be implemented:

- `getComponentsValidator()` - Returns a *`ro.sync.exml.ComponentsValidator`* implementation class used for validating the menus, toolbars, and their actions.
- The `ComponentsValidator` interface provides methods to filter various features from being added to the GUI of Oxygen XML Editor:
  - `validateMenuOrTaggedAction(String[] menuOrActionPath)` - Checks if a menu or a tag action from a menu is allowed and returns a `boolean` value. A tag is used to uniquely identifying an action. The `String[]` argument is the tag of the menu / action and the tags of its parent menus if any.
  - `validateToolbarTaggedAction(String[] toolbarOrAction)` - Checks if an action from a toolbar is allowed and returns a *boolean* value. The `String[]` argument is the tag of the action from a toolbar and the tag of its parent toolbar if any.
  - `validateComponent(String key)` - Checks if the given component is allowed and returns a `boolean` value. The `String` argument is the tag identifying the component. You can remove toolbars entirely using this callback.
  - `validateAccelAction(String category, String tag)` - Checks if the given accelerator action is allowed to appear in the GUI and returns a `boolean` value. An accelerator action can be uniquely identified so it will be removed both from toolbars or menus. The first argument represents the action category, the second is the tag of the action.
  - `validateContentType(String contentType)` - Checks if the given content type is allowed and returns a `boolean` value. The `String` argument represents the content type. You can instruct Oxygen XML Editor to ignore content types such as `text / xsl` or `text / xquery`.
  - `validateOptionPane(String optionPaneKey)` - Checks if the given options page can be added in the preferences option tree and returns a `boolean` value. The `String` argument is the option pane key.
  - `validateOption(String optionKey)` - Checks if the given option can be added in the option page and returns a `boolean` value. The `String` argument is the option key. This method is mostly used for internal use and it is not called for each option in a preferences page.
  - `validateLibrary(String library)` - Checks if the given library is allowed to appear listed in the **About** dialog box and returns a `boolean` value. The `String` argument is the library. This method is mostly for internal use.
  - `validateNewEditorTemplate(EditorTemplate editorTemplate)` - Checks if the given template for a new editor is allowed and returns a `boolean` value. The `EditorTemplate` argument is the editor template. An `EditorTemplate` is used to create an editor for a given extension. You can thus filter what appears in the list of the **New** dialog box.
  - `isDebuggerperspectiveAllowed()` - Checks if the debugger *perspective* is allowed and returns a `boolean` value.
  - `validateSHMarker(String marker)` - Checks if the given marker is allowed and returns a `boolean` value. The `String` argument represents the syntax highlight marker to be checked. If you decide to filter certain content types, you can also filter the syntax highlight options so that the content type is no longer present in the Preferences options tree.
  - `validateToolbarComposite(String toolbarCompositeTag)` - Checks if the toolbar composite is available. A toolbar composite is a toolbar component such as a drop-down menu.

Tip: The best way to decide what to filter is to observe the values that Oxygen XML Editor passes when these callbacks are called. You have to create an implementation for this interface that lists in the console all values received by each function. Then you can decide on the values to filter and act accordingly.

Custom Protocol Plugin Extension

This type of *plugin* allows you to work with a custom designed protocol for retrieving and storing files.

It provides the following API:

- The interface `URLStreamHandlerPluginExtension` - There is one method that must be implemented:
  - `getURLStreamHandler(String protocol)` - It takes as an argument the name of the protocol and returns a `URLStreamHandler` object, or null if there is no URL handler for the specified protocol.

This type of *plugin* extension can be usually combined with a *Workspace Access plugin extension* that can add a custom toolbar with custom actions for opening documents from a certain source.

As an alternative, two older *plugin* extensions can also be used to add a toolbar action for showing a custom URL chooser:

- With the help of the `URLChooserPluginExtension2` interface, it is possible to create your own dialog box that works with the custom protocol. This interface provides two methods:
  - `chooseURLs(StandalonePluginWorkspace workspaceAccess)` - Returns a `URL[]` object that contains the URLs the user decided to open with the custom protocol. You can invoke your own URL chooser dialog box here and then return the chosen URLs having your own custom protocol. You have access to the workspace of Oxygen XML Editor.
  - `getMenuName()` - Returns a `String` object that is the name of the entry added in the **File** menu.
- With the help of the `URLChooserToolbarExtension` interface, it is possible to provide a toolbar entry that is used for launching the custom URLs chooser from the `URLChooserPluginExtension` implementation. This interface provides two methods:
  - `getToolbarIcon()` - Returns the `javax.swing.Icon` image used on the toolbar.
  - `getToolbarTooltip()` - Returns a `String` that is the tooltip used on the toolbar button.

Lock Handler Plugin Extension
This type of *plugin extension* is used for locking resources from a specific protocol.

It provides the following API:

- The interface `LockHandlerFactoryPluginExtension`.

  You need to implement the following two methods:

  - `LockHandler getLockHandler()`

    Gets the lock handler for the current handled protocol. Might be `null` if not supported.
  - `boolean isLockingSupported(String protocol)`

    Checks if a lock handler can be provided for a specific protocol.

  To use this type of extension in your *plugin*, create an extension of `LockHandlerFactory` type in your `plugin.xml` file and specify the class implementing `LockHandlerFactoryPluginExtension`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plugin SYSTEM "../plugin.dtd">
<plugin name="CustomLockHandler" .............>
 <runtime>
  ........
 </runtime>

 <extension type="LockHandlerFactory"
        class="LockHandlerFactoryPluginExtensionImpl"/>
 ...............

</plugin>
```

Open Redirect Plugin Extension
This type of *plugin* is useful for opening multiple files with only one open action.

For example, when a zip archive or an ODF file or an OOXML file is open in the ***Archive Browser*** *view* a *plugin* of this type can decide to open a file also from the archive in an XML editor panel. This file can be the `document.xml` main file from an OOXML file archive or a specific XML file from a zip archive.

The *plugin* must implement the interface `OpenRedirectExtension`. It only has one callback: `redirect(URL)` that receives the URL of the file opened by the Oxygen XML Editor user. If the *plugin* decides to open also other files it must return an array of information objects (`OpenRedirectInformation[]`) that correspond to these files. Such an information object must contain the URL that is opened in a new editor panel and the content type (for example, `text/xml`). The content type is used for determining the type of editor panel. A `null` content type allows auto-detection of the file type.

Option Page Plugin Extension
This type of *plugin extension* allows you to add custom **Preferences** pages.

The extension must implement the `ro.sync.exml.plugin.option.OptionPagePluginExtension` interface. The provided callbacks allow you to create a custom *Swing* component that will be added to the page and to react to various calls to persistently save the page settings using the `OptionsStorage` API.

All preferences pages that are contributed by a *plugin* are listed in the **Preferences** dialog box in the **Plugins** category. As long as the added preferences page has the same name as its *plugin*, it will be promoted to the first level of the hierarchy within the **Plugins** category.

The `plugin.xml` configuration file can specify one or more such extensions using constructs like this:

```
<extension type="OptionPage" class="my.pack.CustomOptionPagePluginExtension"/>
```

Resource Locking Custom Protocol Plugin Extension
This type of *plugin* allows you to work with a custom designed protocol for retrieving and storing files and it can lock a resource when opening it in Oxygen XML Editor.

This type of *plugin* extends the custom protocol *plugin* type with resource locking support and provides the following API:

- The interface `URLStreamHandlerWithLockPluginExtension` - The *plugin* receives callbacks following the simple protocol for resource locking and unlocking imposed by Oxygen XML Editor.

  There are two additional methods that must be implemented:

  - `getLockHandler()` - Returns a `LockHandler` implementation class with the implementation of the lock specific methods from the *plugin*.
  - `isLockingSupported(String protocol)` - Returns a `boolean` that is `true` if the *plugin* accepts to manage locking for a certain URL protocol scheme (such as `ftp`, `http`, `https`, or `customName`).

Styles Filter Plugin Extension
This type of *plugin* allows you to dynamically modify the CSS styles used to render elements in the **Author** mode.

The *plugin* must extend the *`ro.sync.exml.plugin.author.css.filter.GeneralStylesFilterExtension`* class. This class has a callback on which you can alter the styles for an **Author** mode element.

This extension point is similar with the Styles Filter that you set at the *framework* level. The only difference is that the *plugin* filters styles are used for any opened XML document, regardless of the document type. The changes made by this *plugin* are prioritized over the changes made by the *framework*-level filter.

Note: Alternatively, you can use the *AuthorStylesheet plugin extension*, which does not require any additional Java development and is compatible with Oxygen XML Web Author Component.

Related Information:
Configuring CSS Styles Filter on page 1228

Targeted URL Stream Handler Plugin Extension
This type of *plugin* can be used when it is necessary to impose custom URL stream handlers for specific URLs.

This *plugin* extension can handle the following protocols: `http`, `https`, `ftp` or `sftp`, for which Oxygen XML Editor usually provides specific fixed URL stream handlers. If it is set to handle connections for a specific protocol, this extension will be asked to provide the URL stream handler for each opened connection of a URL having that protocol.

To use this type of *plugin*, you have to implement the `ro.sync.exml.plugin.urlstreamhandler.TargetedURLStreamHandlerPluginExtension` interface, that provides the following methods:

- `boolean canHandleProtocol(String protocol)`

  This method checks if the *plugin* can handle a specific protocol. If this method returns `true` for a specific protocol, the `getURLStreamHandler(URL)` method will be called for each opened connection of a URL having this protocol.
- `URLStreamHandler getURLStreamHandler(URL url)`

This method provides the URL handler for the specified URL and it is called for each opened connection of a URL with a protocol for which the `canHandleProtocol(String)` method returns `true`.

If this method returns `null`, the default Oxygen XML Editor `URLStreamHandler` is used.

To use this type of extension in your *plugin*, create an extension of `TargetedURLHandler` type in your `plugin.xml` file and specify the class that implements `TargetedURLStreamHandlerPluginExtension`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plugin SYSTEM "../plugin.dtd">
<plugin name="CustomTargetedURLStreamHandlerPlugin" .............>
 <runtime>
  ........
 </runtime>

 <extension type="TargetedURLHandler"
          class="CustomTargetedURLStreamHandlerPluginExtension"/>
 ...............

</plugin>
```

This extension can be useful in situations when connections opened from a specific host must be handled in a particular way. For example, the Oxygen XML Editor `HTTP URLStreamHandler` may not be compatible for sending and receiving SOAP using the SUN Web Services implementation. In this case, you can override the stream handler (set by Oxygen XML Editor) to use the default SUN `URLStreamHandler`, since it is more compatible with sending and receiving SOAP requests.

```
public class CustomTargetedURLStreamHandlerPluginExtension
  implements TargetedURLStreamHandlerPluginExtension {

  @Override
  public boolean canHandleProtocol(String protocol) {
    boolean handleProtocol = false;
    if ("http".equals(protocol) || "https".equals(protocol)) {
      // This extension handles both HTTP and HTTPS protocols
      handleProtocol = true;
    }
    return handleProtocol;
  }

  @Override
  public URLStreamHandler getURLStreamHandler(URL url) {
    // This method is called only for the URLs with a protocol
    // where canHandleProtocol(String) method returns true (HTTP and HTTPS)

    URLStreamHandler handler = null;

    String host = url.getHost();
    String protocol = url.getProtocol();
    if ("some_host".equals(host)) {
      // When there are connections opened from some_host, the SUN HTTP(S)
      // handlers are used
      if ("http".equals(protocol)) {
        handler = new sun.net.www.protocol.http.Handler();
      } else {
        handler = new sun.net.www.protocol.https.Handler();
      }
    }
    return handler;
  }
}
```

Workspace Access Plugin Extension

This type of *plugin* allows you to contribute actions to the main menu and toolbars, create custom views, interact with the application workspace, make modifications to opened documents, and add listeners for various events.

Many complex integrations (such as integrations with Content Management Systems) usually requires access to some workspace resources such as toolbars, menus, views, and editors. This type of *plugin* is also useful because it allows you to make modifications to the XML content of an opened editor.

The *plugin* must implement the *`ro.sync.exml.plugin.workspace.WorkspaceAccessPluginExtension`* interface. The callback method `applicationStarted` of this interface allows access to a parameter of the *`ro.sync.exml.workspace.api.standalone.StandalonePluginWorkspace`* type (allows for API access to the application workspace).

The `StandalonePluginWorkspace` interface has three methods that can be called to customize toolbars, menus, and views:

- `addToolbarComponentsCustomizer` - Contributes to or modifies existing toolbars. You can specify additional toolbar IDs in the associated `plugin.xml` descriptor file using the following construct:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plugin SYSTEM "../plugin.dtd">
<plugin name="CustomWorkspaceAccess" .............>
 <runtime>
  ........
 </runtime>

 <extension type="WorkspaceAccess" ...............>
 ...............
  <toolbar id="SampleID" initialSide="NORTH" initialRow="1"/>
</plugin>
```

  The `toolbar` element adds a toolbar in the Oxygen XML Editor interface and allows you to contribute your own *plugin*-specific actions. The following attributes are supported:

  - `id` - Unique identifier for the toolbar.
  - `initialSide` - Specifies the place where the toolbar is initially displayed. The allowed values are `NORTH` and `SOUTH`.
  - `initialRow` - Specifies the initial row on the specified side where the toolbar is displayed. For example, the first toolbar has an initial row of `0` and the next toolbar has an initial row of `1`.

  The *`ro.sync.exml.workspace.api.standalone.ToolbarInfo`* toolbar component information with the specified ID will be provided to you by the customizer interface. Therefore, you will be able to provide Swing components that will appear on the toolbar when the application starts.
- `addViewComponentCustomizer` - Contributes to or modifies existing views, or contributes to the reserved custom view. You can specify additional view IDs in the associated `plugin.xml` descriptor using the following construct:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plugin SYSTEM "../plugin.dtd">
<plugin name="CustomWorkspaceAccess" .............>
 <runtime>
  ........
 </runtime>

 <extension type="WorkspaceAccess" ...............>
 ...............
 <view id="SampleID" initialSide="WEST" initialRow="0"/>
</plugin>
```

  The `view` element adds a view in the Oxygen XML Editor interface and allows you to contribute your own *plugin*-specific UI components. The following attributes are supported:

  - `id` - Unique identifier of the view component.
  - `initialSide` - Specifies the place where the view is initially displayed. The allowed values are: `NORTH`, `SOUTH`, `EAST`, and `WEST`.
  - `initialRow` - Specifies the initial row on the specified side where the view is displayed. For example, in Oxygen XML Editor, the ***Project*** *view* has an initial row of `0` and the ***Outline*** *view* has an initial row of `1`. Both views are in the WEST part of the workbench.
  - `initialState` - Specifies the initial state of the view. The allows values are: `hidden`, `docked`, `autohide`, and `floating`. By default, the view is visible and docked.

  The `ro.sync.exml.workspace.api.standalone.ViewInfo` view component information with the specified ID will be provided to you by the customizer interface. Therefore, you will be able to provide Swing components that will appear on the view when the application starts.
- `addMenuBarCustomizer` - Contributes to or modifies existing menu components.

Access to the opened editors can be done by first getting access to all URLs opened in the workspace using the `StandalonePluginWorkspace.getAllEditorLocations(int editingArea)` API method. There are two available editing areas: the **DITA Maps Manager** editing area and the main editing area. Using the URL of an opened resource, you can gain access to it using the `StandalonePluginWorkspace.getEditorAccess(URL location, int editingArea)` API method. A *`ro.sync.exml.workspace.api.editor.WSEditor`* then allows access to the current editing page.

A special editing API is supported for the **Text** mode (*ro.sync.exml.workspace.api.editor.page.text.WSTextEditorPage*) and the **Author** mode (*ro.sync.exml.workspace.api.editor.page.author.WSAuthorEditorPage*).

To be notified when editors are opened, selected, and closed, you can use the `StandalonePluginWorkspace.addEditorChangeListener` API method to add a listener.

Adding a Custom View in Oxygen XML Editor

To add a custom view in Oxygen XML Editor, follow this procedure:

1. In your `plugins` directory, locate the `plugin.xml` descriptor file. Define the ID of the view you want to add and specify the location where it will be placed:

```
<view id="SampleWorkspaceAccessID" initialSide="WEST" initialRow="0"/>
```

2. In your *Workspace Access Plugin Extension* on page 1342 implementation, where the `applicationStarted` callback is received, add a view component customizer like this:

```
pluginWorkspaceAccess.addViewComponentCustomizer(new ViewComponentCustomizer() {
 public void customizeView(ViewInfo viewInfo) {
   if(
     //The view ID defined in the "plugin.xml"
     "SampleWorkspaceAccessID".equals(viewInfo.getViewID())) {
     cmsMessagesArea = new JTextArea("CMS Session History:");
     viewInfo.setComponent(new JScrollPane(cmsMessagesArea));
     viewInfo.setTitle("CMS Messages");
     viewInfo.setIcon(Icons.getIcon(Icons.CMS_MESSAGES_CUSTOM_VIEW_STRING));
   }
 }
});
```

3. Define the `cmsMessagesArea` as a *static* field (if you can access the messages area from anywhere in your code).

Related Information:

Workspace Access Plugin Extension (JavaScript-Based)
This is a JavaScript-based *plugin* extension that allows you to contribute actions to the main menu and toolbars, create custom views, interact with the application workspace, make modifications to opened documents, and add listeners for various events.

This extension can use the same API as the *Workspace Access plugin extension*, but the implementation is JavaScript-based and uses the bundled *Rhino* library to create and work with Java API from the JavaScript code.

The *plugin* descriptor file (named `plugin.xml`) needs to reference a JavaScript file, as in the following example:

```
<!DOCTYPE plugin PUBLIC "-//Oxygen Plugin" "../plugin.dtd">
<plugin
 id="unique.id.value"
 name="Add Action To DITA Maps Manager popup-menu"
 description="Plugin adds action to DITA Maps Manager contextual menu."
 version="1.0"
 vendor="Syncro Soft"
 class="ro.sync.exml.plugin.Plugin"
 classLoaderType="preferReferencedResources">
 <extension type="WorkspaceAccessJS" href="wsAccess.js"/>
</plugin>
```

In the example above, the JavaScript file `wsAccess.js`, located in the `plugin` folder, will be called. This JavaScript file needs to have two JavaScript methods defined inside. Methods that will be called when the application starts and when it ends:

```
function applicationStarted(pluginWorkspaceAccess) {
..........
}

function applicationClosing(pluginWorkspaceAccess) {
..........
}
```

In regards to the `applicationStarted` callback, besides using the *ro.sync.exml.workspace.api.standalone.StandalonePluginWorkspace* API with the `pluginWorkspaceAccesspluginWorkspaceAccess` parameter, you can also use a globally defined field called `jsDirURL` that points to the folder where the JavaScript file is located.

Below is a much larger example with a JavaScript Workspace Access *plugin* extension implementation that adds a new action in the contextual menu of the ***DITA Maps Manager*** *view*. The action starts the `notepad.exe` application and passes the reference to the currently selected `topicref` to it.

```
function applicationStarted(pluginWorkspaceAccess) {
 Packages.java.lang.System.err.println("Application started "
      + pluginWorkspaceAccess);
 edChangedListener = {
  /*Called when a DITA Map is opened*/
   editorOpened: function (editorLocation) {
   Packages.java.lang.System.err.println("\nrunning " + editorLocation);
  /*Get the opened DITA Map*/
   editor = pluginWorkspaceAccess.getEditorAccess(editorLocation,
   Packages.ro.sync.exml.workspace.api.PluginWorkspace.DITA_MAPS_EDITING_AREA);
   ditaMapPage = editor.getCurrentPage();
  /*Add listener called when right-click is done in the DITA Maps manager*/
   customizerObj = {
   customizePopUpMenu: function (popUp, ditaMapDocumentController) {
   Packages.java.lang.System.err.println("RIGHT CLICK" + popUp);
   tree = ditaMapPage.getDITAMapTreeComponent();
  /*Selected tree path*/
   sel = tree.getSelectionPath();
   if (sel != null) {
   selectedElement = sel.getLastPathComponent();
  /*Reference attribute*/
   href = selectedElement.getAttribute("href");
   if (href != null) {
     try {
  /*Create absolute reference*/
   absoluteRef = new Packages.java.net.URL(selectedElement.getXMLBaseURL(),
           href.getValue());
   Packages.java.lang.System.err.println("Computed absolute reference "
         + absoluteRef);
   mi = new Packages.javax.swing.JMenuItem("Run notepad");
        popUp.add(mi);
        actionPerfObj = {
        actionPerformed: function (e) {
          try {
           Packages.java.lang.Runtime.getRuntime().exec("notepad.exe "
               + pluginWorkspaceAccess.getUtilAccess().locateFile(absoluteRef));
          }
        catch (e1) {
         e1.printStackTrace();
          }
         }
        }
   mi.addActionListener(new JavaAdapter(Packages.java.awt.event.ActionListener,
            actionPerfObj));
       }
       catch (e1) {
        Packages.java.lang.System.err.println(e1);
       }
      }
     }
    }
   }

   ditaMapPage.setPopUpMenuCustomizer(new Packages.ro.sync.exml.workspace.api.
editor.page.ditamap.DITAMapPopupMenuCustomizer(customizerObj));
  }
 }
   edChangedListener = new JavaAdapter(Packages.ro.sync.exml.workspace.api.
listeners.WSEditorChangeListener, edChangedListener);
   pluginWorkspaceAccess.addEditorChangeListener(
   edChangedListener,
   Packages.ro.sync.exml.workspace.api.PluginWorkspace.DITA_MAPS_EDITING_AREA);
}

 function applicationClosing(pluginWorkspaceAccess) {
   Packages.java.lang.System.err.println("Application closing "
        + pluginWorkspaceAccess);
}
```

For more information and some samples, see *GitHub Project with Multiple Workspace Access JavaScript-Based Plugin Samples*.

XML Refactoring Operations Plugin Extension

This type of *plugin* allows you to specify one or more directories where the XML Refactoring operation resources are loaded.

The `RefactoringOperationsProvider` extension can be used to specify the location where custom XML Refactoring operation resources (XQuery Update script or XSLT stylesheet and Operation Descriptor files) are stored. Oxygen XML Editor will scan the specified locations to load the custom operations when the XML Refactoring tool is opened, and allows you to share your custom refactoring operations.

Example: XML Refactoring Operations Plugin Extension

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plugin PUBLIC "-//Oxygen Plugin" "../plugin.dtd">

<plugin
 id="refactoring.operations"
 name="Refactoring operations plugin"
 description="Contains operation descriptors and related scripts"
 version="1.0">
 <extension type="RefactoringOperationsProvider">
  <folder path="customDir/"/>
  <folder path="customDir2"/>
 </extension>
</plugin>
```

Plugin Extensions Designed to Work only in the Text Editing Mode

These *plugin* extensions operate only when editing documents in the **Text** mode. They are mounted automatically by the application on the contextual menu in the **Plugins** submenu.

General Plugin Extension
This type of *plugin* allows you to invoke custom code to interact with the workspace in **Text** mode.

This *plugin* is the most general *plugin* type and provides a limited API:

- The interface `GeneralPluginExtension` - Intended for general-purpose *plugins*, kind of external tools but triggered from the **Plugins** menu. The implementing classes must provide the method `process(GeneralPluginContext)`, which must provide the *plugin* processing. This method takes as a parameter a `GeneralPluginContext` object.
- The class `GeneralPluginContext` - Represents the context in which the general *plugin* extension does its processing. The `getPluginWorkspace()` method allows you access to the workspace of Oxygen XML Editor.

Selection Plugin Extension
This type of *plugin* allows you to manage selections of text.

A **selection** *plugin* can be applied to both XML and non-XML documents. The *plugin* is started by making a selection in the editor, then selecting the corresponding menu item from the **Plugins** submenu in the contextual menu of **Text** mode.

This *plugin* type provides the following API:

- The interface `SelectionPluginExtension` - The context containing the selected text is passed to the extension and the processed result is going to replace the initial selection. The `process(GeneralPluginContext)` method must return a `SelectionPluginResult` object that contains the result of the processing. The `String` value returned by the `SelectionPluginResult` object can include *editor variables* such as *${caret}* and *${selection}*.
- The `SelectionPluginContext` object represents the context. It provides four methods:
  - `getSelection()` - Returns a `String` that is the current selection of text.
  - `getFrame()` - Returns a `Frame` that is the editing frame.
  - `getPluginWorkspace()` - Returns access to the workspace of Oxygen XML Editor.
  - `getDocumentURL()` - Returns the URL of the current edited document.

Related Information:
Editor Variables on page 169

Example - Uppercase Plugin

The following *plugin* is called `UppercasePlugin` and is an example of a *Selection plugin.*. It is used in Oxygen XML Editor for capitalizing the characters in the current selection. This example consists of two Java classes and the *plugin* descriptor file (`plugin.xml`):

- `UppercasePlugin.java`:

```
package ro.sync.sample.plugin.uppercase;

import ro.sync.exml.plugin.Plugin;
import ro.sync.exml.plugin.PluginDescriptor;
```

```
public class UppercasePlugin extends Plugin {
    /**
    * Plugin instance.
    */
    private static UppercasePlugin instance = null;

    /**
    * UppercasePlugin constructor.
    *
    * @param descriptor Plugin descriptor object.
    */
    public UppercasePlugin(PluginDescriptor descriptor) {
        super(descriptor);

        if (instance != null) {
            throw new IllegalStateException("Already instantiated !");
        }
        instance = this;
    }

    /**
    * Get the plugin instance.
    *
    * @return the shared plugin instance.
    */
    public static UppercasePlugin getInstance() {
        return instance;
    }
}
```

- `UppercasePluginExtension.java`:

```
package ro.sync.sample.plugin.uppercase;

import ro.sync.exml.plugin.selection.SelectionPluginContext;
import ro.sync.exml.plugin.selection.SelectionPluginExtension;
import ro.sync.exml.plugin.selection.SelectionPluginResult;
import ro.sync.exml.plugin.selection.SelectionPluginResultImpl;

public class UppercasePluginExtension implements SelectionPluginExtension {
    /**
    * Convert the text to uppercase.
    *
    *@param  context  Selection context.
    *@return          Uppercase plugin result.
    */
    public SelectionPluginResult process(SelectionPluginContext context) {
        return new SelectionPluginResultImpl(
            context.getSelection().toUpperCase());
    }
}
```

- `plugin.xml`:

```
<!DOCTYPE plugin SYSTEM "../plugin.dtd">
<plugin
    name="UpperCase"
    description="Convert the selection to uppercase"
    version="1.0.0"
    vendor="SyncRO"
    class="ro.sync.sample.plugin.uppercase.UppercasePlugin">
    <runtime>
        <library name="lib/uppercase.jar"/>
    </runtime>
    <extension type="selectionProcessor"
     class="ro.sync.sample.plugin.uppercase.UppercasePluginExtension"/>
</plugin>
```

Document Plugin Extension

This type of *plugin* allows you to manage the current document.

The **document** *plugin* type can only be applied to an XML document. It can modify the current document that is received as a callback parameter.

The *plugin* is started by selecting the corresponding menu item from the **Plugins** submenu in the contextual menu of **Text** mode. It provides the following API:

- **Interface `DocumentPluginExtension`** - Receives the context object containing the current document. The `process(GeneralPluginContext)` method can return a `DocumentPluginResult` object containing a new document.
- **`DocumentPluginContext` object** - Represents the context and provides three methods:

- `getDocument()` - Returns a `javax.swing.text.Document` object that represents the current document.
- `getFrame()` - Returns a `java.awt.Frame` object that represents the editing frame.
- `getPluginWorkspace()` - Returns access to the workspace of Oxygen XML Editor.

Oxygen XML Editor Plugin How to...

This section includes information about how to implement complex *plugins*.

How to Write a CMS Integration Plugin

To have a complete integration between Oxygen XML Editor and a CMS, you usually have to write a *plugin* that combines the following two available *plugin* extensions:

- *Workspace Access*
- *Custom protocol*

The usual set of requirements for an integration between Oxygen XML Editor and the CMS are as follows:

- Contribute to the Oxygen XML Editor toolbars and main menu with your custom **Check Out** and **Check In** actions:
  - **Check Out** triggers your custom dialog boxes that allow you to browse the remote CMS and choose the resources you want to open.
  - **Check In** allows you to send the modified content back to the server.

    You can use the **Workspace Access** *plugin extension* (and provided sample Java code) for all these operations.
- When **Check Out** is called, use the Oxygen XML Editor API to open your custom URLs (URLs created using your custom protocol). It is important to implement and use a **Custom Protocol** extension to be notified when the files are opened and saved and to be able to provide the content for the relative references the files may contain to Oxygen XML Editor . Your custom `java.net.URLStreamHandler` implementation checks out the resource content from the server, stores it locally and provides its content. Sample **Check Out** implementation:

```
  /**
   * Sample implementation for the "Check Out" method.
   *
   * @param pluginWorkspaceAccess (Workspace Access plugin).
   * @throws MalformedURLException
   */
    private void checkOut(StandalonePluginWorkspace pluginWorkspaceAccess)
throws MalformedURLException {
    //TODO Show the user a custom dialog box for browsing the CMS
    //TODO after user selected the resource create a URL with a custom protocol
    //which will uniquely map to the resource on the CMS using the URLHandler
    //something like:
    URL customURL = new URL("mycms://host/path/to/file.xml");
    //Ask Oxygen to open the URL
    pluginWorkspaceAccess.open(customURL);
    //Oxygen will then your custom protocol handler to provide the contents for
    //the resource "mycms://host/path/to/file.xml"
    //Your custom protocol handler will check out the file in a temporary
    //directory, for example, and provide the content from it.
    //Oxygen will also pass through your URLHandler if you have any relative
    //references which need to be opened/obtained.
  }
```

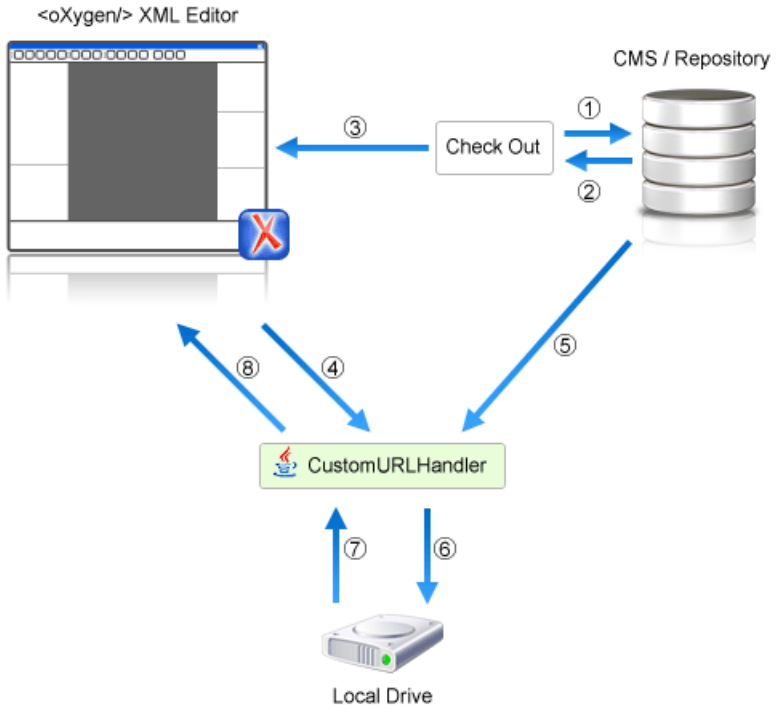


Figure 591: Check Out Process Diagram

Each phase is described below:

1. Browse CMS repository
2. User chooses a resource
3. Use API to open custom URL: `mycms://path/to/file.xml`
4. Get content of URL: `mycms://path/to/file.xml`
5. Get content of resource
6. Store on disk for faster access
7. Retrieve content from disk if already checked out
8. Retrieved content

- Contribute a special **Browse CMS** action to every dialog box in Oxygen XML Editor where a URL can be chosen to perform a special action (such as the **Reuse Content** or **Insert Image** action). Sample code:

```
      //Add an additional browse action to all dialog boxes/places
      //where Oxygen allows selecting a URL.
      pluginWorkspaceAccess.addInputURLChooserCustomizer
(new InputURLChooserCustomizer() {
        public void customizeBrowseActions
(List<Action> existingBrowseActions, final InputURLChooser chooser) {
          //IMPORTANT, you also need to set a custom icon on the action
          //for situations when its text is not used for display.
          Action browseCMS = new AbstractAction("CMS") {
            public void actionPerformed(ActionEvent e) {
              URL chosenResource = browseCMSAndChooseResource();
              if (chosenResource != null) {
                try {
                  //Set the chosen resource in the combo box chooser.
                  chooser.urlChosen(chosenResource);
                } catch (MalformedURLException e1) {
                  //
                }
              }
            }
          };
          existingBrowseActions.add(browseCMS);
        }
      });
```

When inserting references to other resources using the actions already implemented in Oxygen XML Editor, the reference to the resource is made by default relative to the absolute location of the edited XML file. You can gain control over the way in which the reference is made relative for a specific protocol like this:

```
//Add a custom relative reference resolver for your custom protocol.
//Usually when inserting references from one URL to another Oxygen
//makes the inserted path relative.
//If your custom protocol needs special relativization techniques then
//it should set up a custom relative
//references resolver to be notified when resolving needs to be done.
pluginWorkspaceAccess.addRelativeReferencesResolver(
    //Your custom URL protocol for which you already have a
    //custom URLStreamHandlerPluginExtension set up.
    "mycms",
    //The relative references resolver
    new RelativeReferenceResolver() {
  public String makeRelative(URL baseURL, URL childURL) {
    //Return the referenced path as absolute for example.
    //return childURL.toString();
    //Or return null for the default behavior.
    return null;
  }
});
```

- Write the `plugin.xml` descriptor file. Your *plugin* combines the two extensions using a single set of libraries. The descriptor would look like this:

```
<!DOCTYPE plugin SYSTEM "../plugin.dtd">
<plugin
 name="CustomCMSAccess"
 description="Test"
 version="1.0.0"
 vendor="ACME"
 class="custom.cms.CMSAccessPlugin">
 <runtime>
  <library name="lib/cmsaccess.jar"/>
 </runtime>
 <!--Access to add actions to the main menu and toolbars or to add custom views.-->
 <!--See the "CustomWorkspaceAccessPluginExtension" Java sample for more details-->
 <extension type="WorkspaceAccess"
  class="custom.cms.CustomWorkspaceAccessPluginExtension"/>
 <!--The custom URL handler that will communicate with the CMS implementation-->
 <!--See the "CustomProtocolURLHandlerExtension" Java sample for more details-->
 <extension type="URLHandler"
  class="custom.cms.CustomProtocolURLHandlerExtension"/>
</plugin>
```

- Create a `cmsaccess.jar` *JAR* archive containing your implementation classes.
- Copy your new *plugin* directory in the `plugins` subfolder of the Oxygen XML Editor install folder and start Oxygen XML Editor.

Class Loading Issues

It is possible that the Java libraries you have specified in the *plugin* libraries list conflict with the ones already loaded by Oxygen XML Editor. To instruct the *plugin* to prefer its libraries over the ones used by Oxygen XML Editor, you can add the following attribute on the `<plugin>` root element: `classLoaderType="preferReferencedResources"` from the `plugin.xml` descriptor file.

A **Late Delegation Class Loader** (the main class loader in Oxygen XML Editor) is a `java.net.URLClassLoader` extension that prefers to search classes in its own libraries list and only if a class is not found there to delegate to the parent class loader.

The main Oxygen XML Editor Class Loader uses as libraries all *JARS* specified in the *`[OXYGEN_INSTALL_DIR]`*`\lib` directory. Its parent class loader is the default JVM Class loader. For each *plugin* instance, a separate class loader is created having as parent the Oxygen XML Editor Class Loader.

The *plugin* class loader can be either a standard `java.net.URLClassLoader` or a `LateDelegationClassLoader` (depending on the attribute `classLoaderType` in the `plugin.xml`). Its parent class loader is always the Oxygen XML Editor `LateDelegationClassLoader`.

If you experience additional problems, such as:

```
java.lang.LinkageError: ClassCastException:
attempting to cast
jar:file:/C:/jdk1.6.0_06/jre/lib/rt.jar!/
javax/xml/ws/spi/Provider.class
tojar:file:/D:/Program
 Files/Oxygen XML Editor
 12/plugins/wspcaccess/../../xdocs/lib/jaxws/
```

```
jaxws-api.jar!/javax/xml/ws/spi/Provider.class
 at javax.xml.ws.spi.Provider.provider(
Provider.java:94) at
 javax.xml.ws.Service.<init>(Service.java:56)
..............................................
```

The cause could be the fact that some classes are instantiated using the context class loader of the current thread. The most straightforward fix is to write your code in a *try/finally* statement:

```
ClassLoader oldClassLoader =
    Thread.currentThread().getContextClassLoader();
try {
  //This is the implementation of the
  //WorkspaceAccessPluginExtension plugin interface.
  Thread.currentThread().setContextClassLoader(
    CustomWorkspaceAccessPluginExtension.
      this.getClass().getClassLoader());
  //WRITE YOUR CODE HERE
} finally {
  Thread.currentThread().
   setContextClassLoader(oldClassLoader);
}
```

How to Write A Custom Protocol Plugin

To create a custom protocol *plugin*, follow these steps:

1. Write the handler class for your protocol that implements the `java.net.URLStreamHandler` interface.

   Be careful to provide ways to encode and decode the URLs of your files.
2. Write the *plugin* class by extending *`ro.sync.exml.plugin.Plugin`*.
3. Write the *plugin* extension class that implements the *`ro.sync.exml.plugin.urlstreamhandler.URLStreamHandlerPluginExtension`* interface.

   It is necessary that the *plugin* extension for the custom protocol implements the `URLStreamHandlerPluginExtension` interface. Without it, you cannot use your *plugin*, because Oxygen XML Editor is not able to find the protocol handler.

   You can choose also to implement the *`URLChooserPluginExtension`* interface. It allows you to write and display your own customized dialog box for selecting resources that are loaded with the custom protocol.

   An implementation of the extension *`URLHandlerReadOnlyCheckerExtension`* allows you to:

   - Mark a resource as read-only when it is opened.
   - Switch between marking the resource as read-only and read-write while it is edited.

   It is useful when opening and editing CMS resources.
4. Write the `plugin.xml` descriptor file.

   Remember to set the name of the *plugin* class to the one from the second step and the *plugin* extension class name with the one you have chosen at step 3.
5. Create a *JAR* archive with all these files.
6. Install your new *plugin* in the `plugins` subfolder of the Oxygen XML Editor install folder.

Packing and Deploying Plugins or Frameworks as Add-ons

Packing a Plugin or Framework as an Add-on

This procedure is suitable for developers who want a better control over the *add-on* package or those who want to automate some of the steps:

1. Pack the *plugin* or *framework* as a `ZIP` file or a *Java Archive*. Note that you should pack the entire root directory not just its contents.
2. Digitally sign the package. Note that you can perform this step only if you have created a *Java Archive* at the previous step. You will need a certificate signed by a trusted authority. To sign the *JAR* you can either use the `jarsigner` command line tool inside Oracle's Java Development Kit. (`[JDK_DIR]/bin/jarsigner.exe`) or, if you are working with *Apache Ant*, you can use the `signjar` task (a front for the `jarsigner` command line tool).

Note: The benefit of having a signed add-on is that you can verify the integrity of the add-on issuer. If you do not have such a certificate you can generate one yourself using the *keytool* command line tool. This approach is mostly recommended for tests since anyone can create a self signed certificate.

3. Create a descriptor file. You can use a template that Oxygen XML Editor provides. To use this template, go to **File** > **New** and select the **Oxygen add-ons update site** template. Once deployed, this descriptor file is referenced as *update site*.

Alternatively, you can use the **Add-ons Packager** plugin by following this procedure:

1. Install the **Add-ons Packager** plugin from *https://www.oxygenxml.com/InstData/Addons/optional/updateSite.xml* as described in the *Installing Add-ons procedure*.
2. Restart Oxygen XML Editor. If the add-on is correctly installed, the **Add-ons packager** toolbar action is available.
3. Invoke the **Add-ons packager** toolbar action and input the required information in the displayed dialog box.
4. Press **OK** to complete the packaging process.

Deploying an Add-on

To deploy an add-on, copy the `ZIP` or *Java Archive* file and the descriptor file to an HTTP server. The URL to this location serves as the *Update Site URL*.

How to Share a Class Loader Between a Framework and Plugin

In some cases you may need to extend the functionality of Oxygen XML Editor both through a *framework* and through a *plugin*. Normally, a *framework* and a *plugin* both run in their own private classloader. If the *framework* and the *plugin* use the same JAVA extensions/classes, it is recommended that they share the same classloader. This way, the common classes are loaded by only one *Class Loader* and they will both use the same static objects and have the ability to cast objects between one another.

To do this, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Document Type Association**, select the document type, go to the **Classpath** tab, and in the **Use parent classloader from plugin with ID** fields introduce the ID of the *plugin*. This ID is declared in the *configuration file of the plugin*.

Important: The shared classes must be specified only in the configuration files of the *plugin*, and not in the configuration file and the document type class path at the same time.

Creating and Running Automated Tests

If you have developed complex custom *plugin* or *framework* (document types), the best way to test your implementation and ensure that further changes will not interfere with the current behavior is to make automated tests for your customization.

An Oxygen XML Editor standalone installation includes a main `oxygen.jar` library located in the *`[OXYGEN_INSTALL_DIR]`*. That *JAR* library contains a base class for testing developer customizations named: `ro.sync.exml.workspace.api.PluginWorkspaceTCBase`.

To develop `JUnit` tests for your customizations using the **Eclipse** workbench, follow these steps:

1. Create a new Eclipse Java project and copy to it the entire contents of the *`[OXYGEN_INSTALL_DIR]`*.
2. Add all *JAR* libraries present in the *`[OXYGEN_INSTALL_DIR]`*`/lib` directory to the **Java Build Path**->**Libraries** tab. Make sure that the main *JAR* library `oxygen.jar` or `oxygenAuthor.jar` is the first one in the Java classpath by moving it up in the **Order and Export** tab.
3. Click **Add Library** and add the *`JUnit`* and *`JFCUnit`* libraries.
4. Create a new Java class that extends `ro.sync.exml.workspace.api.PluginWorkspaceTCBase`.
5. Pass the following parameters on to the constructor of the super class:
   - `File installationFolder` - The file path to the main application installation directory. If not specified, it defaults to the folder where the test is started.
   - `File frameworksFolder` - The file path to the `frameworks` directory. It can point to a custom *framework* directory where it resides.
   - `File pluginsFolder` - The file path to the `plugins` directory. It can point to a custom *plugin* directory where it resides.

- `File optionsFolder` - The folder that contains the application options. If not specified, the application will auto-detect the location based on the started product ID.
- `String licenseKey` - The license key used to license the test class.
- `int productID` - The ID of the product and should be one of the following: `PluginWorkspaceTCBase.XML_AUTHOR_PRODUCT`, `PluginWorkspaceTCBase.XML_EDITOR_PRODUCT`, or `PluginWorkspaceTCBase.XML_DEVELOPER_PRODUCT`.

6. Create test methods that use the API in the base class to open XML files and perform various actions on them. Your test class could look something like this:

```
public class MyTestClass extends PluginWorkspaceTCBase {

/**
 * Constructor.
 */
public MyTestClass() throws Exception {
 super(null, new File("frameworks"), new File("plugins"), null,
"------START-LICENSE-KEY------\n" +
 "\n" +
 "Registration_Name=Developer\n" +
 "\n" +
 "Company=\n" +
 "\n" +
 "Category=Enterprise\n" +
 "\n" +
 "Component=XML-Editor, XSLT-Debugger, Saxon-SA\n" +
 "\n" +
 "Version=14\n" +
 "\n" +
 "Number_of_Licenses=1\n" +
 "\n" +
 "Date=09-04-2012\n" +
 "\n" +
 "Trial=31\n" +
 "\n" +
 "SGN=MCwCFGNoEGJSeiC3XCYIyalvjzHhGhhqAhRNRDpEu8RIWb8icCJO7HqfVP4++A\\=\\=\n" +
 "\n" +
"-------END-LICENSE-KEY-------",
 PluginWorkspaceTCBase.XML_AUTHOR_PRODUCT);
 }

 /**
   * <p><b>Description:</b> TC for opening a file and using a bold operation</p>
   * <p><b>Bug ID:</b> EXM-20417</p>
   *
   * @author radu_coravu
   *
   * @throws Exception
   */
public void testOpenFileAndBoldEXM_20417() throws Exception {
      WSEditor ed = open(new File
("D:/projects/eXml/test/authorExtensions/dita/sampleSmall.xml").toURL());
     //Move caret
     moveCaretRelativeTo("Context", 1, false);

     //Insert <b>
     invokeAuthorExtensionActionForID("bold");
     assertEquals("<?xml version=\"1.0\" encoding=\"utf-8\"?>\n" +
       "<!DOCTYPE task PUBLIC \"-//OASIS//DTD DITA Task//EN\"
 \"http://docs.oasis-open.org/dita/v1.1/OS/dtd/task.dtd\">\n" +
       "<task id=\"taskId\">\n" +
       "    <title>Task <b>title</b></title>\n" +
       "    <prolog/>\n" +
       "    <taskbody>\n" +
       "        <context>\n" +
       "            <p>Context for the current task</p>\n" +
       "        </context>\n" +
       "        <steps>\n" +
       "            <step>\n" +
       "                <cmd>Task step.</cmd>\n" +
       "            </step>\n" +
       "        </steps>\n" +
       "    </taskbody>\n" +
       "</task>\n" +
       "", getCurrentEditorXMLContent());
   }
}
```

Debugging a Plugin Using the Eclipse Workbench

To debug problems in the code of a *plugin* without having to re-bundle the *plugin's* Java classes in a *JAR* library, follow these steps:

1. *Download* and install Oxygen XML Editor.

2. Set up the *Oxygen SDK* following *this set of instructions*.
3. Create an Eclipse Java Project (for example, `MyPluginProject`) from one of the sample *plugins* (the Workspace Access plugin, for example).
4. In the `MyPluginProject` folder, create a folder called `myPlugin`. In this new folder copy the `plugin.xml` file from the sample *plugin*. Modify the added `plugin.xml` to add a library reference to the directory where Eclipse copies the compiled output. To find out where this directory is located, invoke the contextual menu of the project (in the ***Project** view*), and go to **Build Path** > **Configure Build Path**. Then inspect the value of the **Default output folder** text box.

   Example: If the compiled output folder is `classes`, then the you need to add in the `plugin.xml` the following library reference:

   ```
   <library name="../classes"/>
   ```
5. Copy the `plugin.dtd` from the *`[OXYGEN_INSTALL_DIR]`*`/plugins` folder in the root `MyPluginProject` folder.
6. In the `MyPluginProject` build path add external *JAR* references to all the *JAR* libraries in the *`[OXYGEN_INSTALL_DIR]`*`/lib` folder. Now your `MyPluginProject` should compile successfully.
7. In the Eclipse IDE, create a new *Java Application* configuration for debugging. Set the **Main class** box to `ro.sync.exml.Oxygen`. Click the **Arguments** tab and add the following code snippet in the **VM arguments** input box, making sure that the path to the `plugins` directory is the correct one:

   ```
   -Dcom.oxygenxml.app.descriptor=ro.sync.exml.EditorFrameDescriptor -Xmx1024m
   -XX:MaxPermSize=384m -Dcom.oxygenxml.editor.plugins.dir=D:\projects
   \MyPluginProject
   ```
8. *Add a breakpoint* in the source of one of your Java classes.
9. Debug the created configuration. When the code reaches your *breakpoint*, the Eclipse IDE debugging *perspective* should take over.

Debugging an Oxygen SDK Extension Using the Eclipse Workbench

To debug problems in an *extension* code without having to bundle its Java classes in a *JAR* library, perform the following steps:

1. *Download* and install Oxygen XML Editor.
2. Create an Eclipse Java Project (for example, `MySDKProject`) with the corresponding Java sources (for example, a custom implementation of the `ro.sync.ecss.extensions.api.StylesFilter` interface).
3. In the Project build path add external *JAR* references to all the *JAR* libraries in the *`[OXYGEN_INSTALL_DIR]`*`/lib` folder. Now your Project should compile successfully.
4. Start the standalone version of Oxygen XML Editor from the *`[OXYGEN_INSTALL_DIR]`* and in the ***Document Type Association** preferences page*, edit the document type (for example, **DITA**) to open the ***Document Type** configuration dialog box*. In the **Classpath** tab, add a reference to your Project's `classes` directory and in the **Extensions** tab, select your custom `StylesFilter` extension as a value for the **CSS styles filter** property. Close the application to save your changes.
5. Create a new Java Application configuration for debugging. The Main Class should be `ro.sync.exml.Oxygen`. The given VM Arguments should be

   ```
   -Dcom.oxygenxml.app.descriptor=ro.sync.exml.EditorFrameDescriptor -Xmx1024m -
   XX:MaxPermSize=384m
   ```
6. *Add a breakpoint* in one of the source Java classes.
7. Debug the created configuration. When the code reaches your *breakpoint*, the Eclipse IDE debugging *perspective* should take over.

Disabling a Plugin

To disable a *plugin*, use one of the following two methods:

- *Open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Plugins**, and deselect the *plugin* that you want to disable.

- Create an empty file called `plugin.disable` next to the *plugin* configuration file (`plugin.xml`). The *plugin* will be disabled and will no longer be loaded by the application on startup.

  Note: This is useful if you want to temporarily stop work on a *plugin* and use the application without it.

Oxygen XML Author Component

The Oxygen XML Author Component was designed as a subset of Oxygen XML Editor that can be integrated into another application under the terms of the Oxygen XML Editor SDK agreement to provide functionality for editing and authoring XML documents. The component can be embedded in a third-party standalone Java application or customized as a Java Web Applet to provide WYSIWYG-like XML editing directly in your web browser.

The Oxygen XML Author Component startup project for *Java Swing* integrations is available online as a Maven archetype on the Oxygen XML Editor website. More information about the setup can be found on the *Oxygen SDK page*.

Licensing

The licensing terms and conditions for the Oxygen XML Author Component are defined in the ***Oxygen SDK License Agreement***. To obtain the licensing terms and conditions and other licensing information as well, you can also contact our support team at **support@oxygenxml.com**. You may also obtain a free of charge evaluation license key for development purposes, subject to registration. Any deployment of an application developed using the Oxygen XML Author Component is also subject to the terms of the SDK agreement.

There are two main categories of Oxygen XML Author Component integrations:

1. **Integration for internal use**.

   You develop an application that embeds the *Author Component* to be used internally (in your company or by you). You can buy and use previously purchased Oxygen XML Editor floating licenses to enable the runtime usage of the Oxygen XML Author Component as it was integrated into the application.
2. **Integration for external use**.

   Using the Oxygen XML Author Component, you create an application that you distribute to other users outside your company (with a CMS for example). In this case you need to contact us to apply for a Value Added Reseller (VAR) partnership.

From a technical point of view, the Oxygen XML Author Component provides the Java API to:

- Inject floating license server details in the Java code. The following link provides details about how to configure an HTTP floating license server: *https://www.oxygenxml.com/license_server.html#floating_license_servlet*.

```
AuthorComponentFactory.getInstance().init(frameworkZips,
  optionsZipURL, codeBase, appletID,
      //The servlet URL
      "http://www.host.com/servlet",
      //The HTTP credentials user name
      "userName",
      //The HTTP credentials password
      "password");
```

- Inject the licensing information key (for example, the evaluation license key) directly in the component's Java code.

```
AuthorComponentFactory.getInstance().init(
   frameworkZips, optionsZipURL, codeBase, appletID,
   //The license key if it is a fixed license.
   licenseKey);
```

- Display the license registration dialog box. This is the default behavior if a `null` license key is set using the API, this transfers the licensing responsibility to the end-user. The user can license an Oxygen XML Author Component using standard Oxygen XML Editor license keys. The license key will be saved to the local user's disk and on subsequent runs the user will not be asked anymore.

```
AuthorComponentFactory.getInstance().init(
   frameworkZips, optionsZipURL, codeBase, appletID,
```

```
    //Null license key, will ask the user.
    null);
```

Related Information:

Installation Requirements

Running the Oxygen XML Author Component as a Java applet requires:

- Oracle (Sun) Java JRE version 1.6 update 10 or newer.
- At least 100 MB disk space and 100MB free memory.
- The applet needs to be signed with a valid certificate and will request full access to the user machine to store customization data (such as options and *framework* files).
- A table of supported browsers can be found in *Supported Browsers and Operating Systems* on page 1360.

Running the Oxygen XML Author Component embedded in a third-party Java/Swing application requires:

- Oracle (Sun) Java JRE version 1.6 or newer.
- At least 100 MB disk space and 100MB free memory.

Customization

For a special type of XML, you can create a custom *framework* (which also works in a standalone version of Oxygen XML Editor). Oxygen XML Editor already has *frameworks* for editing DocBook, DITA, TEI, and so on. Their sources are available in *the Oxygen SDK*. This custom *framework* is then packed in a zip archive and used to deploy the component.

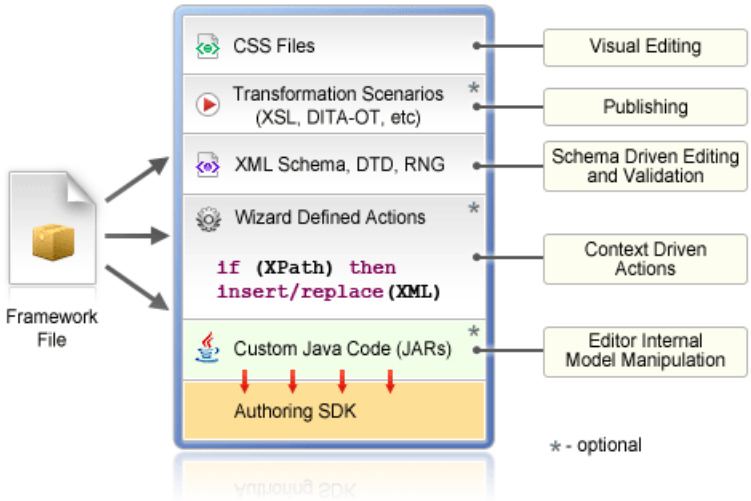


Figure 592: Components of a Custom Framework

Multiple *frameworks* can coexist in the same component and can be used at the same time for editing XML documents.

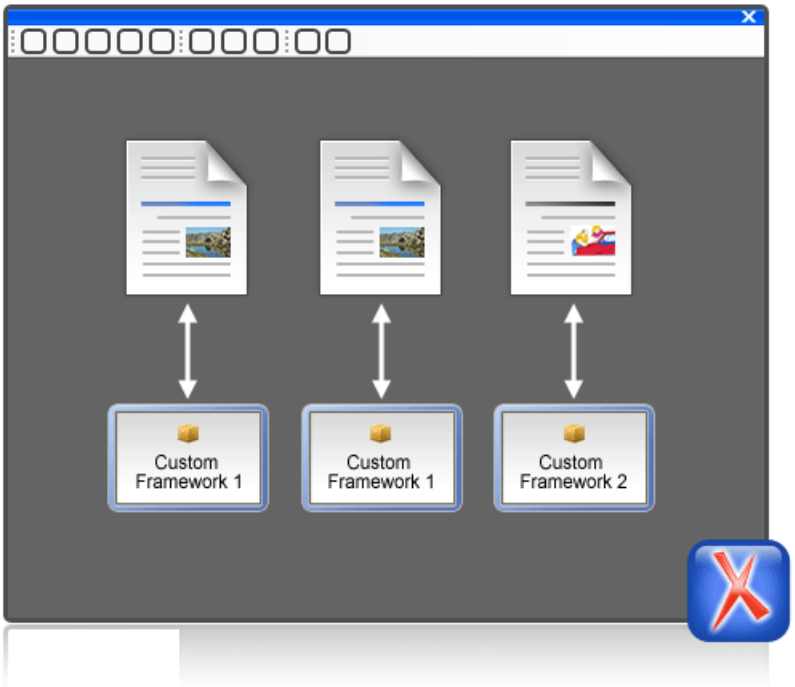


Figure 593: Multiple Frameworks

You can add on your custom toolbar all actions available in the standalone Oxygen XML Editor application for editing in the **Author** mode. You can also add custom actions defined in the *framework* customized for each XML type.

The Oxygen XML Author Component can also provide the ***Outline***, ***Model***, ***Elements***, and ***Attributes*** views, which can be added to your own developed containers.

The main entry point for the Oxygen XML Author Component Java API is the `AuthorComponentFactory` class.

Related Information:

Author Mode Customization Guide on page 1164
Oxygen XML Author Component on page 1355
AuthorComponentFactory API

Example - Customizing the DITA Framework

If you look inside the `bundle-frameworks\oxygen-frameworks` folder distributed with the *Oxygen XML Author Component sample project*, it contains a *framework* folder. Customizations that affect the *framework* configuration for the component should first be done in a standalone installation of Oxygen XML Editor.

An Oxygen XML Editor standalone installation includes a `frameworks` folder that contains the `dita` *framework* located in `[OXYGEN_INSTALL_DIR]\frameworks\dita`. The `dita` *framework* contains a bundled **DITA-OT** distribution which contains the DTDs used for DITA editing. If your DTD specialization is a DITA OT plugin, it *should be installed* in the `DITA-OT-DIR\plugins` folder.

To make changes to the DITA *framework* configuration, *open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Document Type Association**. These changes will affect the `[OXYGEN_INSTALL_DIR]\frameworks\dita\dita.framework` configuration file.

After you do this you can re-pack the Oxygen XML Author Component following the instructions from the README.html file located in the *oxygen-sample-applet* project. The *Author Component* sample project and the Oxygen XML Editor standalone installation should be of the same version.

Related Information:
Advanced Framework Customization on page 1169

Packing a Fixed Set of Options

The Oxygen XML Author Component shares a common internal architecture with the standalone application, although it does not have **Preferences** dialog boxes. However, the *Author Component Applet* can be configured to use a fixed set of user options on startup.

The sample project contains a module called `bundle-options`. The module contains a file called `options.xml` in the `oxygen-options` folder. Such an XML file can be obtained by exporting the options to an XML format from an installation of Oxygen XML Editor.

To create an *options file* in the Oxygen XML Editor:

- Make sure the options that you want to set are not *stored at project level*.
- Set the values you want to impose as defaults in the *Preferences pages*.
- Select **Options** > **Export Global Options**.

Deployment

The Oxygen XML Author Component Java API allows you to use it in your Java application or as a Java applet. The JavaDoc for the API can be found *here*. The sample project found in the `oxygen-sample-applet` module includes Java sources (`ro/sync/ecss/samples/AuthorComponentSample.java`) demonstrating how the component is created, licensed and used in a Java application.

Important: You must obtain the appropriate deployment license (upon payment of a required deployment license fee) to deploy the Oxygen XML Author Component along with your application developed with the SDK.

Web Deployment

The Oxygen XML Author Component can be deployed as a Java Applet using the new Applet with JNLP Java technology, available in Oracle (Sun) Java JRE version 1.6 update 10 or newer.

The *sample project* demonstrates how the Oxygen XML Author Component can be distributed as an applet.

To deploy the Oxygen XML Author Component as a Java Applet, consider the following notes:

- Follow the instructions *here* to setup the sample project and look for Java sources of the sample Applet implementation in the sample project module (**oxygen-sample-applet**). They can be customized to fit your requirements.
- The `default.properties` configuration file must first be edited to specify your custom certificate information used to sign the applet libraries. You also have to specify the code base from where the applet will be downloaded.
- You can look inside the `web-resources/author-component-dita.html` and `web-resources/author-component-dita.js` sample Web resources to see how the applet is embedded in the page and how it can be controlled using JavaScript (to set and get XML content from it).
- The sample Applet `target/jnlp/author-component-dita.jnlp` file contains the list of used libraries. This list is automatically generated from the Maven dependencies of the project.
- The sample *frameworks* and options *JAR* archives can be found in the `bundle-frameworks` and `bundle-options` modules of the sample project.
- Use the Maven command `mvn package` to pack the component. More information are available *here*. The resulting applet distribution is copied in the `target/jnlp/` directory. From this on, you can copy the applet files on your web server.

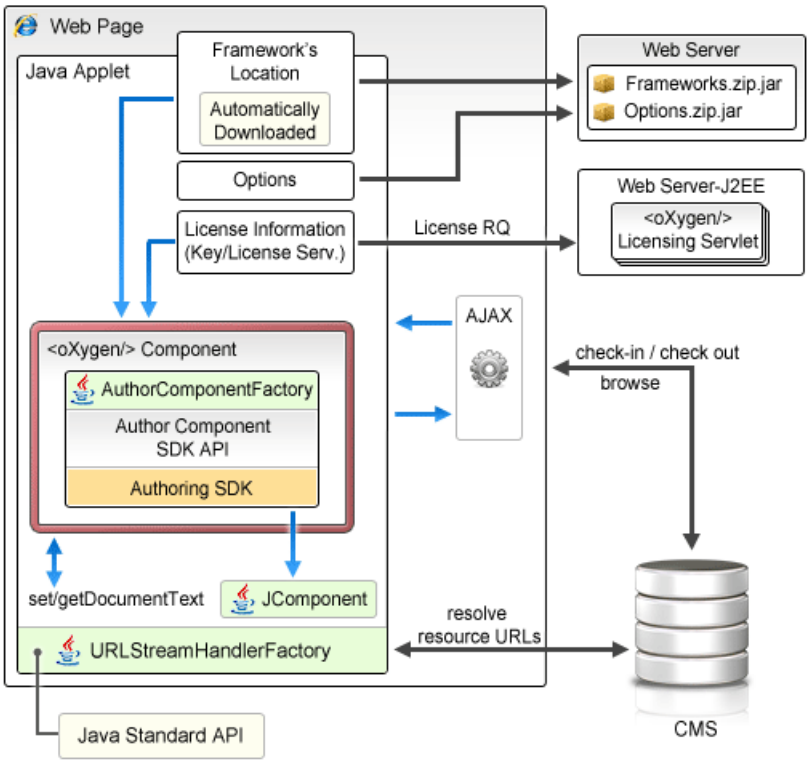


Figure 594: Oxygen XML Author Component Deployed as a Java Applet

Generate a Testing Certificate for Signing an Applet

All *JAR* files of an applet deployed on a remote Web server must be signed with the same certificate before the applet is deployed. The following steps describe how to generate a test certificate for signing the *JAR* files. We will use the tool called **Keytool**, which is included in the Oracle Java Development Kit.

1. Create a *keystore* with an RSA encryption key.

   Invoke the following in a command-line terminal:

   ```
   keytool -genkey -alias myAlias -keystore keystore.pkcs -storetype PKCS12 -
   keyalg RSA -keysize 2048 -dname "cn=your name here, ou=organization unit name,
    o=organization name, c=US"
   ```

   This command creates a *keystore* file called `keystore.pkcs`. The certificate attributes are specified in the `dname` parameter: common name of the certificate, organization unit name (for example, *Purchasing* or *Sales Department*), organization name, country.
2. Generate a self-signed certificate.

   Invoke the following in a command-line terminal:

   ```
   keytool -selfcert -alias myAlias -keystore keystore.pkcs -storetype PKCS12
   ```
3. Optionally display the certificate details in a human readable form.

   First, the certificate must be exported to a separate file with the following command:

   ```
   keytool -export -alias myAlias -keystore keystore.pkcs -storetype PKCS12 -file
    certfile.cer
   ```

   The certificate details are displayed with the command:

   ```
   keytool -printcert -file certfile.cer
   ```

4. Edit the `default.properties` file and fill-in the parameters that hold the path to `keystore.pkcs` file (`keystore` parameter), *keystore* type (`storetype` parameter, with JSK or PKCS12 as possible values), alias (`alias` parameter) and password (`password` parameter).
5. The *JAR* files are automatically signed during the *package* phase of the Maven build.

Supported Browsers and Operating Systems

The applet was tested for compatibility with the following browsers:

	IE 7	IE 8	IE 9	IE 10	IE 11	Edge	Firefox	Safari	Chrome	Opera
Vista	-	Passed	Passed	Passed	Passed		Passed	-	Passed	Passed
Windows 7	-	-	Passed	Passed	Passed		Passed	-	Passed	Passed
Windows 8	-	-	-	Passed	Passed		Passed	-	Passed	Passed
Windows 10	-	-	-	-	-	Passed	Passed	-	Passed	Passed
OS X (10.6 - 10.9)	-	-	-	-	-		Passed	Passed	Failed	Passed
Linux Ubuntu 10	-	-	-	-	-		Passed	-	Failed	Passed

Communication Between the Web Page and Java Applet

Applets can communicate with JavaScript code that runs in the Web Page. JavaScript code can call an applet Java methods and from the Java code you can invoke JavaScript code from the web page.

You are not limited to displaying only *Swing* dialog boxes from the applet. From the operations of an applet, you can invoke a JavaScript API that displays a web page and then obtains the data that has been filled in by the user.

Troubleshooting the Applet

When the applet fails to start:

1. Make sure that your web browser really runs the next generation Java plugin and not the legacy Java plugin.

   For Windows and OS X the procedure is straight forward. Some steps are given below for installing the Java plugin on Linux.

   Manual Installation and Registration of Java Plugin for Linux: *http://www.oracle.com/technetwork/java/javase/manual-plugin-install-linux-136395.html*
2. Refresh the web page.
3. Remove the Java Webstart cache from the local drive and try again.
   - On Windows this folder is located in: `%APPDATA%\LocalLow\Sun\Java\Deployment\cache`.
   - On OS X this folder is located in: `/Users/user_name/Library/Caches/Java/cache`.
   - On Linux this folder is located in: `/home/user/.java/deployment/cache`.
4. Remove the Oxygen XML Author Component cache from the local drive and try again:
   - On Windows Vista/7/8/10 this folder is located in: `%APPDATA%\Roaming\com.oxygenxml.author.component`.
   - On Windows XP this folder is located in: `%APPDATA%\com.oxygenxml.author.component`.
   - On OS X this folder is located in: `/Users/user_name/Library/Preferences/com.oxygenxml.author.component`.
   - On Linux this folder is located in: `/home/user/.com.oxygenxml.author.component`.

5. Problems sometimes occur after upgrading the web browser and/or the JavaTM runtime. Redeploy the applet on the server by running Ant in your Oxygen XML Author Component project. However, doing this does not always fix the problem, which often lies in the web browser and/or in the Java plugin itself.
6. Sometimes when the HTTP connection is slow on first time uses the JVM would simply shut down while the *JARS* were being pushed to the local cache (for example, first time uses). This shut down typically occurs while handling `oxygen.jar`. One of the reasons could be that some browsers (Firefox, for example) implement some form of "Plugin hang detector" See *https://developer.mozilla.org/en/Plugins/Out_of_process_plugins/The_plugin_hang_detector*.
7. If you are running the Applet using Safari on OS X and it has problems writing to disk or fails to start, do the following:
   - In Safari, go to **Safari**->**Preferences**->**Security**.
   - Select **Manage Website Settings**.
   - Then select Java and for the *oxygenxml.com* entry, choose the **Run in Unsafe mode** option.

Enable JavaWebstart logging on your computer to get additional debug information:

1. Open a console and run `javaws -viewer`.
2. In the **Advanced** tab, expand the **Debugging** category and select all boxes.
3. Expand the **Java console** category and choose **Show console**.
4. Save settings.
5. After running the applet, you will find the log files in:
   - On Windows this folder is located in: `%APPDATA%\LocalLow\Sun\Java\Deployment\log`.
   - On OS X this folder is located in: `/Users/user_name/Library/Caches/Java/log`.
   - On Linux this folder is located in: `/home/user/.java/deployment/log`.

Avoiding Resource Caching

A Java plugin installed in a web browser caches access to all HTTP resources that the applet uses. This is useful to avoid downloading all the libraries each time the applet is run. However, this may have undesired side-effects when the applet presents resources loaded via HTTP. If such a resource is modified on the server and the browser window is refreshed, you might end-up with the old content of the resource presented in the applet.

To avoid such a behavior, you need to edit the `ro.sync.ecss.samples.AuthorComponentSampleApplet` class and set a custom `URLStreamHandlerFactory` implementation. A sample usage is already available in the class, but it is commented-out for increased flexibility:

```
//THIS IS HOW YOU CAN REGISTER YOUR OWN PROTOCOL HANDLER TO THE JVM.
//THEN YOU CAN OPEN YOUR CUSTOM URLs IN THE APPLET AND IT WILL USE YOUR HANDLER
URL.setURLStreamHandlerFactory(new URLStreamHandlerFactory() {
 public URLStreamHandler createURLStreamHandler(String protocol) {
  if("http".equals(protocol) || "https".equals(protocol)) {
   return new URLStreamHandler() {
    @Override
    protected URLConnection openConnection(URL u) throws IOException {
     URLConnection connection = new HttpURLConnection(u, null);
     if(!u.toString().endsWith(".jar")) {
      //Do not cache HTTP resources other than JARS
      //By default the Java HTTP connection caches content for
      //all URLs so if one URL is modified and then re-loaded in the
      //applet the applet will show the old content.
      connection.setDefaultUseCaches(false);
     }
     return connection;
    }
   };
  }
  return null;
 }
});
```

Adding MathML support in the Oxygen XML Author Component Web Applet

By default, the Oxygen XML Author Component Web Applet project does not come with the libraries necessary for viewing and editing MathML equations in the **Author** mode. You can view and edit MathML equations either by adding support for *JEuclid* or by adding support for *MathFlow*.

Adding MathML Support Using JEuclid

By default, the `JEuclid` library is excluded from the *Oxygen SDK* artifact dependencies. To enable it, comment the following lines in the `pom.xml` file:

```
<exclusion>
    <artifactId>jeuclid-core</artifactId>
    <groupId>net.sourceforge.jeuclid</groupId>
</exclusion>
```

To edit specialized DITA Composite with MathML content, include the entire MathML2 *framework* directory (`[OXYGEN_INSTALL_DIR]/frameworks/mathml2`) in the *frameworks* bundled with the component in the `bundle-frameworks` module. This directory is used to solve references to MathML DTDs.

Adding MathML support using MathFlow

In the `pom.xml` file add dependencies to the additional libraries used by the `MathFlow` library to parse MathML equations:

1. `MFComposer.jar`
2. `MFExtraSymFonts.jar`
3. `MFSimpleEditor.jar`
4. `MFStructureEditor.jar`
5. `MFStyleEditor.jar`

Note: For MathFlow 2.1, all of these *JAR* files are packaged into one file called `MathFlow.jar`.

You can reference these additional libraries from the `MathFlow` SDK as in the example below:

```
<dependency>
    <groupId>com.dessci</groupId>
    <artifactId>MFComposer</artifactId>
    <version>1.0.0</version>
    <scope>system</scope>
    <systemPath>${MathFlowSDKDir}/lib/MFComposer.jar</systemPath>
</dependency>
```

In addition, you must obtain fixed `MathFlow` license keys for editing and composing *MathML* equations and register them using these API methods: `AuthorComponentFactory.setMathFlowFixedLicenseKeyForEditor` and `AuthorComponentFactory.setMathFlowFixedLicenseKeyForComposer`.

To edit specialized DITA Composite with *MathML* content, include the entire `[OXYGEN_INSTALL_DIR]/frameworks/mathml2` Mathml2 *framework* directory in the *frameworks* bundled with the component in the `bundle-frameworks` module. This directory is used to solve references to *MathML* DTDs.

More documentation is available on the *Design Science MathFlow* website.

Adding Support to Insert References from a WebDAV Connection

Predefined actions that insert references, such as the **Insert Image** action, includes a URL chooser field with a drop-down menu that allows you to select a **Browse Data Source Explorer** action. This action opens the ***Data Source Explorer*** that allows you to view a WebDAV connection.

To use a WebDAV connection in the Oxygen XML Author Component, follow these steps:

1. Open a standalone Oxygen XML Editor 19.0 and *configure a WebDAV connection*.
2. Pack the *fixed set of options* from the standalone to use them with the Oxygen XML Author Component project.
3. In the Oxygen XML Author Component, the defined connection still does not work when expanded because the additional *JAR* libraries used to browse the WebDAV repository are missing. By default, the **httpclient** dependency of the *Oxygen SDK* artifact is excluded. You can enable it by commenting the following lines:

```
<exclusion>
    <artifactId>httpclient</artifactId>
    <groupId>org.apache.httpcomponents</groupId>
</exclusion>
```

If you want to have multiple WebDAV connection URLs, user names, and passwords (depending on the user who started the component), you can use a more flexible approach by using the following API:

```
    //DBConnectionInfo(String id, String driverName, String url, String user,
String passwd, String host, String port)
    DBConnectionInfo info = new DBConnectionInfo("WEBDAV", "WebDAV FTP",
"http://host/webdav-user-root", "userName", "password", null, null);
    AuthorComponentFactory.getInstance().setObjectProperty
("database.stored.sessions1", new DBConnectionInfo[] {info});
```

Using Plugins with the Oxygen XML Author Component

To bundle Workspace Access *plugins* that are developed for standalone application with the Oxygen XML Author Component, follow these steps:

- The `bundle-plugins` module must contain the additional plugin directories in the `dropins` subdirectory. The content must also contain a `plugin.dtd` file. Copy the `plugin.dtd` file from an *`[OXYGEN_INSTALL_DIR]`*`\plugins` folder.
- In the class that instantiates the `AuthorComponentFactory`, for example the `ro.sync.ecss.samples.AuthorComponentSample` class, call the methods `AuthorComponentFactory.getPluginToolbarCustomizers()`, `AuthorComponentFactory.getPluginViewCustomizers()` and `AuthorComponentFactory.getMenubarCustomizers()`, obtain the customizers that have been added by the plugins and call them to obtain the custom swing components that they contribute. There is a commented-out example for this in the `AuthorComponentSample.reconfigureActionsToolbar()` method for adding the toolbar from the **Acrolinx** plugin.

Important: As the Oxygen XML Author Component is just a subset of the entire application, there is no guarantee that all the functionality of the plugin works.

Sample SharePoint Integration of the Oxygen XML Author Component

This section presents the procedure to integrate the Oxygen XML Author Component as a Java applet on a SharePoint site.

Microsoft SharePoint®

Microsoft SharePoint® is a Web application platform developed by Microsoft®.

SharePoint comprises a multipurpose set of Web technologies backed by a common technical infrastructure. It provides the benefit of a central location for storing and collaborating on documents, which can significantly reduce emails and duplicated work in an organization. It is also capable of keeping track of the multiple versions created by multiple users.

Why Integrate the Oxygen XML Author Component with SharePoint

The Oxygen XML Author Component can be embedded in a SharePoint site as a Java applet. This is a simple and convenient way for you to retrieve, open, and save XML and XML related documents stored on your company's SharePoint server, directly from your web browser.

For example, suppose that you are working on a team project that uses the DITA *framework* for writing product documentation and you have the *DITA maps* and topics stored on a SharePoint repository. By using a custom defined action from the contextual menu of a document, you can easily open it in the Oxygen XML Author Component applet that is embedded in your SharePoint Documents page.

You can embed the applet either on a site that is located on a standalone SharePoint server, or on your company Microsoft Office 365 account.

This example can be used as a starting point for other CMS integrations.

Deploying Resources

You can embed the Oxygen XML Author Component in a SharePoint site as a Java Applet, using the new Applet with JNLP Java technology. Sign with a valid certificate the JNLP file and the associated *JAR* files that the applet needs.

Deploy these resources on a third party server (other than the SharePoint server). The Java applet downloads the resources as needed. If you deploy the JNLP and *JAR* files on the SharePoint server, the Java Runtime Environment will not be able to access the applet resources because it is not aware of the current authentication tokens from your browser. This causes the Java Class Loader to fail loading classes, making the applet unable to start.

Accessing Documents

One of the main challenges when integrating the Oxygen XML Author Component applet in your SharePoint site is to avoid authenticating twice when opening a document resource stored in your SharePoint repository.

You have already signed in when you started the SharePoint session, but the applet is not aware of your current session. In this case every time the applet is accessing a document it will ask you to input your credentials again.

As a possible solution, do not execute HTTP requests directly from the Java code, but forward them to the web browser that hosts the applet, because it is aware of the current user session (authentication cookies).

To open documents stored on your SharePoint repository, register your own protocol handler to the JVM. We implemented a handler for both *http* and *https* protocols that forwards the HTTP requests to a JavaScript `XMLHttpRequest` object. This way, the browser that executes the JavaScript code is responsible for handling the authentication to the SharePoint site.

To install this handler, add the following line to your Java Applet code (in our case, in the `ro.sync.ecss.samples.AuthorComponentSampleApplet` class):

```
URL.setURLStreamHandlerFactory(new
  ro.sync.net.protocol.http.handlers.CustomURLStreamHandlerFactory(this));
```

To allow JavaScript calls from your Java applet code, set the MAYSCRIPT attribute to `true` in the `<applet>` element embedded in you HTML page:

```
<applet width="100%" height="600"
  code="ro.sync.ecss.samples.AuthorComponentSampleApplet"
  name="authorComponentAppletName" id="authorComponentApplet"
  MAYSCRIPT="true">
    .....
</applet>
```

Tip: If the applet is not working, or you cannot open documents from your SharePoint repository, enable the debugging tools that come bundled with your Web Browser or the Java Console from your operating system to try to identify the cause of the problem.

Integrating the Oxygen XML Author Component

To integrate the Oxygen XML Author Component as a Java applet with your SharePoint site, you need the Oxygen XML Author Component start-up project.

The project is available as a Maven archetype online. For more information about the setup, go to the *Oxygen XML SDK Website*.

An online demo applet is deployed at *https://www.oxygenxml.com/demo/AuthorDemoApplet/author-component-dita-requirements.html*.

Customize Your Applet

Follow these steps to customize the Oxygen XML Author Component Java applet:

1. Follow *this set of instructions* to setup the sample project and look for the Java sources (these can be customized to fit your requirements) of the sample applet implementation;

   Note: The Java source files are located in the `src` folder of the `oxygen-sample-applet` module.
2. Look inside `web-resources/sharepoint/author-component-dita.aspx` and the associated `*.js` resources, to see how the applet is embedded in the page and how it can be controlled using JavaScript (to set and get XML content from it).
3. Edit the `default.properties` configuration to specify your custom certificate information, used to sign the applet libraries. Also, specify the code base from where the applet resources will be downloaded.
4. The sample Applet `target/jnlp/author-component-dita.jnlp` file contains the list of used libraries. This list is automatically generated from the Maven dependencies of the project. The sample *frameworks* and

options *JAR* archives are located in the `bundle-frameworks` and `bundle-options` modules of the sample project.

Note: The JNLP file and the associated resources and libraries must be deployed on a non-SharePoint web server. Otherwise, the applet will not be loaded.

5. Use the Maven command `mvn package` to pack the component. More information are available *here*. The resulting applet distribution is copied in the `target/jnlp/` directory. From now on, you can copy the applet files on your web server.

Add Resources to Your SharePoint Site

Copy the following resources to a sub-folder (in our example named `author-component`) of the `SitePages` folder from your SharePoint site, where you want to embed the applet:

1. **`author-component-dita.aspx`** - an HTML document containing the Java applet.

   Note: It has an `.aspx` extension instead of `.html`. If you use the latter extension, the browser will download the HTML document instead of displaying it.

   Note: Edit the **`.aspx`** file and change the value of the applet parameter jnlp_href to the URL of the deployed `author-component-dita.jnlp`. Keep in mind that the JNLP file should be deployed on a third party server. For example:

```
<applet>
    <param name="jnlp_href"
        value="http://www.oxygenxml.com/DemoApplet/author-component-dita.jnlp"/>
    ..........
</applet>
```

2. **`author-component-dita.css`** - contains custom styling rules for the HTML document.
3. **`author-component-dita.js`** - contains JavaScript code, giving access to the Oxygen XML Author Component contained by the Java applet.
4. **`connectionUtil.js`** - contains JavaScript utility methods.

   Note: Replace the value of the SPRootSiteURL property with the URL of your SharePoint root site, without trailing `'/'`. This is used by the `openListItemInAuthor(itemUrl)` method, to compute the absolute URL of the list item that is to be opened in the applet.

Copy Resources Using Oxygen XML Editor

You can use Oxygen XML Editor to copy your resources to the SharePoint server:

1. Configure a new connection to your SharePoint site in the ***Data Source Explorer*** *view*.

   Note: To watch our video demonstration about connecting to repository located on a SharePoint server, go to *https://www.oxygenxml.com/demo/SharePoint_Support.html*.
2. Browse your new SharePoint connection site and select the **`SitePages`** folder.
3. Create a folder named `author-component` using the **New Folder** contextual menu action.
4. Upload your resources to this folder using the **Import Files** contextual menu action.

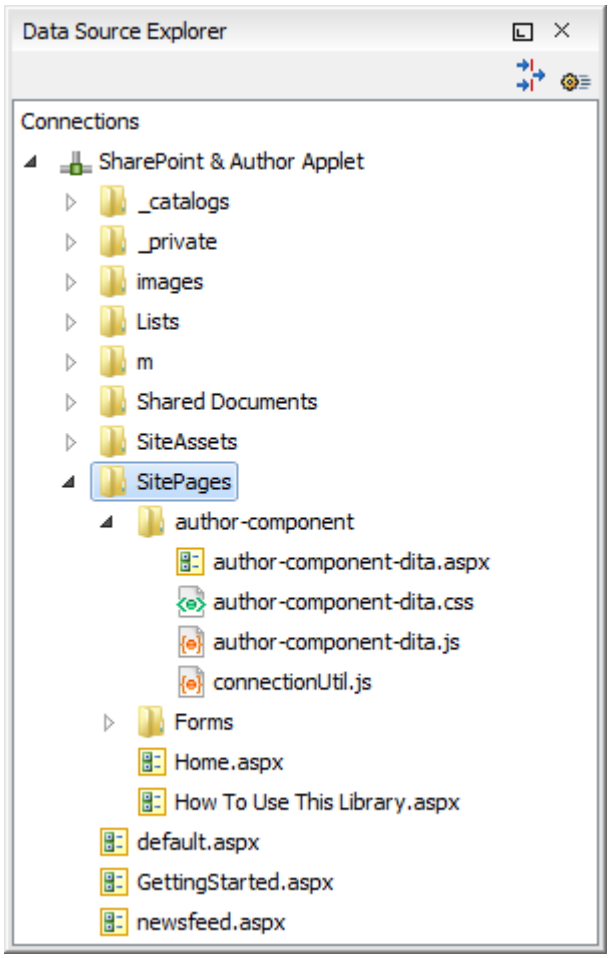


Figure 595: Data Source Explorer View

Embed the Java Applet in Your SharePoint Site

To embed the Java Applet in your SharePoint site, edit the page that contains the applet and add a new Script Editor Web Part next to an existing Documents web part.

Note: It is recommended that you deselect the **Enable Java content in the browser** option from the **Java Control Panel** until you finish editing the page. Otherwise, the browser will load the applet for every change that you will make.

Edit the page directly in your browser, following these steps:

1. Go to the home page of your SharePoint site where you want to add the *Author Component* Java applet.
2. Select the **Page** tab from the ribbon located at top of the page and click the **Edit** button.
3. Select the **Insert** tab and click **Web Part**.
4. In the **Categories** panel, select **Media and Content**.
5. In the **Parts** panel, select the **Script Editor** Web Part.
6. Click the **Add** button to insert the selected Web Part to your page content.
7. Select the newly added Web Part.
8. Select the **Web Part** tab and click the **Web Part Properties** button.
9. Click the **Edit Snippet** link under your Web Part.
10. Insert the following HTML snippet to your newly created Web Part:

```
<div>
    <iframe
       id="appletIFrame"
       src="/applet/SitePages/author-component/author-component-dita.aspx"
       width="800px" height="850px">
    </iframe>
    <script type="text/JavaScript">
        function openInAuthor(itemUrl) {
```

```
            var appletFrame = document.getElementById("appletIFrame");
            var appletWin = appletFrame.contentWindow;
            appletWin.openListItemInAuthor(itemUrl);
        }
    </script>
</div>
```

The above HTML fragment contains an `IFrame` that points to the page where the Java applet resides. Replace the value of the `src` attribute with the path of the `author-component-dita.aspx` HTML page that you added earlier to the `SitePages` folder;

Note: Use the `iframe` element from the HTML fragment with the expanded form (`<iframe></iframe>`). Otherwise, the Web Part will not display the target page of the frame.

11. Save the changes you made to the page.

    Note: Do not forget to select the **Enable Java content in the browser**, to allow the browser to load the Java applet.

Create a SharePoint Custom Action

To open a document from your SharePoint repository in the Oxygen XML Author Component applet, add a new custom action to the contextual menu of your Documents Library:

1. Open your SharePoint site in **Microsoft SharePoint Designer**®.
2. Click **Lists and Libraries** in the **Navigation** pane.
3. Open the **Documents** library.
4. Go to the **Custom Actions** panel.
5. Click the **New** button to add a new custom action.
6. Give a name to the action (for example, `Open In Oxygen XML Author`).
7. In the **Select the type of action** section, select the **Navigate to URL** option and enter the following text:

   ```
   javascript:openInAuthor("{ItemUrl}")
   ```

   Note: This translates to a call to the `openInAuthor(itemUrl)` JavaScript function defined in the HTML fragment that was embedded in the Script Editor Web Part. The `{ItemUrl}` parameter will be expanded to the URL of the list item that the action is invoked on.
8. Click the **OK** button to save the action.

Integration Result

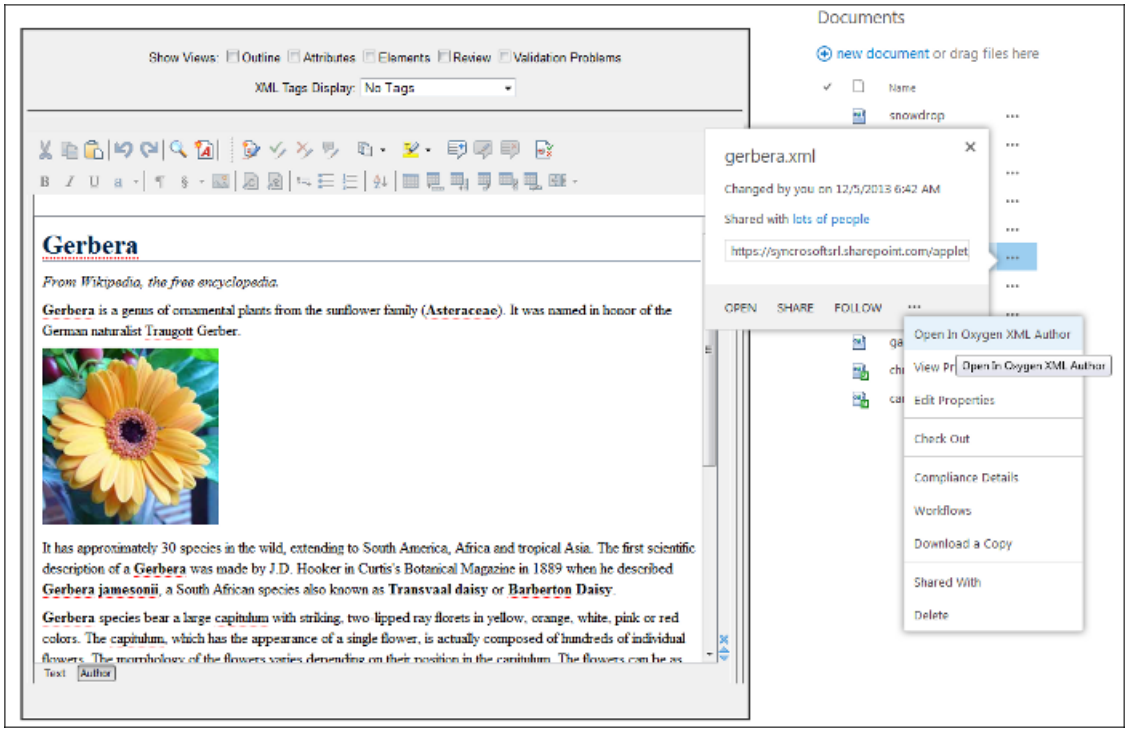


Figure 596: Oxygen XML Author Component Applet Embedded in a SharePoint Site

Frequently Asked Questions

Installation and Licensing

1. What hosting options are available for applet delivery and licensing services (for example, Apache, IIS, etc.)?

   For applet delivery any web server. We currently use Apache to deploy the sample on our site. For the floating license server you would need a J2EE server (such as Tomcat) if you want to restrict the access to the licenses.

   If you do not need the access restrictions that are possible with a J2EE server you can simplify the deployment of the floating license server by using the TCP version of this server. The TCP license server is a simple Java application that communicates with the Oxygen XML Author Component by TCP/IP connections.
2. Are there any client requirements beyond the Java VM and (browser) Java PlugIn Technology?

   Oracle (formerly Sun) Java JRE version 1.6 update 10 or newer. At least 200 MB disk space and 200MB free memory would be necessary for the Oxygen XML Author Component applet.
3. Are there any other client requirements or concerns that could make deployment troublesome (for example, browser security settings, client-side firewalls, and AV engines)?

   The applet is signed and will request access to the user machine to store *framework* customization data. The applet needs to be signed with a valid certificate.
4. How sensitive is the applet for the automatic Java VM updates that are typically on by default (for example, could automatic updates potentially "break" the run-time)?

   The component should work well with newer Java versions but we cannot guarantee this.
5. How and when are "project" related files deployed to the client (for example, applet code, DTD, styling files, customizations, etc.)?

   Framework files are downloaded on the first load of the applet. Subsequent loads will re-use the cached customization files and will be much faster.
6. For on-line demo (`http://www.oxygenxml.com/demo/AuthorDemoApplet/author-component-dita.html`), noted a significant wait during initial startup. Any other mechanisms to enhance startup time?

See the explanation above.

7. Does the Oxygen XML Author Component support multiple documents being open simultaneously? What are the licensing ramifications?

   A single `AuthorComponentFactory` instance can create multiple `EditorComponentProvider` editors that can then be added and managed by the developer who is customizing the component in a Swing `JTabbedPane`. A single license (floating or user-based) is enough for this.

   If you need to run multiple Java Applets or distinct Java processes using the Oxygen XML Author Component, the current floating license model allows for now only two concurrent components from the same computer when using the HTTP floating license server. An additional started component will take an extra license seat.

8. Is there any internet traffic during an editing session (user actively working on the content, on the client side, in the Oxygen XML Author Component)?

   No.

Functionality

1. How and when are saves performed back to the hosting server?

   What you can see on our web site is just an example of the Oxygen XML Author Component (which is a Java Swing component) used in an Applet.

   This applet is just for demonstration purposes. It's source can be at most a starting point for a customization. You should implement, sign and deploy your custom applet implementation.

   The save operation could be implemented either in JavaScript by requesting the XML content from the Applet or in Java directly working with the Oxygen XML Author Component. You would be responsible to send the content back to the CMS.

2. Is there a particular XML document size (or range) when the applet would start to exhibit performance problems?

   The applet has a total amount of used memory specified in the JNLP JavaWebstart configuration file, which can be increased if necessary. By default, it is 156 Mb. It should work comfortably with documents of 1-3 megabytes.

3. What graphic formats can be directly rendered in the Oxygen XML Author Component?

   GIF, JPEG, PNG, BMP and SVG.

4. Can links be embedded to retrieve (from the server) and "play" other types of digital assets, such as audio or video files?

   You could add listeners to intercept clicks and open the clicked links. This would require a good knowledge of the *Oxygen SDK*. The Oxygen XML Author Component can only render static images (no GIF animations).

5. Does the Oxygen XML Author Component provide methods for uploading ancillary files (new graphics, for instance) to the hosting server?

   No.

6. Does the Oxygen XML Author Component provide any type of autosave functionality?

   By default no but you could customize the applet that contains the Oxygen XML Author Component to save its content periodically to a file on disk.

7. Assuming multiple documents can be edited simultaneously, can content be copied, cut and pasted from one Oxygen XML Author Component "instance" to another?

   Yes.

8. Does the Oxygen XML Author Component support pasting content from external sources (such as a web page or a Microsoft Word document and, if so, to what extent?

   If no customizations are available the content is pasted as simple text. We provide customizations for the major *frameworks* (DITA, DocBook, TEI, etc.) that use a conversion XSLT stylesheet to convert HTML content from clipboard to the target XML.

9. Can UTF-8 characters (such as Greeks, mathematical symbols, etc.) be inserted and rendered?

Any UTF-8 character can be inserted and rendered, provided that the font used for editing supports rendering the characters. The font can be changed by developers but not by the users. When using a logical font (by default, *Serif* for the Oxygen XML Author Component), the JVM will know how to map all characters to glyphs. There is no character map available but you could implement one

Customization

1. Please describe, in general terms, the menus, toolbars, contextual menu options, helper panes, and so on, that are available for the Oxygen XML Author Component out-of-the box.

   You can mount on your custom toolbar all actions available in the standalone Oxygen XML Editor application for editing in the **Author** mode. This includes custom actions defined in the *framework* customized for each XML type.

   The Oxygen XML Author Component also can provide the ***Outline***, ***Model***, ***Elements***, and ***Attributes*** views that can be added to your own panels (see sample applet).
2. Please describe, in general terms, the actions, project resources (for example, DTD/Schema for validation purposes, CSS/XSL for styling, etc.) and typical level of effort that would be required to deploy a Oxygen XML Author Component solution for a customer with a proprietary DTD.

   The **Author** mode internal engine uses CSS to render XML.

   For a special type of XML, you can create a custom *framework* (which also works in an Oxygen XML Editor standalone version) that would also contain default schemas and custom actions. A simple *framework* would probably need 2-3 weeks development time. For a complex *framework* with many custom actions it could take a couple of months. Oxygen XML Editor already has *frameworks* for editing (DocBook, DITA, TEI, etc.) Sources for them are available in *the Oxygen SDK*.

   Multiple *frameworks* can co-exist in the same Oxygen XML Editor instance (the desktop standalone version or the applet version) and can be used at the same time for editing XML documents.
3. Many customers desire a very simplistic interface for contributors (with little or no XML expertise) but a more robust XML editing environment for editors (or other users with more advanced XML expertise). How well does the Oxygen XML Author Component support varying degrees of user interface complexity and capability?

   - *Showing/hiding menus, toolbars, helpers, etc.*

     You assemble all the UI parts from the Oxygen XML Author Component. For example, you could provide two applet implementations: one for advanced users and one for content authors.
   - *Forcing behaviors (for example, ensuring change tracking is on and preventing it from being shut down).*

     You could avoid placing the *change tracking* toolbar actions in the custom applet. You could also use API to turn *change tracking* ON when the content has been loaded.
   - *Preventing access to "privileged" editor processes (for example, accept/reject changes).*

     You can remove the *change tracking* actions completely in a custom applet implementation. Including the ones from the contextual menu.
   - *Presenting and/or describing XML constructs (for example, tags) in "plain-English".*

     Using our API, you can customize what the Outline or Breadcrumb presents for each XML tag. You can also customize the in-place content completion list.
   - *Presenting a small subset of the overall XML tag set (rather than the full tag set) for use by contributors (for example, allowing an author to only insert Heading, Para and inline emphasis).*

     The API allows for a content completion filter that also affects the *Elements* view.
4. Does the Oxygen XML Author Component API provide access to the XML document, for manipulation purposes, using common XML syntax (such as DOM, XPath, etc.)?

   Yes, using the Oxygen XML Author Component API.
5. Can custom dialog boxes be developed and launched to collect information in a "form" (with scripting behind to push tag the collection information and embed it in the XML document?

   Yes.
6. Can project resources and customizations be readily shared between the desktop and component versions of your Oxygen XML Author Component product line?

A *framework* developed for the desktop version of the Oxygen XML Editor application can then be bundled with the Oxygen XML Author Component in a custom applet. For example, the demo applet from our web site is DITA-aware using the same *framework* as the Oxygen XML Editor standalone distribution.

A custom version of the applet that includes one or more customized *frameworks* and user options can be built and deployed for non-technical authors by a technical savvy user using a built-in tool of Oxygen XML Editor. All the authors that load the deployed applet from the same server location will share the same *frameworks* and options.

A custom editing solution can deploy one or more *frameworks* that can be used at the same time.

Print Document Within the Oxygen XML Author Component

Question

Can a document be printed within the Oxygen XML Author Component?

Answer

You can use the following API method to either print the document content to the printer or to show the Print Preview dialog box, depending on the `preview` parameter value:

```
AuthorComponentProvider.print(boolean preview)
```

Here is the online Javadoc for this method: *https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/ecss/extensions/api/component/AuthorComponentProvider.html#print(boolean)*

Feature Matrix

The Oxygen XML Author Component was designed to provide the functionality of the standard **Author** mode and can be embedded either in a third-party standalone Java application or customized as a Java Web Applet to provide WYSIWYG-like XML editing directly in your choice of web browsers.

The Oxygen XML Web Author and Oxygen XML Web Author Component are re-implementations of the Oxygen XML Editor **Author** mode user interface, based on JavaScript and HTML5. Its purpose is to enable XML editing and reviewing on your mobile devices and desktops, directly in a web browser environment. Since the interface was thinned down as much as possible, the core XML processing was moved into a Java-enabled server.

Table 47: Feature Matrix Showing Differences Between Web Author, Web Author Component, and Author Component

Feature	Oxygen XML Web Author	Oxygen XML Web Author Component	Oxygen XML Author Component
Intended audience	Reviewers and occasional contributors.	Reviewers and occasional contributors.	Content authors, technical writers.
Mobile device support	Specifically designed for mobile devices.	Specifically designed for mobile devices.	No
Compatibility with the standard version of Oxygen XML Editor	Covers only editing and reviewing features.	Covers only editing and reviewing features.	100%
Text Mode	Yes	Yes	Yes
Grid Mode	No	No	Yes
Client-side setup	Yes, with an *Administration* page.	Yes, with an *Administration* page.	Requires Java to be installed, and Java Applets to be allowed to run.

Feature	Oxygen XML Web Author	Oxygen XML Web Author Component	Oxygen XML Author Component
Server-side setup	Simply requires running an installer.	Requires a servlet container and performing a Maven build.	Requires a web server.
Ability to configure installation bundles	No	Yes	Yes

Oxygen XML Web Author Component Component

The *Oxygen SDK* includes a sample project that you can use to integrate the Oxygen XML Web Author Component.

This section describes the various ways that you can customize the Oxygen XML Web Author Component, and how to deploy it.

Deploying Oxygen XML Web Author Component

Oxygen XML Web Author Component is a web-based editing platform that utilizes the advanced authoring technology of oXygen XML Editor to bring XML editing and reviewing to your mobile devices, as well as your desktop systems. It is supported on Windows, Linux, and Mac OS X platforms and the most popular browsers.

Server Requirements

Even though the requirements are not very strict, you should consider the following metrics when provisioning the server for running the Oxygen XML Web Author Component:

- A processor core can handle 50 to 100 active users.
- Editing an average DITA file consumes about 10MB of RAM. However, the Oxygen XML Web Author Component includes a *configurable caching mechanism* that stores the oldest files to disk when memory resources become low.

Software Requirements

On the server side, the following applications are supported:

- Servlet container:
  - Apache Tomcat 7 or 8
  - WildFly 10.0.0.Final
  - IBM WebSphere Liberty 8.5.5.8
- Java Virtual Machine 1.8 or newer

Oxygen Data Directory and Other Important Deployment Notes

- All Oxygen XML Web Author Component configuration files are stored in a single folder that can be shared amongst multiple servers in a distributed deployment. It can also be reused when you update the server to a new version or when stored on a shared file system to be used by multiple server instances.

  The default location of this folder depends on the distribution, as follows:

  Windows, Linux, and All Platforms Distributions

  OXYGEN_DATA_DIRECTORY = `[OXYGEN_WEBAUTHOR_INSTALL_DIR]/tomcat/work/Catalina/localhost/oxygen-xml-web-author`

  Web Archive Distribution

  OXYGEN_DATA_DIRECTORY = Depends on the servlet container. For example, in Tomcat it is located in `work/Catalina/localhost/oxygen-xml-web-author`.

However, the default location can be overridden by the *`oxygen.data.dir` system property*.

Attention: If the Oxygen XML Web Author Component is started in security mode, you must *set the `oxygen.data.dir` system property*.

Note: WildFly and WebSphere will erase the folder with configuration files upon restart. For these servers, you must *set the `oxygen.data.dir` system property* to a folder that persists across restarts.

- It is recommended that you install the Oxygen XML Web Author Component in its own instance of Tomcat, without sharing it with other applications.
- If you want to reload the application, you have to restart the server.

Related Information:

Setting up an HTTP Floating License Server on page 37

Licensing

The Oxygen XML Web Author Component uses a floating license model, where the license key is stored on a server and individual users consume license seats from a common pool.

How it works

The license key contains the maximum number of users that can simultaneously access the Oxygen XML Web Author Component at any given moment. After a period of inactivity, the license allocated to that user becomes available.

While no personal information is sent to the server, a cookie that identifies the user is auto-generated. Note that the use of two different browsers (for example, Firefox and Chrome) by a single user, will consume two floating licenses. However, using two or more windows or tabs of the same browser, consumes a single floating license.

Licensing

Follow these steps to license a deployment of the Oxygen XML Web Author Component:

1. *Install a floating license server*. You can deploy the HTTP license server in the same Tomcat server, alongside with Oxygen XML Web Author Component.
2. Configure the license server connection.

   Note: Information about your license will be displayed in the **About** section of the Oxygen XML Web Author Component **Dashboard**.

Configuring the License Server Connection

For information on configuring the license server connection with a simple GUI, see *Configuring the License Server Connection* on page 1376.

Bundling a default License Server Configuration in the Oxygen XML Web Author Component

If you need to build the Oxygen XML Web Author Component with a default license configuration bundled inside, use this method.

The connection to the server should be configured in a properties file located in `WEB-INF/license.properties`. This file might look like this:

```
licensing.server.type=http
licensing.server.url=http://example.com:8080/oxygenLicenseServlet/license-servlet
licensing.server.user=<USER>
licensing.server.password=<CHANGE-ME>
```

The following keys are supported:

licensing.server.type

Type of licensing server. Set it to `http`.

licensing.server.url

The URL of the license server.

licensing.server.user

The name of the user with role `user` used to connect to the license server.

licensing.server.password

The password used for the license server.

licensing.server.encryptedPassword

This is an alternative to **licensing.server.password**. The value should be the encrypted password.

To encrypt the password, setup the *SDK project* and run the following command in the `oxygen-sample-webapp` module:

```
mvn exec:java -Dexec.mainClass="com.oxygenxml.webapp.EncryptionTool" -
Dexec.args="encryptionKey passwd"
```

where `passwd` is the password and `encriptionKey` is the encryption key used to encrypt the password.

backup.licensing.server.url

(optional) For some of the licensing packages we offer a backup license key that can be deployed on a second license server in order to provide higher availability in presence of machine and network failures. This key contains the URL of this backup license server.

backup.licensing.server.user

(optional) The name of the user with role `user` used to connect to the backup license server.

backup.licensing.server.password

(optional) The password to use for the backup license server.

backup.licensing.server.encryptedPassword

(optional) This is an alternative to **backup.licensing.server.password**. In order to generate its value, see the instructions for **licensing.server.encryptedPassword**.

Upgrading

Every new version of Oxygen XML Web Author Component comes with a lot of new features, improvements, bug fixes, and security upgrades. Therefore, you should always use the latest version. However, you may want to perform some internal tests before allowing the users to use the new features. For example, you may want to disable some of the new features to keep the UI simple or to tweak some of the settings.

It is recommended that you make a separate deployment of the new version on a different VM, perform some acceptance tests, and then switch all of your users to the new deployment.

To upgrade Oxygen XML Web Author Component, follow these steps:

1. Deploy the new version of Oxygen XML Web Author Component on a separate virtual machine and connect it to your existing license server. The two deployments will work in parallel, both using the same license pool.
2. It might be necessary to update your license key. If you receive an error message in regards to the license key version, follow *this procedure*. Your existing Oxygen XML Web Author Component deployment will continue to work with a license for the newer version.
3. Tweak the settings for the new version to suit your needs.
4. Switch all of the users to the new version. This can be achieved either by changing the DNS entry or configuring the *Load Balancer* to use the new deployment.

Administration Page

Oxygen XML Web Author Component includes a user-friendly **Administration Page** that helps you to configure your instance of the Oxygen XML Web Author Component. You can use this page to configure a variety of settings. You need to enable the **Administration Page** before you can access it, but once you do, you can access it from a link on the top-right corner of the **Dashboard** page.

Enabling the Administration Page

If you used the ***Linux***, ***Windows***, *or* ***All Platforms*** *installation kits*, the administration page is already enabled.

If you used the **Web Application Archive** version, you need to manually enable the **Administration Page** by following these steps:

1. Edit the `shiro-users.ini` file located in the *Oxygen Data Directory*.
   a. To define a user, use this format:
   ```
   user.USERNAME = PASSWORD,ROLE
   ```
   b. To define a role, use this format:
   ```
   role.ROLENAME = PERMISSIONS
   ```
2. Your Administration Page is now enabled. To access it, go to the following URL:
   ```
   http://example.com:8080/oxygen-webapp/app/admin.html
   ```
   where `http://example.com:8080/oxygen-webapp` is the URL of your instance of the Oxygen XML Web Author Component.
3. You will be prompted for authentication credentials and you will enter those configured in the steps above.

For information about resetting the admin credentials, see *Resetting Admin Credentials*.

Accessing the Administration Page

You can easily access the **Administration** page from a link on the **Dashboard** page.

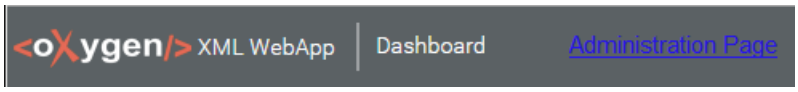


Figure 597: Administration Page Link

How to Hide the Administration Page Link

You can hide the **Administration Page** link from regular users by deselecting the **Show a link to the Administration Page** option that is available in the **Settings** section of the **Administration Page**.

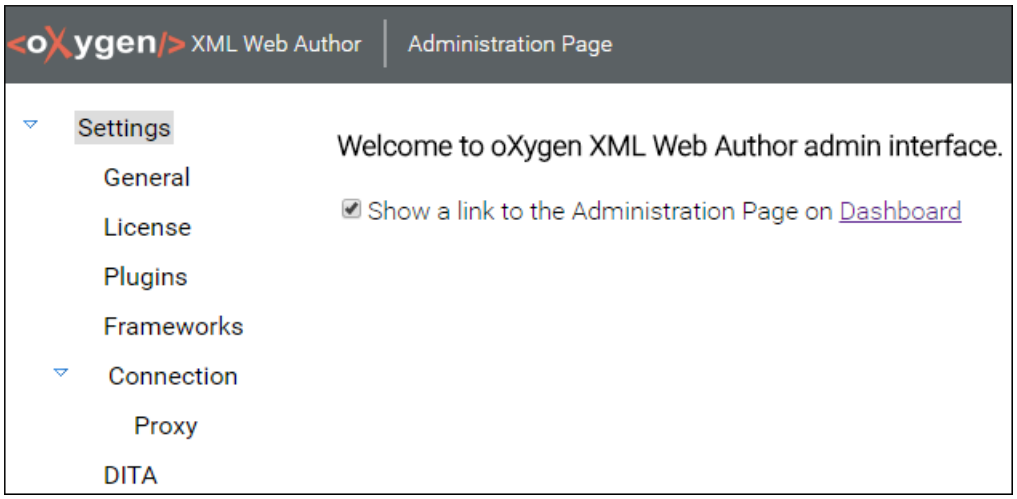


Figure 598: Show Administration Page Link Option

Administration Page Settings

You can click on any of the listed types of settings to access configurable options for each type. The **Administration Page** allows you to configure or view the following settings in the various links:

General

Change tracking initial state

Allows you the choose the initial state of the *Change Tracking feature*. You can choose between **Stored in document**, **Always On**, and **Always Off**.

Show all possible elements in content completion list

When enabled, the *Content Completion Assistant* includes all possible elements, including those that are invalid at the current location, but those proposals are rendered in a lighter shade of gray, italicized, and appear after the valid proposals.

Logging

This section displays the location of the **Log file** or **Config file** that Oxygen XML Web Author Component uses for logging purposes.

Options

This section displays the location of the **Options file** that Oxygen XML Web Author Component uses for various default settings.

License

Displays licensing information and allows you to *configure a license server connection* with **Change Server** and **Manage Server** buttons.

Plugins

Displays the various *plugins* for your Oxygen XML Web Author Component and allows you to *add and configure them*. It also includes an **Upload plugin** button for adding new ones to the list.

Frameworks

Displays the various *frameworks* for your Oxygen XML Web Author Component and allows you to *add and configure them*. It also includes an **Upload framework** button for adding new ones to the list.

Connection

Automatically accept a security certificate, even if invalid

If enabled, security certificates are automatically accepted, regardless of their validity.

Connection > Proxy

Allows you to *configure the proxy settings* for your Oxygen XML Web Author Component.

DITA

Allows you to specify the default directory of the DITA Open Toolkit distribution (bundled with the Oxygen XML Web Author Component installation) to be used for validating and publishing DITA content. You can select from the following:

Built-in DITA-OT 1.8

If this is set, all defined DITA transformation scenarios will run with DITA-OT 1.8.

Built-in DITA-OT 2.x (with support for DITA 1.3 and Lightweight DITA)

All defined DITA transformation scenarios will run with DITA-OT 2.x. This also gives you access to DITA 1.3 and Lightweight DITA file templates when you create new documents.

Configuring the License Server Connection

To configure the license server connection for your Oxygen XML Web Author Component, follow these steps:

Note: This method only applies to *the Oxygen HTTP Floating License Server*.

1. Go to your *Administration Page*.
2. Select **License**.
3. Enter the server URL and your authentication credentials for the license server.
4. Click **Submit**.

You should see a message that informs you that the license configuration was successfully applied. After it is configured, you can change or manage your server connection by using the **Change Server** or **Manage Server** buttons.

Adding and Configuring Plugins

Oxygen XML Web Author Component includes a user-friendly **Administration Page** that helps you to manage *plugins* for the Oxygen XML Web Author Component.

Add a New Plugin

To add a new *plugin* to the Oxygen XML Web Author Component, follow these steps:

1. Go to your *Administration Page*.
2. Select **Plugins**.
3. Click **Upload Plugin** and choose a *plugin* archive to upload.

   Important: Oxygen XML Web Author Component does a validation check to make sure the uploaded plugin archive contains the proper extension descriptor file (`plugin.xml`). The archive should contain exactly one folder. Otherwise you will receive an error that the upload was rejected.

   Note: If the upload would result in a duplicate plugin, Oxygen XML Web Author Component will load the latest version and report the older versions that were not loaded. You will also have the option to delete the older versions or you can ignore this and then delete the newly uploaded version to revert to an older version.
4. Click **OK** to upload the file.

   Result: The *plugin* should appear in the list on this **Plugins** page. Uploaded plugins appear with a light green highlight so that you can identify them easily.
5. Once you are finished with all of your changes, restart the server.

Enable or Disable a Plugin

To enable or disable a *plugin*, follow these steps:

1. Go to your *Administration Page*.
2. Select **Plugins**.
3. Click the **Enable/Enabled** check box on the right side of the particular plugin to toggle it to the desired state.
4. Once you are finished with all of your changes, restart the server.

Configure a Plugin

If the *plugin* can be configured, a **Configure** link is displayed under its name in the *Administration Page*. To configure the *plugin*, click the link and follow the instructions.

Delete a Plugin

To delete a *plugin* from your Oxygen XML Web Author Component, follow these steps:

1. Go to your *Administration Page*.
2. Select **Plugins**.
3. Find the *plugin* you want to delete and click the **Delete the plugin** button on the right side of its name.
4. Restart the server to apply the changes.

Creating a Plugin Configuration Page

To create a configuration page for a *plugin* that does not already have the **Configure** icon displayed in the **Administration** page, follow these steps:

1. Register a `WebappServlet` extension type with a `role` attribute set to `config` in your `plugin.xml` file, as in the following example:

   ```
   <extension type="WebappServlet" role="config" class="com.ex.MyPluginConfigExt"/>
   ```

   The `class` attribute value should point to an extension of the *`PluginConfigExtension`* class.
2. When extending the *`PluginConfigExtension`* class, consider the following notes:

- Implement the `getOptionsForm` method to return an HTML form and make sure that contains inputs with the `name` attribute the same as the option you want to configure.
- Implement the `getOptionsJson` method to return a JSON string that contains the options that you want to make available to the JavaScript code. They will be accessible in your *plugin's* JavaScript code using the `sync.options.PluginsOptions.getClientOption(optionName)` method.

  Note: The JSON should only contain key values, where values are of the type `string | number | boolean` with no arrays or other objects.

  Tip: You should not include any sensitive information (e.g. OAuth secrets) in the options returned by this method.
- Implement the `getPath` method to return a non-empty string that represents the path for which this extension will be served.

  For example:

  ```
  {webapp-context}/plugins-dispatcher/RESULT_OF_GETPATH
  ```
- If you need to override the `init` method, make sure you call `super.init()`. Otherwise, options will not be saved to disk and will be lost when you restart the application.
- If you need to override any of the `doPut/doDelete` methods, make sure you call the `saveOptions` method at the end to save the options to disk.
- If you need to override the `doGet` method, make sure it responds with the result of `getOptionsForm` for header `Accept=text/html`, and with the result of `getOptionsJson` when called with header `Accept=application/json`. Use the `getOption` or `getDefaultOptions` methods to access the current or default options.

Tip: For an implementation example, you can look at `com.oxygenxml.sdksamples.github.GithubPluginConfigExtension` in the *`webapp-github-plugin` project*.

Result: A **Configure** icon is now displayed next to its name in the **Administration** page.

Related Information:
Adding and Configuring Plugins on page 1377

Adding or Removing a Framework

Add a New Framework

To add a new *framework* to your Oxygen XML Web Author Component, follow these steps:

1. Go to your *Administration Page*.
2. Select **Frameworks**.
3. Click **Upload Framework** and choose a *framework* archive to upload.

   Important: Oxygen XML Web Author Component does a validation check to make sure the uploaded framework archive contains the proper extension descriptor file (`*.framework`). The archive should contain exactly one folder. Otherwise you will receive an error that the upload was rejected.
4. Click **OK** to upload the file.

   Result: The *framework* should appear in the list on this **Frameworks** page. Uploaded frameworks appear with a light green highlight so that you can identify them easily.
5. Once you are finished with all of your changes, restart the server.

Delete a Framework

To delete a *framework* from your Oxygen XML Web Author Component, follow these steps:

1. Go to your *Administration Page*.
2. Select **Frameworks**.

3. Find the *framework* you want to delete and click the **Schedule for deletion** button on the right side of its name.
4. Restart the server to apply the changes.

Configuring Proxy Settings

Configure Proxy Settings for Oxygen XML Web Author Component

To configure the proxy setting for your Oxygen XML Web Author Component, follow these steps:

1. Go to your *Administration Page*.
2. Select **Connection** > **Proxy**.
3. Specify how HTTP(S) connections go through the proxy server. You can choose between the following:
   - **Direct connection** - HTTP(S) connections will go directly to the target host without going through a proxy server.
   - **Use system settings** (default setting) - HTTP(S) connections will go through the proxy server set in the operating system.
   - **Manual proxy configuration** - HTTP(S) connections will go through the proxy server specified in the **Web Proxy (HTTP/HTTPS)** section.

     Depending on the option you choose, you can configure the following additional options:

     Web Proxy (HTTP/HTTPS) section

     Address

     The address of the proxy server used for manual configurations.

     Port

     The port of the proxy server used for manual configurations.

     No proxy for

     Specifies the hosts that the connections must not go through a proxy server. A host needs to be written as a fully qualified domain name (for example, `myhost.example.com`) or as a domain name (for example, `example.com`). Use a comma to separate multiple hosts.

     User

     The user name for authentication with the proxy server.

     Password

     The password for authentication with the proxy server.

     SOCKS Proxy section

     Address

     The address of a SOCKS proxy that all connections will pass through. If this field is empty, the connections do not use a SOCKS proxy.

     Port

     The port of a SOCKS proxy that all connections will pass through.
4. Click **Apply** to accept your changes.

Resetting Admin Credentials

To reset the admin user credentials that are used to access the *Administration Page* in Oxygen XML Web Author Component, follow these steps:

1. Delete the `shiro-users.ini` file located in the *Oxygen Data Directory*.
2. Restart the server.
3. In your browser, go to the Oxygen XML Web Author Component **Dashboard** page (for example, **http://example.com:8080/oxygen-webapp/app/oxygen.html** or the **Admininstration Page** (for example, **http://example.com:8080/oxygen-webapp/app/admin.html**) and you will be redirected to a configuration page.
4. Enter your new credentials.

Setting up a Load-Balanced Server

To scale a deployment to a larger number of users and to increase the availability, Oxygen XML Web Author Component can be deployed on a set of *load-balanced* servers. This topic describes the required setup for such a scenario.

Configure Session *Stickyness*

Every Oxygen XML Web Author Component server keeps the state of the edited documents in memory for performance reasons. This implies that every user should connect to the same back-end server for the duration of an editing session. To achieve this result in a *load-balanced* setting, you should enable **session *stickyness***.

Configure Server Health Checks

To detect unhealthy servers, Oxygen XML Web Author Component offers a health check `REST API`. The endpoint has the following interface:

URL

`rest-public/status`

Response status code

200 for a healthy server and 503 for an unhealthy server.

Response body

For an unhealthy server, the response body contains a text description of the problem.

Configure the License Server

All Oxygen XML Web Author Component servers can use the same pool of floating licenses. To this end, they all need to be *configured to use the same license server*.

Share Options Between All Instances

The configuration options are stored in the file system in a directory that can be changed using the *`oxygen.data.dir` system property*. In order for all instances to use the same options, `oxygen.data.dir` should point to the same directory on a shared file system.

Oxygen XML Web Author Component Customization Overview

The core of Oxygen XML Editor can be deployed on a server, allowing a variety of HTML5-enabled client devices to edit and review XML content.

Oxygen XML Web Author Component is a highly versatile application that can be customized to work with any XML vocabulary and any file repository system. It uses the same extension points as the standalone version of Oxygen XML Editor (*frameworks*, options, and *plugins*).

Note: The *frameworks*, options and *plugins* used by the Oxygen XML Web Author Component are similar with those used in the Oxygen XML Editor product suite. This means that you can use them with a high degree of compatibility across Oxygen products.

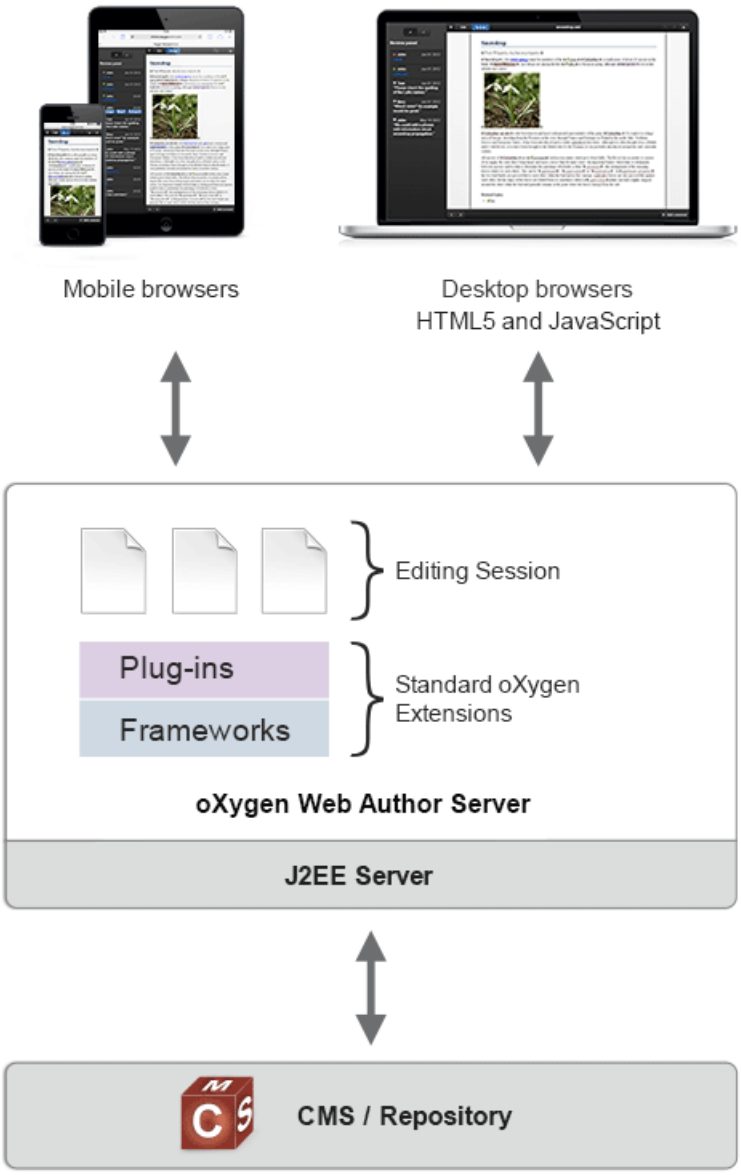


Figure 599: Graphical Description of the Oxygen XML Web Author Component System

Customizing Options

The Oxygen XML Web Author Component functionality that is common with the standalone distribution of Oxygen XML Editor share the same options. This allows you to configure a consistent editing experience for all users.

Author Mode Options

Oxygen XML Web Author Component stores its options in an `options.xml` file.

If you are using the *Oxygen XML SDK project*, the file is located in the `bundle-options/oxygen-options/` folder and will be bundled with the web application. If you are using one of the installation kits, it is located in the `options` folder of the *Oxygen Data Directory*.

There are multiple ways to configure these options:

- Some of the options can be changed using the *Administration Page* in your web browser.

- Use an options file exported from Oxygen XML Editor standalone application. To export the file, use the ***Options*** > ***Export Global Options*** *menu action*.

  Note: Archived options are unpackaged to the `options` folder only if an `options.xml` file is not already present.
- Manually edit the options file. To learn more about the supported options and the file format, see *Oxygen XML Author Options Supported by Web Author* on page 1395.

Oxygen XML Web Author Component specific options

A small number of options are specific only to the Oxygen XML Web Author Component and they can be configured in the `WEB-INF/web.xml` file. Each option is specified as a *context-param* element.

The following is a list of options and their accepted values:

Option name	Value	Default Value	Description
com.oxygenxml.loadBuiltinProtocolHandlers	*true/false*	True	Controls whether or not the built-in handlers for HTTP/HTTPS and FTP/SFTP protocols are installed. Default value is *true*.
com.oxygenxml.webapp.datastore.docs.memory.size	An integer number	10,000	Indicates the number of editing sessions stored in memory.
com.oxygenxml.webapp.datastore.docs.memory.expire	Duration (*)	2d	Indicates the delay after which inactive sessions are stored on disk.
com.oxygenxml.webapp.datastore.docs.disk.size	An integer number	1,000,000	Indicates the number of inactive editing sessions that can be stored on disk.
com.oxygenxml.webapp.datastore.docs.disk.expire	Duration (*)	30d	Indicates the delay after which inactive sessions are discarded.
com.oxygenxml.validation.threads.no	An integer number	Half the number of cores on the server	Configures the number of validation threads.

(*) - Duration is represented by an integer, followed by one of "d", "h", "m", or "s", representing days, hours, minutes, or seconds, respectively.

Example:

Here is an example of how to configure a context parameter:

```
<context-param>
  <param-name>com.oxygenxml.loadBuiltinProtocolHandlers</param-name>
  <param-value>false</param-value>
</context-param>
```

Related Information:

How to Set a System Property on page 1382

How to Set a System Property

A variety of Java system properties can be set to influence the behavior of Oxygen XML Web Author Component. For example, you can change the location of the *Oxygen Data Directory* (`oxygen.data.dir`).

To set a system property, follow the procedure below for the type of installer you used:

Windows Installer

To set a system property for a Windows installation, follow these steps:

1. Go to the installation directory of Oxygen XML Web Author Component.
2. Launch **Manage Web Author**.
3. Go to the **Java** tab.
4. In the **Java Options** section, add the system property using the following pattern: `-D[option_name]=[option_value]`
5. Restart the application.

Linux Installer or All Platforms Kit

To set a system property for a for a Linux or All Platforms installation, follow these steps:

1. Go to the installation directory of Oxygen XML Web Author Component.
2. Edit the `oXygenXmlWebAuthor.vmoptions` file.
3. Add the system property on a new line, using the following pattern: `-D[option_name]=[option_value]`
4. Restart the application.

Customizing Oxygen XML Web Author Component Frameworks

Custom *frameworks* that are designed for documentation purposes can be reused interchangeably between the Oxygen XML Editor standalone distribution and the Oxygen XML Web Author Component. However, some fine-tuning might be necessary to maximize the editing experience for your content authors. The advantages of using a common *framework* include:

- Easier development and testing, since you can test most of the functionality in the standalone version of Oxygen XML Editor using advanced tools such as the ***CSS Inspector***, ***CSS Editor***, or the ***Document Type Association*** *customization dialog box*.
- Uniform experience across multiple Oxygen XML Editor distributions.
- Ability to reuse previously developed *frameworks*.
- Many of the customized items that are added to your *framework* in the Oxygen XML Editor standalone distribution also carry over to Oxygen XML Web Author Component. For example, items that are added to the list of proposals for the *Content Completion Assistant* will appear in both distributions.

Developing and Testing a Framework Using the Oxygen XML Web Author Test Server Add-on

The following procedures assumes that you have access to an Oxygen XML Editor standalone installation. This is not a mandatory requirement, but rather a way to speed up the development process.

1. Use the standalone installation of Oxygen XML Editor to customize a specific *framework* for whatever type of documentation that you require. Modifications made to the *framework* are instantly visible in the standalone version of Oxygen XML Editor, but if you want to preview them in the Oxygen XML Web Author Component, proceed to the next step.
2. *Run the Oxygen XML Web Author Component Test Serve Add-on and test the framework*.

   Note: The changes that you make to your *framework* will not automatically be reflected in the Oxygen XML Web Author Component if it was already running. To see the results of changes, close the server using the **Close and stop server** button and start it again.

Deploying a Framework

1. Copy your customized *framework* into the `bundle-frameworks/oxygen-frameworks/` folder of the *Oxygen XML SDK project*.
2. *Build the SDK project and deploy it*.

Customization Tips

- If you want to use CSS rules that only apply when the *framework* is used in the Oxygen XML Web Author Component, use the following `media` query:

```
@media oxygen AND (platform:webapp) {
 ...
}
```

- In the web folder of each *framework*, you can add a `framework.js` file that calls the *JavaScript API* to implement custom editing actions. The possible use cases include the following:
  - Create custom actions and add them to the toolbar or contextual menu. For more details, see the *JS custom action tutorial*.
  - Create custom form controls. For more details, see the *JS form control tutorial*.
  - Add more views. For more details, see the *JS custom view tutorial*.
- If the *framework* contains **Author** mode operations (Java implementations of the `ro.sync.ecss.extensions.api.AuthorOperation` interface), they can be enabled to be used by the Oxygen XML Web Author Component using the `ro.sync.ecss.extensions.api.WebappCompatible` annotation.

  Note: **Author** mode operations that use *Java Swing* components to display a graphical interface are not compatible with the Oxygen XML Web Author Component and they should not be annotated.
- The Oxygen XML Web Author Component continuously validates the XML documents using the default validation scenarios defined at *framework* level. Only the validation units that have the ***Automatic Validation** option selected in the **Edit Scenario** dialog box* that is accessed by editing a scenario in the ***Validation** subtab* when editing a document type.

Oxygen XML Web Author Component CSS Limitations

The Oxygen XML Web Author Component CSS support is compatible with that offered by the standalone distribution of Oxygen XML Editor, with the following exceptions:

- The + (*direct adjacent*) and > (*child selector*) structural selectors cannot be used to match table-related elements.
- Oxygen XML Editor CSS extensions are ignored on `print` media. If an Oxygen XML Editor CSS extension is used on the `screen` media, it will also be used on the `print` media.
- Oxygen XML Editor CSS extension *properties* and *functions* cannot be used in a rule that has a `:hover` pseudo-class in the selector. The `attr` function is also not supported in such a rule due to a lack of browser support.
- The `:hover` pseudo-class is only available for mouse-enabled platforms.
- Oxygen XML Editor CSS extensions used in property values that express lengths may not behave as expected. Nevertheless, it is a good approximation.
- Oxygen XML Editor synthetic DOM nodes `comment`, `reference`, `cdata`, `pi`, and `error` interfere with the + (*direct adjacent*) structural selector. For example:

  ```
  b + b {
  color: red;
  }
  ```

  will not match the following XML structure:

  ```
  <root>
    <b/>
    <!--comment-->
    <b/>
  </root>
  ```
- The Oxygen XML Web Author Component does not render non-`table-row` children elements of tables and non-`table-cell` elements of `table-row` elements.
- A `width` or `height` property set on any element other than the root XML element may cause some resize handles (that cannot be disabled) to be displayed in IE 11. This is also true for elements that have a `position` property with a value of `absolute` or `fixed`. For more information about this issue, see *this Microsoft Connect article*.
- The Oxygen XML Web Author Component does not support the following:
  - `:nth-last-of-type`, `:first-of-type`, `:last-of-type`, `:nth-last-of-type` pseudo-classes.
  - `-oxy-tags-color`, `-oxy-tags-background-color`, and `-oxy-foldable` properties.
  - Subject selectors, since they are not supported by web browsers.
  - Specifying widths for *inline elements*.
  - Attribute selectors that use wildcard for the attribute name.

- Oxygen XML Editor CSS extensions to style `:before` and `:after` pseudo-elements, except in the `content` property.
- CSS property values that contain the `oxy_xpath` function are not refreshed correctly.

To overcome these differences you can use media queries described in *Customization Tips*.

Oxygen XML Web Author Component Form Controls

Oxygen XML Web Author Component supports a variety of built-in form controls that allow users to interact with documents with familiar user interface objects. The **oxy_label** function is also supported and can be used to change the style of generated text.

Table 48: Supported Properties for Form Controls and Functions in Oxygen XML Web Author Component

Form control	Description	Supported properties
Button *form control*	This form control provides a button object that is used to invoke a specific action.	*color* *actionID* *transparent*
Button Group *form control*	This form control provides a group of button objects that are used to invoke specific actions.	*label* *icon* *actionDisplayStyle* *tooltips* *transparent*
Checkbox *form control*	This form control offers a single checkbox or multiple checkboxes that can be used to enable or disable an option.	*resultSeparator* *tooltips* *values* *uncheckedValues* *labels* *color*
Combo Box *form control*	This form control provides a graphical object that is a drop-down menu of proposed values. It can also be used for a combination of a drop-down menu and an editable single-line text field.	*columns* *editable* *tooltips* *values* *fontInherit* (always true) *labels* *color*
Date Picker *form control* **Note:** Some browsers interpret the HTML5 *datePicker* as a text-field.	This form control offers a text field that allows the user to choose a date in a specified format. Some browsers include a calendar browsing mechanism along with the text field.	*columns* *color* *format*
HTML Content *form control*	This form control is used for rendering HTML content. It is displayed as a graphical element shaped as a box. The shape of the box is determined by a specified width and the height is computed based upon the length of the text.	*href* *ID* *width*

Form control	Description	Supported properties
Pop-up *form control*	This form control offers a contextual menu that provides quick access to various actions. A pop-up form control can display single or multiple selections.	*color* *tooltips* *values* *resultSeparator* *selectionMode* *labels* *columns* *rendererSort* (has to be identical to *editorSort*)
Text Area *form control*	This form control provides a text box that can be used to enter multiple lines of text.	*type* *columns* (in average character width) *rows* *fontInherit* *tooltip* *color*
Text Field *form control*	This form control provides a text field box that is used for entering a single line of text.	*type* *columns* *fontInherit* (always true) *tooltip* *color*
URL Chooser *form control*	This form control provides a dialog box that allows you to select the location of local or remote resources. The inserted reference is made relative to the URL of the currently opened editor.	*columns* *color* *fontInherit* (always true)
oxy_label() *function*	This function can be used as a value for the CSS *content* property to change the style of generated text.	*text* *width* *color* *background-color* *inherit* *styles*

Oxygen XML Web Author Component Editor Variable Limitations

The Oxygen XML Web Author Component processes Oxygen XML Editor *editor variables*. However, the following categories of editor variables are not supported:

- Editor variables related with functionality that is not available in the Oxygen XML Web Author Component, such as *${dbgXML}* or *${dbgXSL}*.
- Editor variables related with Oxygen XML Editor project location, such as *${pdu}*, *${pd}*, or *${pn}*.
- Any editor variable that displays Java Swing-based components, such as *${ask}*.
- Editor variables related with the Oxygen XML Editor standalone installation directory, such as *${oxygenHome}* or *${oxygenInstallDir}*.

Related Information:

Editor Variables on page 169

Customizing Oxygen XML Web Author Component Plugins

The Oxygen XML Web Author Component server side can be customized with a variety of *plugin* types. We currently provide support for the following extension types:

- **URLStreamHandler** - This extensions can be used to integrate the Oxygen XML Web Author Component with XML databases or CMS. There is an example URLStreamHandler provided in *Oxygen XML SDK project* in the `oxygen-sample-plugins/oxygen-sample-plugin-custom-protocol` folder. The extension uses the **cproto** protocol to access the file system of the server and can be used as a starting point.

  Note: For more details about implementing an authentication mechanism, see *Implementing a CMS Authentication Mechanism* on page 1392.
- **WorkspaceAccess** - Most of the methods used to configure the Oxygen XML Editor GUI are unavailable in theses extensions, but they can still be used, for example, to configure a `javax.xml.transform.URIResolver`.

  Note: The *`ro.sync.exml.workspace.api.PluginWorkspace`* instance passed to the extension also implements the *`ro.sync.ecss.extensions.api.webapp.access.WebappPluginWorkspace`* interface and provides access to some Oxygen XML Web Author Component-specific functionality.
- **WebappServlet** - This extension allows you to provide an implementation of a servlet-like interface (*`ro.sync.ecss.extensions.api.webapp.plugin.WebappServletPluginExtension`*) that will be dynamically loaded by the Oxygen XML Web Author Component. Your implementation will also provide the path to the location where the servlet will be exposed.
- ***AuthorStylesheet*** - Allows you to add a stylesheet (CSS or LESS) used for rendering all XML documents.
- **WebappStaticResourcesFolder** - This extension allows you to access a static resource folder. It should provide a `path` attribute (the static resources folder path relative to the `plugins` directory) and an `href` attribute that declares the *plugin*. An example of a use-case is you can use it to have the Oxygen XML Web Author Component provide icons for *plugin*-specific actions.

  In the following example, the static resources will be available at *`[OXYGEN_WEBAUTHOR_INSTALL_DIR]`*`/plugin-resources/relative-href/path-to-file`, with the `path-to-file` being relative to the static resources folder:

```
<extension type="WebappStaticResourcesFolder" path="path-to-resorce-folder"
 href="relative-href"/>
```

Loading plugin-related custom JavaScript code

If your *plugin* needs accompanying JavaScript code to be loaded and executed on the client-side you can bundle it together with your *plugin* code. The Oxygen XML Web Author Component loads all files with the `.js` extension located in the web folder of the *plugin*. The files are loaded in *lexicographical* order, meaning that their alphabetical order is based upon the *total order* rather than *sequence* (for example, `abc10.js` would be loaded before `abc2.js`).

Adding the Plugins in the Oxygen XML Web Author Component

If you have already developed such Oxygen XML Editor *plugins*, they can be added in the `bundle-plugin/dropins` folder in the Maven project.

If you are developing a new Oxygen XML Editor *plugin* you are encouraged to use as a starting point any of the existing *plugins*. Then you should add the resulting Maven project as a dependency (or even a sub-module) in the `oxygen-sample-plugins` module.

Public Plugin Integration Projects

Some public projects are available on **github.com** that can be used to help you integrate Oxygen XML Web Author Component.

- **WebDAV Support for oXygen XML Web Author** (*https://github.com/oxygenxml/webapp-webdav-integration*) - This project is a very simple integration of Oxygen XML Web Author Component with a WebDAV-enabled server, which can be extended with more features or can be adapted to work with any CMS.
- Here is a list with *all open-source Oxygen XML Web Author Component plugins provided by Oxygen*. For example, there are plugins for integrating with Google Drive, editing files in a GitHub repository, adding support for special UTF-8 characters, implementing REST-endpoints for content handling, and more.

Related Information:

Setting Up a Development Environment for Oxygen XML Web Author Component Plugins on page 1393

Adding and Configuring Plugins on page 1377

Customizing Oxygen XML Web Author Component Client Side

Client side customization is available through a JavaScript API. Unlike the server side customization, it can be used to modify the application's GUI.

The Oxygen XML Web Author Component is an editing platform, but it is the job of the integrator to provide a way for the user to select which file is going to be edited. Afterwards, the user should be redirected to the Oxygen XML Web Author Component editing page, and the following three URL parameters specified:

- `url` - An absolute URL of the edited file.
- `ditamap` - (Optional parameter) An absolute URL taken into account only when editing a DITA file. Provides the *DITA map* context of the edited DITA file.
- `author` - The author name.
- `trackChanges` - Flag that controls whether the editor should track changes or not. Possible values:
  - default - The status of change tracking is determined by server's global options.
  - enabled - Change tracking is enabled but the user can disable it using a toolbar action.
  - forced - Change tracking is enabled and the user cannot disable it, not can she accept or reject any changes.

  If you use other option than 'default', the server change tracking status (as configured in the Administration Page) should not be "Stored in the document".
- `autoSaveInterval` - the interval of time (in seconds) to wait until an auto-save is performed. If <= 0 or falsy, auto-save is disabled.

Example:

Suppose that the Oxygen XML Web Author Component is deployed at the following URL:

```
http://www.example.com/oxygen-sdk-sample-webapp/
```

The user (John Doe) wants to edit a file (located at `http://www.test.com/topics/topic.xml`) in the context of a *DITA map* (located at `http://www.test.com/map.xml`). In this case, the editing URL should be:

```
http://www.example.com/oxygen-sdk-sample-webapp/app/oxygen.html
?url=http%3A%2F%2Fwww.test.com%2Ftopics%2Ftopic.xml
&ditamap=http%3A%2F%2Fwww.test.com%2Fmap.xml
&author=John%20Doe
```

Note: The parameter values are percent encoded before being added to the editing URL.

Customize Oxygen XML Web Author Component Using JavaScript Code

To extend the client-side functionality provided by Oxygen XML Web Author Component you can use the *JavaScript API*. To load your JavaScript customization code, use one of the following methods:

- Create a file called `plugin.js` and copy it in the `app` folder of the Oxygen XML Web Author Component deployment.
- *Bundle JavaScript code with a Java plugin.*
- Bundle the JavaScript code with a *framework*. Save your code in the `framework.js` file (located in the `web` folder inside the particular *framework* folder).

Related Information:

Customizing Oxygen XML Web Author Component Frameworks on page 1383

Oxygen XML Web Author Component JavaScript API

Oxygen XML Web Author Component How to ...

This section includes information on how to accomplish a variety of common use cases.

Adding an Edit Link in Your Interface

Oxygen XML Web Author Component can be launched to edit a file by simply following a link with a carefully built target. The base URL for the **Edit** link used to open the Web Author is the URL that you see in the browser when the **Dashboard** page is opened.

Example: The base URL for oXygen XML Web Author demo deployment

```
https://www.oxygenxml.com/webapp-demo-aws/app/oxygen.html
```

Table 49: The information about the file to open and the options used to load it can be specified as URL parameters:

Parameter name	Description
`url`	The *OXY-URL* of the file to be opened.
`ditamap`	For DITA files, the *OXY-URL* of the DITA Map to use as a context for resolving keys.
`trackChanges`	**default** Whether or not changes are tracked depend on the server settings. **enabled** The changes are tracked by default but the user is able to disabled it. **forced** The changes are tracked and the user cannot accept or reject changes made by others.
`author`	The name of the author

The *OXY-URL* uses a custom schema according to the file storage system the file comes from.

GitHub

The URLs have the following format:

```
github://getFileContent/<user>/<repo>/<branch>/path/to/file.dita
```

Git

The URLs have the following format:

```
github://getFileList/<encoded-repo-http-url>/<branch>/path/to/file.dita
```

where `encoded-repo-http-url` is the `http` URL of the git repository (for example: `https://user@bitbucket.org/user/flowers.git`).

WebDAV

The URLs have the following format:

```
webdav-https://dav.box.com/dav/path/to/file.dita
```

where the WebDAV URL of the file is: `https://dav.box.com/dav/path/to/file.dita`.

SharePoint

The URL of the file is the SharePoint file URL, prefixed with `spo-`.

REST

The URLs have the following format:

```
rest://<server-id>/path/to/file.dita
```

The format can be further specified by the implementer of the REST API.

Adding Custom Dictionaries for the Spell Checker

Oxygen XML Web Author Component includes an automatic spell checking feature and you can customize the words that are detected by adding custom dictionaries.

To add a custom dictionary, follow these steps:

1. Download the files needed for your custom dictionary (or create your own). You will need a *dictionary* file (`.dic` file extension) and an *affix* file (`.aff` file extension). If the dictionary did not include an affix file (`.aff`), you can create one and leave it empty, but it is needed for the mechanism to work properly.

   Tip: An example of a website that includes numerous dictionary files is: *http://extensions.services.openoffice.org/dictionary*.
2. Copy both files (`.dic` and `.aff`) to the `dicts` folder that is located inside your *Oxygen Data Directory* (`oxygen.data.dir`).

   Important: The name of the dictionary file should begin with a two letter prefix that indicates the language it should be attached to, followed by an underscore or hyphen (for example, `en_medical.dic` for a medical dictionary in the English language). For a list of language codes, see *https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes*.
3. Restart the server for the spell checker to start using the new dictionary.

Configuring Minimal File Access Permissions

The Oxygen XML Web Author Component requires access to the following file resources:

- *READ* access to the directory where the Oxygen XML Web Author Component is deployed.
- *READ* and *WRITE* access to the application's working directory.
- *READ* and *WRITE* access to JVM's temporary directory.

It is a good security practice to allow a component to access only the information and resources that are necessary for its purpose. In an environment that uses Apache Tomcat, you can enforce these rules following these steps:

- Start the Apache Tomcat server using the `-security` flag.
- Edit the `catalina.policy` file and add the following snippet:

```
grant codeBase "file:${catalina.base}/webapps/oxygenxml-web-author/-" {
  // Oxygen uses System properties for various configuration purposes.
  permission java.util.PropertyPermission "*", "read,write";
  // Oxygen custom protocols need access to network.
  permission java.net.NetPermission "*";
  permission java.net.SocketPermission "*", "accept,connect,listen,resolve";
  // The web framework used by Oxygen Webapp uses reflection and classloaders.
  permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
  permission java.security.SecurityPermission "*";
  permission java.util.logging.LoggingPermission "control";
  permission java.lang.RuntimePermission "*";

  // Oxygen requires these permissions to connect to a URL.
  permission java.net.URLPermission "http:*", "*";
  permission java.net.URLPermission "https:*", "*";
  permission java.net.URLPermission "file:*", "*";

  // Oxygen should be allowed to read JVM jars
  permission java.io.FilePermission "${java.home}/-", "read";

  // Oxygen uses the JVM's java.io.tempdir for various file handling tasks.
  permission java.io.FilePermission "${java.io.tmpdir}/-", "read,write,delete";
  permission java.io.FilePermission "${java.io.tmpdir}", "read,write,delete";

  // Folder used by Oxygen to deploy the plugins to.
  permission java.io.FilePermission "${oxygen.data.dir}/-", "read,write,delete";
  permission java.io.FilePermission "${oxygen.data.dir}", "read,write,delete";
};
// The jar that contains sandboxing code.
grant codeBase
  "jar:file:${catalina.base}/webapps/oxygenxml-web-author/WEB-INF/lib/
oxygen-sandbox.jar!/-" {
    permission java.security.AllPermission;
};
// Give all permissions to plugins code unless otherwise instructed by vendor.
grant codeBase "file:${oxygen.data.dir}/plugins/-" {
    permission java.security.AllPermission;
};
// Give all permissions to frameworks code unless otherwise instructed by vendor
grant codeBase "file:${oxygen.data.dir}/frameworks/-" {
    permission java.security.AllPermission;
```

```
};
```

Note: In the previous example, replace `oxygenxml-web-author` with the name of your deployment of the Oxygen XML Web Author Component.

Configuring File Permissions to Custom Locations

There are cases when the Oxygen XML Web Author Component needs to access files system resources, but due to security reasons, you want to prevent your users from opening them directly in the Oxygen XML Web Author Component editing page using the `file://` protocol.

You can do this by following these steps:

- Edit the `catalina.policy` file and add a line such as:

  ```
  permission java.io.FilePermission "path/to/yourSecretDir/-", "read,write,delete";
  permission java.io.FilePermission "path/to/yourSecretDir", "read,write,delete";
  ```
- Use the following system property when starting the Tomcat server:

  ```
  -Dfile.protocol.blacklist=/path/to/yourSecretDir
  ```

  Note: Use the value of `path.separator` system property to separate more directories. For example, under Linux, the value of `path.separator` property is a colon punctuation character `:`.

Embedding Oxygen XML Web Author Component in a CMS Page

Oxygen XML Web Author Component can be embedded in a CMS to offer editing functionality for documents stored in the CMS.

To embed Oxygen XML Web Author Component, load the main page (`app/oxygen.html`) in an `iframe`.

Security Restrictions

Oxygen XML Web Author Component uses cookies to enforce its licensing and to maintain the editing state of the opened documents. Some browsers block third-party cookies for security reasons. To avoid having the cookies blocked, Oxygen XML Web Author Component should be served from the same origin as the CMS host page.

Passing Parameters to the Editor

To control the editor, you can use URL parameters described in the *options section*. To pass custom parameters, you can implement a *plugin for Oxygen XML Web Author Component* that contains JavaScript code to interpret those parameters.

Communicating with the Editor

To communicate with the editor, you can send messages between the host CMS page and the Oxygen XML Web Author Component page.

Browsers offer the `window.postMessage` JavaScript API that allows messages to be sent between pages. To receive the message, create a *plugin for Oxygen XML Web Author Component* that contains JavaScript code that listens for the messages sent by the CMS host page and uses the *Oxygen XML Web Author Component JavaScript API* to implement the required functionality.

Other problems

- We experienced problems if the *iframe* is hidden while loading using `display: none`. You can use `visibility: hidden` for the same purpose.
- Some XML vocabularies (for example, DITA) support cross-document links. By default, clicking on a cross-document link will open that document in the Oxygen XML Web Author Component in another browser tab. If the editor is embedded, you may want to change this behavior using the `sync.api.Editor.EventTypes.LINK_OPENED` event that is triggered on the *`sync.api.Editor`* instance.

Enabling File Browsing for a Custom Protocol Handler

To allow users to insert images more easily, the Oxygen XML Web Author Component provides a file browsing JavaScript widget that can be used for any custom protocol *plugin*. To enable this widget, follow these steps:

1. Develop a *plugin* that implements the `ro.sync.exml.plugin.urlstreamhandler.URLStreamHandlerPluginExtension` interface. The `getURLStreamHandler` method should return an instance of `java.net.URLStreamHandler` (for example, `myHandler`).
2. `myHandler.openConnection()` should open an instance of the `ro.sync.net.protocol.FileBrowsingConnection` interface.
3. On the client-side, register the `sync.api.FileBrowsingDialog` widget as a `UrlChooser` using the following code snippet:

```
workspace.setUrlChooser(new sync.api.FileBrowsingDialog());
```

Implementing a CMS Authentication Mechanism

Suppose you want to impose an authentication step to all users who want to edit documents in the Oxygen XML Web Author Component. This is usually required when the CMS needs authentication before granting access to a file. The Oxygen XML Web Author Component provides both a server-side and client-side API that allows you to implement such a mechanism.

Implement CMS Authentication Mechanism

The following is a list of the basic building blocks of the authentication mechanism:

- Develop a *plugin* that implements the `ro.sync.exml.plugin.urlstreamhandler.URLStreamHandlerPluginExtension` interface. Considering the multiple user context of the Oxygen XML Web Author Component, the `getURLStreamHandler` method should return an instance of the `ro.sync.ecss.extensions.api.webapp.plugin.URLStreamHandlerWithContext` class. This class tracks the user, based on the URL connection that will be made.
- If the CMS rejects the connection attempt with a message that the user is not authenticated, you should throw a `ro.sync.ecss.extensions.api.webapp.plugin.UserActionRequiredException` exception. This exception is automatically relayed to the client-side as a `sync.api.WebappMessage` JavaScript object.
- On the client side, follow these steps:
  1. Use the `sync.api.Editor.EventTypes.CUSTOM_MESSAGE_RECEIVED` event to intercept the messages sent from the server-side.
  2. Display a dialog box to collect more authentication information from the user.
  3. Send the credentials to the server, more specifically to the `ro.sync.ecss.extensions.api.webapp.plugin.URLStreamHandlerWithContext` instance defined at step 1. For this part, you will need to implement a secure way to transmit the credentials. This can range from a simple servlet that runs in the Oxygen XML Web Author Component to an *OAuth* implementation.
  4. Retry the operation that triggered the authentication procedure.

Locating and Configuring Logs

How to Locate the Log File and the Log Configuration File

To locate the **Log file** where the logs are written, or the **Config file** used to configure logging, go to the ***Administration Page*** and in the **General** tab, you can view the location of the these files.

Enabling HTTP Request Logging

To enable a detailed logging of the HTTP requests sent by Oxygen XML Web Author Component, edit the **Config file** and add the following lines:

```
log4j.category.org.apache.http.impl.conn=debug
log4j.category.org.apache.http.impl.client=debug
```

```
log4j.category.org.apache.http.client=debug
log4j.category.org.apache.http.wire=debug
log4j.category.org.apache.http=debug
```

Note: Making changes to the **Config file** requires that you restart the application.

Opening a File as Read-Only in Oxygen XML Web Author Component

Note: This topic assumes that you are using a *`URLStreamHandler` plugin* to open the files in your repository.

When a file is opened, your *plugin* is asked to provide a `URLConnection` object that Oxygen XML Editor uses to read the contents of the file.

By opening the file as read-only, the Oxygen XML Web Author Component is instructed to visually warn users that the content of the document cannot edited. This is an alternative to having them edit the content and then receive an error when trying to check it in.

You can instruct the Oxygen XML Web Author Component to open a file as read-only by using one of the following methods:

- Implement the `java.net.URLConnection.getHeaderField(String)` method to return a value of `true` for the `oxygen_read_only` header name.
- Use a JavaScript extension that invokes an **Author** mode operation that changes the document status to read-only:

```
editor.getActionsManager().invokeOperation(
  'SetReadOnlyStatusOperation',
  {'read-only': true}
);
```

Registering Actions to Create or Open Documents for Configured Plugins

The Oxygen XML Web Author Component has a user-friendly **Dashboard** and if a particular *plugin* registers a create or open action, then it will include sections for creating and opening documents for *plugin* integrations.

Register Create and Open actions for the Oxygen XML Web Author Component Dashboard

To register *Create* and *Open* actions, follow these steps:

1. *Create a plugin* that has a `plugin.js` file in its `web` folder.
2. In the `plugin.js` file, add code similar to the following example:

```
// The base url to browse.
var initialUrl = 'file:/[PATH_TO_RESOURCE_FOLDER]';

var createAction = new sync.api.CreateDocumentAction(
    new sync.api.FileBrowsingDialog({initialUrl : initialUrl}));
createAction.setActionName('New Document');

var openAction =new sync.actions.OpenAction(
    new sync.api.FileBrowsingDialog({initialUrl : initialUrl}));
openAction.setActionName('Open Document');

var actionsManager = workspace.getActionsManager();
actionsManager.registerCreateAction(createAction);
actionsManager.registerOpenAction(openAction);
```

   Note: This example will provide browsing capabilities on your local file system. More complex connectors can be implemented using our Java and *JavaScript API*. Also, to see information for a more complex example, you can also look at our *Oxygen XML Web Author Component storage services integration project that is hosted on GitHub*.
3. *Deploy the plugin* in the Oxygen XML Web Author Component.

Result: The registered *create* action will appear in the **New** section on your **Dashboard**, while a registered *open* action will appear in the **Open** section.

Setting Up a Development Environment for Oxygen XML Web Author Component Plugins

You will need a recent Eclipse EE.

This procedure describes a development environment that can be used to increase your productivity in writing *plugins* for Oxygen XML Web Author Component.

Developing a *plugin* for Oxygen XML Web Author Component might require repetitive coding-testing cycles. Since the *process of building a whole SDK project* requires a full Maven build, the whole process might prove to be time consuming. The following procedure provide a faster alternate way of testing the *plugin*:

1. Go to the following repository and follow the instructions: *https://github.com/oxygenxml/web-author-plugin-archetype*.
2. Run Oxygen XML Web Author Component in a Tomcat server. You can either use one of the *installation kits*, or build it using the *Oxygen XML SDK*.
3. Look in the Tomcat logs (or in the console) for a line like "`Loading plugins from: ${path}`" and note the path of the `plugins` folder.
4. In the `plugins` folder, create a sub-folder with a name of your choice (for example, `myplugin`).
5. In that folder (`myplugin`), create a `plugin.redirect` file that contains the path to your *plugin* project (created in step 2) on a single line.
6. Import your *plugin* project in Eclipse.
   a) Click **File** > **Import**.
   b) Choose **Existing Maven Project**.
   c) Browse for the location of your *plugin*.
7. Modify the `plugin.xml` file to add a library reference to the directory where Eclipse places the compiled output.

   With the default setup of a Maven project, this step requires that you add the following element:

   ```
   <library name="target/classes/"/>
   ```

8. You can now open a document in the Oxygen XML Web Author Component and it will automatically load your *plugin*.

   Every time you make changes to the *plugin* sources, you will need to restart Oxygen XML Web Author Component.

Once you are happy with the result, you need to add the *plugin* back in your SDK project and follow *these instructions* to perform a final testing of the project.

Related Information:

Adding and Configuring Plugins on page 1377

Sharing a Tomcat Instance

If you want to share a Tomcat instance between Oxygen XML Web Author Component and another web application, the following issues need to be considered:

- Oxygen XML Web Author Component reads and sets system properties, and while we try to namespace those that are specific to Oxygen XML Web Author Component, there is no guarantee that there will not be any clashes with those set by other applications.
- You have to adapt the JVM's memory configuration to the scenario where there will be more applications competing for the same pool of memory.
- The Oxygen XML Web Author Component currently does not restart (or *reload* in Apache Tomcat terminology) correctly unless the Servlet Container is also restarted.

Troubleshooting Oxygen XML Web Author Component Errors

This topic is intended for system administrators and integration developers that want to troubleshoot deployment or programming errors. If you are not in either of those categories, you should contact your system administrator.

Oxygen XML Web Author Component intentionally does not provide explanatory messages for errors caused by deployment or programming mistakes. If you are trying to troubleshoot such a problem, you can find more information in the browser logs or in the server logs.

Browser logs

To see the browser logs, open the **Developer Tools** in your browser. This can usually be achieved by pressing **F12**.

Server logs

To see the server logs, follow the instructions detailed in the *Locating and Configuring Logs* on page 1392 topic.

Related Information:

Locating and Configuring Logs on page 1392

How to Set a System Property on page 1382

Using an IIS Reverse Proxy

If you want to use Oxygen XML Web Author Component with IIS as a reverse proxy, follow this procedure:

1. Configure IIS to allow double escaping in URLs. See the following examples:
   - For Microsoft Azure, the `applicationHost.xdt` file should contain the following:

     ```
     <?xml version="1.0" encoding="utf-8"?>
      <configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transform">
         <system.webServer>
          <security>
           <requestFiltering allowDoubleEscaping="true"
     xdt:Transform="SetAttributes(allowDoubleEscaping)" />
          </security>
         </system.webServer>
     </configuration>
     ```

   - For other types, insert the following fragment inside the `applicationHost.config` file:

     ```
     <security>
       <requestFiltering allowDoubleEscaping="true"/>
     </security>
     ```

2. Configure Tomcat to allow escaped slashes. Append the following line in the `tomcat\conf\catalina.properties` file:

   ```
   org.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH=true
   ```

3. Set an environment variable to instruct Oxygen XML Web Author Component that the URL path is already decoded. Insert the following line in the *[OXYGEN_WEBAUTHOR_INSTALL_DIR]*`\tomcat\bin\catalina.bat` file:

   ```
   set "URL_DECODING_PROXY=true"
   ```

Oxygen XML Author Options Supported by Web Author

Oxygen XML Web Author Component supports some of the options used by the Oxygen XML Author desktop application. The supported options are applied for all Web Author users. The options are stored in an XML file with the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<serialized version="18.1" xml:space="preserve">
  <map>
    <entry>
      <String>author.show.comments</String>
      <Boolean>true</Boolean
    </entry>
  </map>
</serialized>
```

For each option, an additional entry should be added in this file.

Table 50: Oxygen XML Author Options Supported in Oxygen XML Web Author Component

Key	Type	Description
dita.ot.directory	String	The directory path to the default DITA OT installation.

Key	Type	Description
track.changes.initial.state	Integer	Option for *track changes* initial state. • **0** - Initial state stored in document. • **1** - *Track changes* always on. • **2** - *Track changes* always off.
automatically.accept.certificates	boolean	Option that controls if Oxygen will accept all HTTPS certificates.
additional.frameworks.directories	*See example entry below*	An array of java.lang.String objects representing paths to the additional *frameworks* folders (may also contain editor variables).
WEBAPP_SHOW_ADMIN_PAGE_LINK	boolean	"True" to display the admin page link on the dashboard.
author.convert.external.content.on.paste	boolean	Option that controls whether or not the content pasted in Author mode should be converted to match the destination styles.
author.convert.external.content.space.preserve	boolean	Option that controls whether or not the content pasted in **Author** mode should be converted to match the destination styles in *space-preserved elements*.
http.read.timeout.seconds	Integer	An integer number that configures the timeout used when waiting for an HTTP request.
http.proxy.system	boolean	"True" to detect HTTP proxy from system.
http.proxy.set	boolean	HTTP proxy uses manual configuration.
http.proxy.port	Integer	Proxy port.
http.proxy.host	String	Proxy hostname or IP address.
http.proxy.user	String	Proxy user.
http.proxy.password	String	Proxy password.
http.proxy.direct	String	Comma separated list of hosts for which the proxy is bypassed.
http.max.simultaneous.connections.per.host	Integer	Limits the number of connections the HTTP client can open to the same server host.
author.show.comments	boolean	Show the comment nodes in the author page.
author.show.processing.instructions	boolean	Show the pi nodes in the author page.
autocorrect.feature.state	boolean	Used to enable/disable the auto-correct feature.
autocorrect.use.suggeestions.from.spell.checkerdicts	boolean	Used to enrich the auto-correct suggestions with entries from spell checking dictionaries.

Key	Type	Description
author.format.compatibility	Integer	Possible values include: 0, 1, 2 0: Nothing special is done for compatibility purpose. 1: Do not break lines, do not indent. 2: Break lines only after elements displayed as blocks, do not indent.

Example entry for `additional.frameworks.directories` option:

```
<entry>
  <String>
   additional.frameworks.directories
  </String>
  <String-array
    <String>
     /path/to/frameworks
    </String>
  </String-array>
</entry>
```

Feature Matrix

The Oxygen XML Author Component was designed to provide the functionality of the standard **Author** mode and can be embedded either in a third-party standalone Java application or customized as a Java Web Applet to provide WYSIWYG-like XML editing directly in your choice of web browsers.

The Oxygen XML Web Author and Oxygen XML Web Author Component are re-implementations of the Oxygen XML Editor **Author** mode user interface, based on JavaScript and HTML5. Its purpose is to enable XML editing and reviewing on your mobile devices and desktops, directly in a web browser environment. Since the interface was thinned down as much as possible, the core XML processing was moved into a Java-enabled server.

Table 51: Feature Matrix Showing Differences Between Web Author, Web Author Component, and Author Component

Feature	Oxygen XML Web Author	Oxygen XML Web Author Component	Oxygen XML Author Component
Intended audience	Reviewers and occasional contributors.	Reviewers and occasional contributors.	Content authors, technical writers.
Mobile device support	Specifically designed for mobile devices.	Specifically designed for mobile devices.	No
Compatibility with the standard version of Oxygen XML Editor	Covers only editing and reviewing features.	Covers only editing and reviewing features.	100%
Text Mode	Yes	Yes	Yes
Grid Mode	No	No	Yes
Client-side setup	Yes, with an *Administration* page.	Yes, with an *Administration* page.	Requires Java to be installed, and Java Applets to be allowed to run.
Server-side setup	Simply requires running an installer.	Requires a servlet container and performing a Maven build.	Requires a web server.

Feature	Oxygen XML Web Author	Oxygen XML Web Author Component	Oxygen XML Author Component
Ability to configure installation bundles	No	Yes	Yes

Tools

17

Topics:

- *Refactoring XML Documents*
- *Generating Sample XML Files*
- *Converting Schema to Another Schema Language*
- *Converting Database to XML Schema*
- *Flatten an XML Schema*
- *XML to JSON Converter*
- *Compiling an XSL Stylesheet for Saxon*
- *Format and Indent (Pretty-Print) Multiple Files*
- *Generate Documentation*
- *Canonicalizing Files*
- *Signing Files*
- *Verifying Signature*
- *WSDL SOAP Analyzer*
- *XML Schema Regular Expressions Builder*
- *Large File Viewer*
- *Hex Viewer*
- *Standalone SVG Viewer*
- *Tree Editor*
- *Compare Files*
- *Compare Directories*
- *Compare Directories Against a Base (3-Way)*
- *Syncro SVN Client*
- *External Tools*

Oxygen XML Editor includes a variety of helpful tools to help you accomplish XML-related tasks. This section presents many of those tools. These tools are available in the **Tools** menu and some of them can be launched through keyboard shortcuts or command-line scripts.

Refactoring XML Documents

In the life cycle of XML documents there are instances when the XML structure needs to be changed to accommodate various needs. For example, when an associated schema is updated, an attribute may have been removed, or a new element added to the structure.

These types of situations cannot be resolved with a traditional *Find/Replace* tool, even if the tool accepts regular expressions. The problem becomes even more complicated if an XML document is computed or referenced from multiple modules, since multiple resources need to be changed.

To assist you with these types of refactoring tasks, Oxygen XML Editor includes a specialized **XML Refactoring** tool that helps you manage the structure of your XML documents.

XML Refactoring Tool

The **XML Refactoring** tool is presented in the form of an easy to use wizard that is designed to reduce the time and effort required to perform various structure management tasks. For example, you can insert, delete, or rename an attribute in all instances of a particular element that is found in all documents within your project.

To access the tool, select the **XML Refactoring** action from one of the following locations:

- The **Tools** menu.
- The **Refactoring** submenu from the contextual menu in the ***Project*** *view*.
- The **Refactoring** submenu from the contextual menu in the ***DITA Maps Manager*** *view*.

Note: The predefined refactoring operations are also available from the **Refactoring** submenu in the contextual menu of **Author** or **Text** mode. This is useful because by selecting the operations from the contextual menu, Oxygen XML Editor considers the editing context to skip directly to the wizard page of the appropriate operation and to help you by preconfiguring some of the parameter values. For your convenience, the last 5 operations that were *finished* or *previewed* also appear in the **Refactoring** submenu of the contextual menu in the **Project** view and the **DITA Maps Manager**.

XML Refactoring Wizard

The XML Refactoring tool includes the following wizard pages:

Refactoring operations

The first wizard page presents the available operations, grouped by category. To search for an operation, you can use the filter text box at the top of the page.

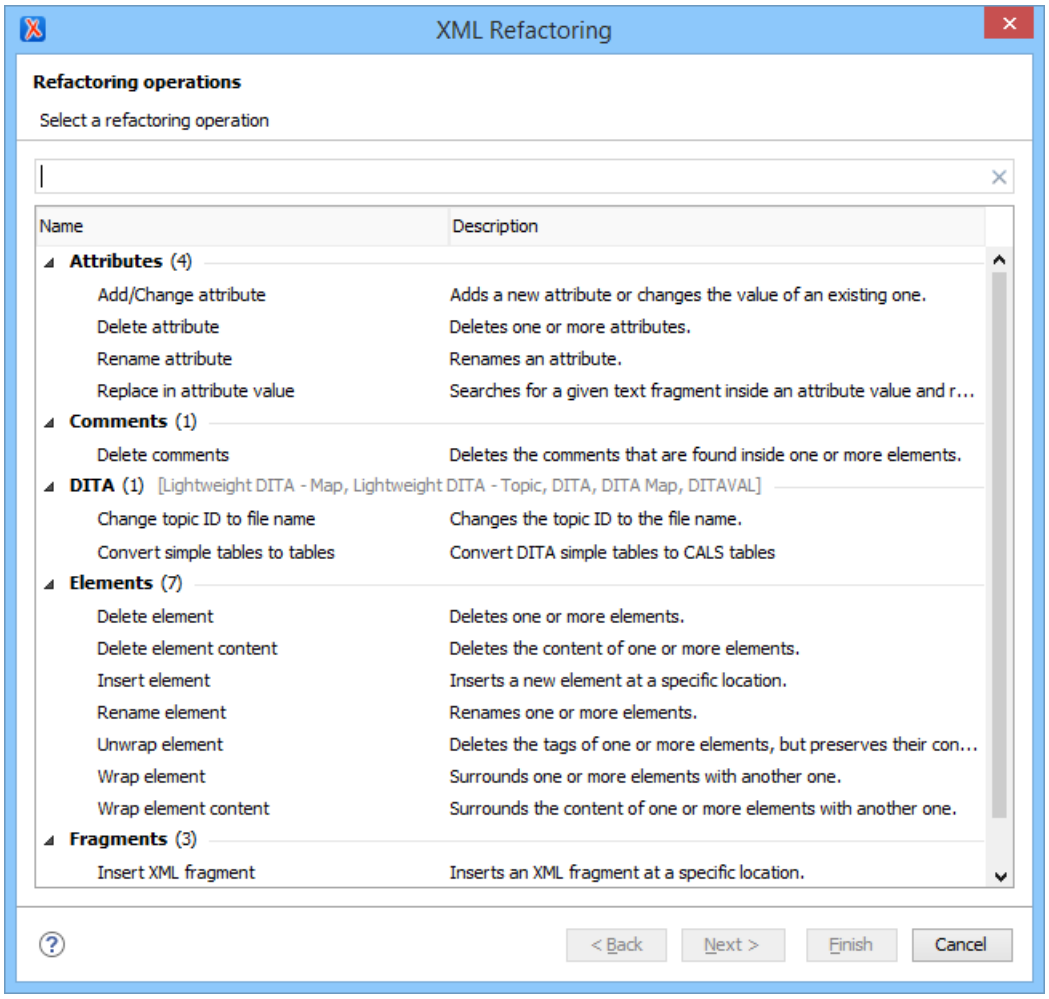


Figure 600: XML Refactoring Wizard

Configure Operation Parameters

The next wizard page allows you to specify the parameters for the refactoring operation. The parameters are specific to the type of refactoring operation that is being performed. For example, to delete an attribute you need to specify the parent element and the qualified name of the attribute to be removed.

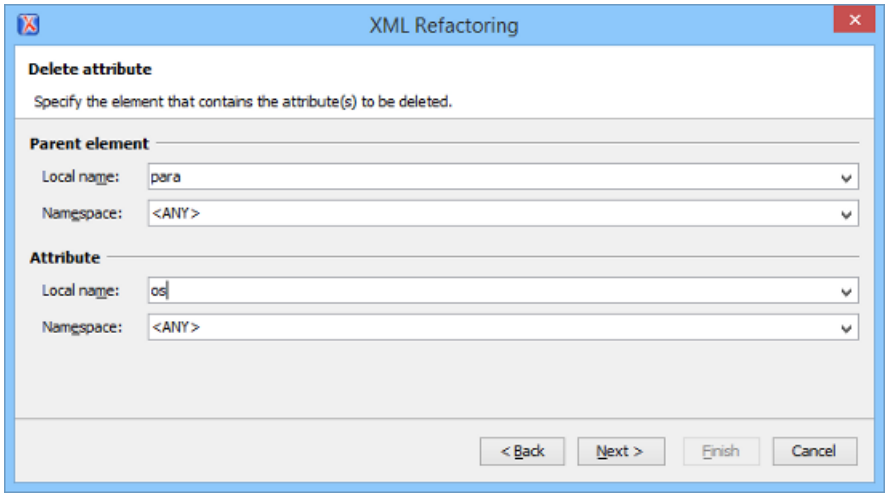


Figure 601: XML Refactoring 2nd Wizard Page (Delete Attribute Operation)

Scope and Filters

The last wizard page allows you to select the set of files that represent the input of the operation.

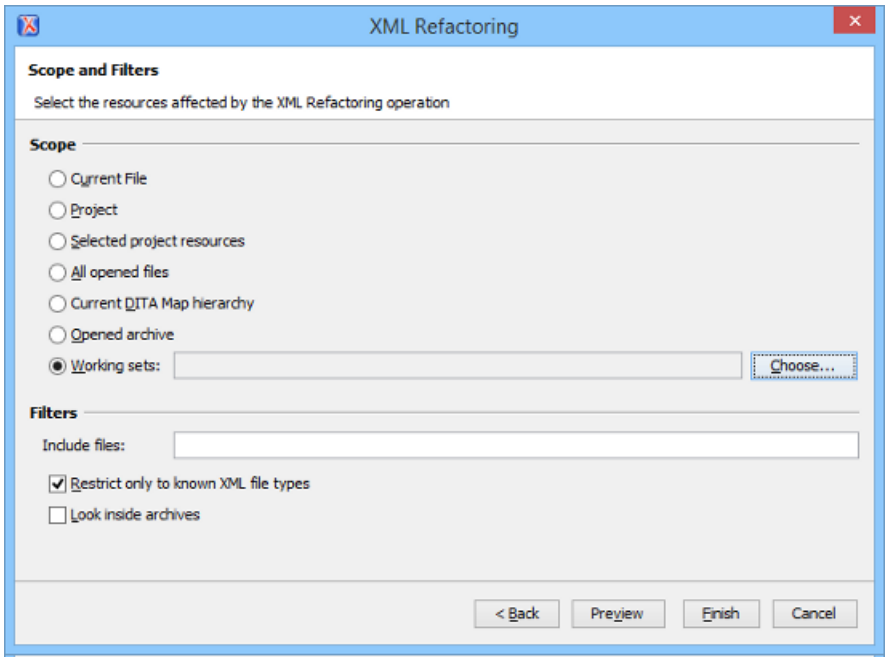


Figure 602: XML Refactoring - Scope and Filters Wizard Page

Scope section

In the **Scope** section, you can select from predefined resource sets (such as the current file, your whole project, the current *DITA map* hierarchy for DITA projects, etc.) or you can define your own set of resources by creating a *working set*.

Filters

The **Filters** section includes the following options:

- **Include files** - Allows you to filter the selected resources by using a file pattern. For example, to restrict the operation to only analyze build files you could use `build*.xml` for the file pattern.

- **Restrict only to known XML file types** - When selected, only resources with a known XML file type will be affected by the operation.
- **Look inside archives** - When selected, the resources inside archives will also be affected.

Preview

You can use the **Preview** button to open a comparison panel where you can review all the changes that will be made by the refactoring operation before applying the changes.

Finish

After clicking the **Finish** button, the operation will be processed and Oxygen XML Editor provides no automatic means for reverting the operations. Any **Undo** action will only revert changes on the current document.

Tip: If an operation takes longer than expected you can use the **Stop** button in the progress bar to cancel the operation.

Predefined Refactoring Operations

The XML Refactoring tool includes a variety of predefined operations that can be used for common refactoring tasks. They are grouped by category in the **Refactoring operations** wizard page. You can also access the operations from the **Refactoring** submenu in the contextual menu of **Author** or **Text** mode. The operations are also grouped by category in this submenu. When selecting the operations from the contextual menu, Oxygen XML Editor considers the editing context to get the names and namespaces of the current element or attribute, and uses this information to preconfigure some of the parameter values for the selected refactoring operation.

Tip: Each operation includes a link in the lower part of the wizard that opens the **XML / XSLT-FO-XQuery / XPath** preferences page where you can configure XPath options and declare namespace prefixes.

The following predefined operations are available:

Refactoring Operations for *Attributes*

Add/Change attribute

Use this operation to change the value of an attribute or insert a new one. This operation allows you to specify the following parameters:

- **Parent element** section
  - **Element** - The parent element of the attribute to be changed, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.
- **Attribute** section
  - **Local name** - The local name of the affected attribute.
  - **Namespace** - The namespace of the affected attribute.
  - **Value** - The value for the affected attribute.
- **Options** section
  - You can choose between one of the following options for the **Operation mode**:
    - **Add the attribute in the parent elements where it is missing**
    - **Change the value in the parent elements where the atrribute already exists**
    - **Both**

Delete attribute

Use this operation to remove one or more attributes. This operation requires you to specify the following parameters:

- **Element** - The parent element of the attribute to be deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.
- **Attribute** - The name of the attribute to be deleted.

Rename attribute

Use this operation to rename an attribute. This operation requires you to specify the following parameters:

- **Element** - The parent element of the attribute to be renamed, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.
- **Attribute** - The name of the attribute to be renamed.
- **New local name** - The new local name of the attribute.

Replace in attribute value

Use this operation to search for a text fragment inside an attribute value and change the fragment to a new value. This operation allows you to specify the following parameters:

- **Target attribute** section
  - **Element** - The parent element of the attribute to be modified, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.
  - **Attribute** - The name of the attribute to be modified.
- **Find / Replace** section
  - **Find** - The text fragments to find. You can use Perl-like regular expressions.
  - **Replace with** - The text fragment to replace the target with. This parameter can bind regular expression capturing groups ($1 , $2, etc.) from the find pattern.

Refactoring Operations for *Comments*

Delete comments

Use this operation to delete comments from one or more elements. This operation requires you specify the following parameter:

- **Element** - The target element (or elements) for which comments will be deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

Note: Comments that are outside the root element will not be deleted because the *serializer* preserves the content before and after the root.

Refactoring Operations for *DITA*

Change topic ID to file name

Use this operation to change the ID of a topic to be the same as its file name.

Convert conrefs to conkeyrefs

Use this operation to convert `conref` attributes to `conkeyref` attributes. For more information and instructions for using this operation, see *Converting Conrefs to Conkeyrefs* on page 1658.

Convert simple tables to CALS tables

Use this operation to convert DITA simple tables to CALS tables.

Convert to Concept

Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept). For more information, see *Converting DITA Topics to Another Type* on page 1624.

Convert to Reference

Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference). For more information, see *Converting DITA Topics to Another Type* on page 1624.

Convert to Task

Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task). For more information, see *Converting DITA Topics to Another Type* on page 1624.

Convert to Topic

Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic). For more information, see *Converting DITA Topics to Another Type* on page 1624.

Convert to Troubleshooting

Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting). For more information, see *Converting DITA Topics to Another Type* on page 1624.

All of these DITA refactoring actions allow you to choose a scope for the operation and some filters:

- **Scope** - Select from a variety of options to define the scope for which resources will be affected by the operation. For example, you can choose to affect all resources in the **Project**, **All opened files**, **Current DITA map hierarchy**, or just the **Current file**.
- **Filters** section
  - **Include files** - Specifies files to be excluded from the operation. You can specify multiple files by separating them with commas and the patterns can include wildcards (such as ***** or **?**).
  - **Restrict to known XML file types only** - Excludes non-XML file types from the operation.
  - **Look inside archives** - If this option is selected, the scope of the operation will include files inside archives.

Refactoring Operations for *Elements*

Delete element

Use this operation to delete elements. This operation requires you to specify the following parameter:

- **Element** - The target element to be deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

Delete element content

Use this operation to delete the content of elements. This operation requires you to specify the following parameter:

- **Element** - The target element whose content is to be deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

Insert element

Use this operation to insert new elements. This operation allows you to specify the following parameters:

- **Element** section
  - **Local name** - The local name of the element to be inserted.
  - **Namespace** - The namespace of the element to be inserted.
- **Location** section
  - **XPath**- An XPath expression that identifies an existing element to which the new element is relative, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.
  - **Position** - The position where the new element will be inserted, in relation to the specified existing element. The possible selections in the drop-down menu are: **After**, **Before**, **First child**, or **Last child**.

Rename element

Use this operation to rename elements. This operation requires you to specify the following parameters:

- **Target elements (XPath)** - The target elements to be renamed, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.
- **New local name** - The new local name of the element.

Unwrap element

Use this operation to remove the surrounding tags of elements, while keeping the content unchanged. This operation requires you to specify the following parameter:

- **Target elements (XPath)** - The target elements whose surrounding tags will be removed, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

Wrap element

Use this operation to surround elements with element tags. This operation allows you to specify the following parameters:

- **Target elements (XPath)** - The target elements to be surrounded with tags, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.
- **Wrapper element** section
  - **Local name** - The local name of the *Wrapper element*.

- **Namespace** - The namespace of the *Wrapper element*.

Wrap element content

Use this operation to surround the content of elements with element tags. This operation allows you to specify the following parameters:

- **Target elements (XPath)** - The target elements whose content will be surrounded with tags, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.
- **Wrapper element** section
  - **Local name** - The local name of the *Wrapper element* that will surround the content of the target.
  - **Namespace** - The namespace of the *Wrapper element* that will surround the content of the target.

Refactoring Operations for *Fragments*

Insert XML fragment

Use this operation to insert an XML fragment. This operation allows you to specify the following:

- **XML Fragment** - The XML fragment to be inserted.
- **Location** section
  - **XPath** - An XPath expression that identifies an existing element to which the inserted fragment is relative, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.
  - **Position** - The position where the fragment will be inserted, in relation to the specified existing element. The possible selections in the drop-down menu are: **After**, **Before**, **First child**, or **Last child**.

Replace element content with XML fragment

Use this operation to replace the content of elements with an XML fragment. This operation allows you to specify the following parameters:

- **Target elements (XPath)** - The target elements whose content will be replaced, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.
- **XML Fragment** - The XML fragment with which to replace the content of the target element.

Replace element with XML fragment

Use this operation to replace elements with an XML fragment. This operation allows you to specify the following parameters:

- **Target elements (XPath)** - The target elements to be replaced, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.
- **XML Fragment** - The XML fragment with which to replace the target element.

Refactoring Operations for *JATSKit*

Add BITS DOCTYPE - NLM/NCBI Book Interchange 2.0

Use this operation to add an NLM 'BITS' 2.0 DOCTYPE declaration.

Add Blue DOCTYPE - NISO JATS Publishing 1.1

Use this operation to add a JATS 'Blue' 1.1 DOCTYPE declaration.

Normalize IDs

Use this operation to normalize assigned IDs and assigned IDs to elements that are missing them.

All of these JATSKit refactoring actions allow you to choose a scope for the operation and some filters:

- **Scope** - Select from a variety of options to define the scope for which resources will be affected by the operation. For example, you can choose to affect all resources in the **Project**, **All opened files**, or just the **Current file**.
- **Filters** section
  - **Include files** - Specifies files to be excluded from the operation. You can specify multiple files by separating them with commas and the patterns can include wildcards (such as ***** or **?**).
  - **Restrict to known XML file types only** - Excludes non-XML file types from the operation.

- **Look inside archives** - If this option is selected, the scope of the operation will include files inside archives.

Additional Notes

Note: There are some operations that allow <ANY> for the **local name** and **namespace** parameters. This value can be used to select an element or attribute regardless of its local name or namespace. Also, the <NO_NAMESPACE> value can be used to select nodes that do not belong to a namespace.

Note: Some operations have parameters that accept XPath expressions to match elements or attributes. In these XPath expressions you can only use the prefixes declared in the ***Options > Preferences > XML > XSLT-FO-XQUERY > XPath*** page. This preferences page can be easily opened by clicking the link in the note (**Each prefix used in an XPath expression must be declared in the Default prefix-namespace mappings section**) at the bottom of the **Configure Operation Parameters** wizard page.

Custom Refactoring Operations

While Oxygen XML Editor includes a variety of predefined XML refactoring operations to help you accomplish particular tasks, you can also create custom operations according to your specific needs. For example, you could create a custom refactoring operation to convert an attribute to an element and insert the element as the first child of the parent element.

An XML Refactoring operation is defined as a pair of resources:

- An *XQuery Update script* or *XSLT stylesheet* that Oxygen XML Editor will run to refactor the XML files.
- An *XML Operation Descriptor* file that contains information about the operation (such as the name, description, and parameters).

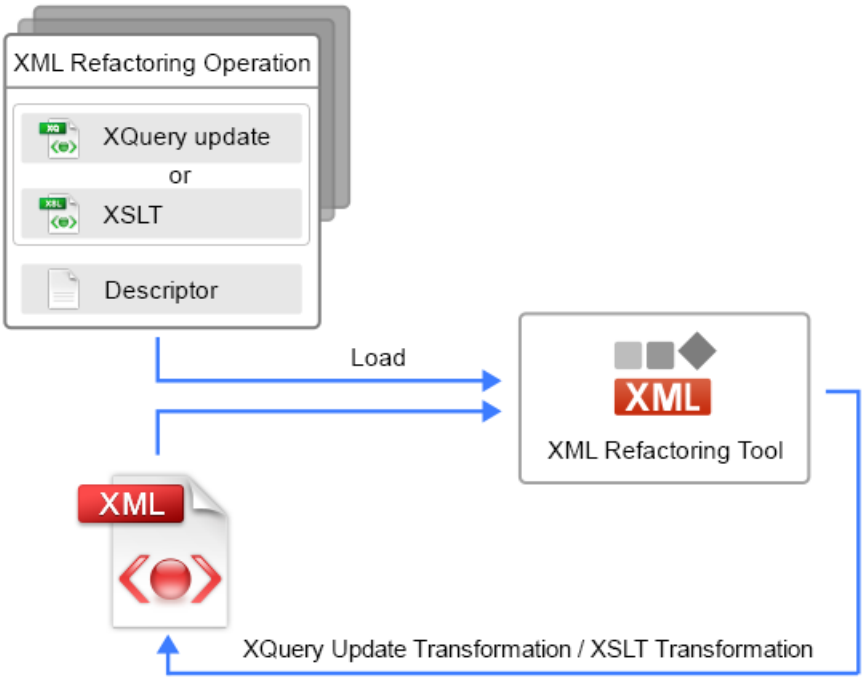


Figure 603: Diagram of an XML Refactoring Operation

All the defined custom operations are loaded by the **XML Refactoring Tool** and presented in *the **Refactoring Operations** wizard page*, along with the predefined built-in operations.

After the user chooses an operation and specifies its parameters, Oxygen XML Editor processes an XQuery Update or XSLT transformation over the input file. This transformation is executed in a **safe mode**, which implies the following:

- When loading the document:

- The **XInclude** mechanism is disabled. This means that the resources included by using XInclude will not be visible in the transformation.
- The DTD entities will be processed without being expanded.
- The associated DTD will be not loaded, so the default attributes declared in the DTD will not be visible in the transformation.

- When saving the updated XML document:
  - The DOCTYPE will be preserved.

    Note: This can be changed using *Saxon extension functions in XSLT*.
  - The DTD entities will be preserved as they are in the original document when the document is saved.
  - The attribute values will be kept in their original form without being normalized.
  - The spaces between attributes are preserved. Basically, the spaces are lost by a regular XML serialization since they are not considered important.

The result of this transformation overrides the initial input file.

Note: To achieve some of the previous goals, the XML Refactoring mechanism adds several attributes that are interpreted internally. The attributes belong to the `http://www.oxygenxml.com/ns/xmlRefactoring/additional_attributes` namespace. These attributes should not be taken into account when processing the input XML document since they are discarded when the transformed document is serialized.

Restriction: *Comments* or *processing instructions* that are in any node before or after the root element cannot be modified by an XML Refactoring operation. In other words, XML Refactoring operations can only be applied on the root element and the nodes inside it. However, as a work around to this limitation, you can use *Saxon extension functions and the XSLT stylesheet method* to implement the new custom XML refactoring operation.

Creating a Custom Refactoring Operation

To create a custom refactoring operation, follow these steps:

1. Create an *XQuery Update script* or *XSLT stylesheet file*.
2. Create an **XML Refactoring Operation Descriptor** file contains the path to the *XQuery Update script* or *XSLT stylesheet*.
3. Store both files in *one of the locations that Oxygen XML Editor* scans when loading the custom operations.

   Result: Once you run the **XML Refactoring** tool again, the custom operation appears in *the **Refactoring Operations** wizard page*.

Related Information:

Storing and Sharing Refactoring Operations on page 508

Custom Refactoring Script

The first step in creating a custom refactoring operation is to create an *XQuery Update script* or *XSLT stylesheet* that is needed to process the refactoring operations. The easiest way to create this script file is to use the **New** document wizard to create a new **XQuery** or **XSLT** file and you can use our *XQuery method example* or *XSLT method example* to help you with the content.

There are cases when it is necessary to add parameters in the *XQuery script* or *XSLT stylesheet*. For instance, if you want to rename an element, you may want to declare an external parameter associated with the name of the element to be renamed. To allow you to specify the value for these parameters, they need to be declared in the *refactoring operation descriptor file* that is associated with this operation.

Note: The XQuery Update processing is disabled by default in Oxygen XML Editor. Thus, if you want to create or edit an XQuery Update script you have to enable this mechanism by creating an *XQuery transformation scenario* and choose **Saxon EE** as the transformation engine. Also, you need to make sure the ***Enable XQuery update** option is selected in the Saxon processor advanced options*.

Note: If you are using an XSLT file, XPath expressions that are passed as parameters will automatically be rewritten to conform with the mapping of the namespace prefixes declared in the ***XML /XSLT-FO-XQuery / XPath** preferences page*.

The next step in creating a custom refactoring operation is to create an **XML Refactoring Operation Descriptor** file contains the path to the *XQuery Update script* or *XSLT stylesheet*.

Related Information:

XQuery Update Script for Creating a Custom Operation on page 502

XSLT Stylesheet for Creating a Custom Operation on page 504

Custom Refactoring Operation Descriptor File

The second step in creating a custom refactoring operation is to create an operation descriptor file. The easiest way to do this is to use the **New** document wizard and choose the **XML Refactoring Operation Descriptor** template.

Introduction to the Descriptor File

This file contains information (such as `name`, `description`, and `id`) that is necessarily when loading an XML Refactoring operation . It also contains the path to the *XQuery Update script* or *XSLT stylesheet* that is associated with the particular operation through the `script` element.

You can specify a `category` for your custom operations to logically group certain operations. The `category` element is optional and if it is not included in the descriptor file, the default name of the category for the custom operations is *Other operations*.

The descriptor file is edited and validated against the following schema: `frameworks/xml_refactoring/operation_descriptor.xsd`.

Declaring Parameters in the Descriptor File

If the XQuery Update script or XSLT stylesheet includes parameters, they should be declared in the **parameters** section of the descriptor file. All the parameters specified in this section of the descriptor file will be displayed in the **XML Refactoring** tool within *the **Configure Operation Parameters** wizard page* for that particular operation.

The value of the first `description` element in the **parameters** section will be displayed at the top of *the **Configure Operation Parameters** wizard page*.

To declare a parameter, specify the following information:

- **`label`** - This value is displayed in the user interface for the parameter.
- **`name`** - The parameter name used in the XQuery Update script or XSLT stylesheet and it should be the same as the one declared in the script.
- **`type`** - Defines the type of the parameter and how it will be rendered. There are several types available:
  - `TEXT` - Generic type used to specify a simple text fragment.
  - `XPATH` - Type of parameter whose value is an XPATH expression. For this type of parameter, Oxygen XML Editor will use a text input with corresponding content completion and syntax highlighting.

    Note: The value of this parameter is transferred as plain text to the XQuery Update or XSLT transformation without being evaluated. You should evaluate the XPath expression inside the XQuery Update script or XSLT stylesheet. For example, you could use the `saxon:evaluate` Saxon extension function.

    Note: A relative XPath expression is converted to an absolute XPath expression by adding `//` before it (`//XPathExp`). This conversion is done before transferring the XPath expression to the XML refactoring engine.

    Note: When writing XPath expressions, you can only use prefixes declared in the ***Options*** > ***Preferences*** > ***XML*** > ***XSLT-FO-XQUERY*** > ***XPath*** options page.
  - `NAMESPACE` - Used for editing namespace values.
  - `REG_EXP_FIND` - Used when you want to match a certain text by using Perl-like regular expressions.
  - `REG_EXP_REPLACE` - Used along with `REG_EXP_FIND` to specify the replacement string.
  - `XML_FRAGMENT` - This type is used when you want to specify an XML fragment. For this type, Oxygen XML Editor will display a text area specialized for inserting XML documents.
  - `NC_NAME` - The parameter for `NC_NAME` values. It is useful when you want to specify the local part of a *QName* for an element or attribute.

- `BOOLEAN` - Used to edit boolean parameters.
- `TEXT_CHOICE` - It is useful for parameters whose value should be from a list of possible values. Oxygen XML Editor renders each possible value as a radio button option.

- **`description`** - The description of the parameter. It is used by the application to display a tooltip when you hover over the parameter.
- **`possibleValues`** - Contains the list with possible values for the parameter and you can specify the default value, as in the following example:

```
<possibleValues onlyPossibleValuesAllowed="true">
    <value name="before">Before</value>
    <value name="after"default="true">After</value>
    <value name="firstChild">First child</value>
    <value name="lastChild">Last child</value>
</possibleValues>
```

Specialized Parameters to Match Elements or Attributes

If you want to match elements or attributes, you can use some specialized parameters, in which case Oxygen XML Editor will propose all declared elements or attributes based on the schema associated with the currently edited file. The following specialized parameters are supported:

`elementLocation`

This parameter is used to match elements. For this type of parameter, the application displays a text field where you can enter the element name or an XPath expression. The text from the `label` attribute is displayed in the application as the label of the text field. The `name` attribute is used to specify the name of the parameter from the XQuery Update script or XSLT stylesheet. If the value of the `useCurrentContext` attribute is set to `true`, the element name from the cursor position is used as proposed values for this parameter.

Example of an `elementLocation`:

```
<elementLocation name="elem_loc" useCurrentContext="false">
                <element label="Element location">
                    <description>Element location description.</description>
                </element>
</ ElementLocation>
```

`attributeLocation`

This parameter is used to match attributes. For this type of parameter, the application displays two text fields where you can enter the parent element name and the attribute name (both text fields accept XPath expressions for a finer match). The text from the `label` attributes is displayed in the application as the label of the associated text fields. The `name` attribute is used to specify the name of the parameter from the XQuery Update script or XSLT stylesheet. The value of this parameter is an XPath expression that is computed by using the values of the expression from the `element` and `attribute` text fields. For example, if `section` is entered for the element and a `title` is entered for the attribute, the XPath expression would be computed as `//section/@title`. If the value of the `useCurrentContext` attribute is set to `true`, the element and attribute name from the cursor position is used as proposed values for the operation parameters.

Example of an `attributeLocation`:

```
<attributeLocation name="attr_xpath" useCurrentContext="true">
                <element label="Element path">
                    <description>Element path description.</description>
                </element>
                <attribute label="Attribute" >
                    <description>Attribute path description.</description>
                </attribute>
</ AttributeLocation>
```

`elementParameter`

This parameter is used to specify elements by local name and namespace. For this type of parameter, the application displays two combo boxes with elements and namespaces collected from the associated schema of the currently edited file. The text from the `label` attribute is displayed in the application as label of the associated combo. The `name` attribute is used to specify the name of the parameter from the XQuery Update script or XSLT stylesheet. If you specify the `allowsAny` attribute, the application will propose *<ANY>* as a

possible value for the **Name** and **Namespace** combo boxes. You can also use the `useCurrentContext` attribute and if its value is set to `true`, the element name and namespace from the cursor position is used as proposed values for the operation parameters.

Example of an `elementParameter`:

```
<elementParameter id="elemID">
    <localName label="Name" name="element_localName" allowsAny="true"
 useCurrentContext="true">
        <description>Local name of the parent element.</description>
    </localName>
    <namespace label="Namespace" name="element_namespace" allowsAny="true">
        <description>Local name of the parent element</description>
    </namespace>
</elementParameter>
```

`attributeParameter`

This parameter is used to specify attributes by local name and namespace. For this type of parameter, the application displays two combo boxes with attributes and their namespaces collected from the associated schema of the currently edited file. The text from the `label` attribute is displayed in the application as the label of the associated combo box. You can also use the `useCurrentContext` attribute and if its value is set to `true`, the attribute name and namespace from the cursor position is used as proposed values for the operation parameters.

Note: An `attributeParameter` is dependant upon an `elementParameter`. The list of attributes and namespaces are computed based on the selection in the `elementParameter` combo boxes.

Example of an `attributeParameter`:

```
<attributeParameter dependsOn="elemID">
    <localName label="Name" name="attribute_localName" useCurrentContext="true">
        <description>The name of the attribute to be converted.</description>
    </localName>
    <namespace label="Namespace" name="attribute_namespace" allowsAny="true">
        <description>Namespace of the attribute to be converted.</description>
    </namespace>
</attributeParameter>
```

Note: All predefined operations are loaded from the *[OXYGEN_INSTALL_DIR]*`/refactoring` folder.

Related Information:

Example of an Operation Descriptor File with an XSLT Stylesheet on page 506
Example of an Operation Descriptor File with an XQuery Update script on page 503

XSLT Stylesheet for Creating a Custom Operation

To demonstrate creating a custom operation, consider that we have a task where we need to convert an attribute into an element and insert it inside another element. A specific example would be if you have a project with a variety of `image` elements where a deprecated `alt` attribute was used for the description and you want to convert all instances of that attribute into an element with the same name and insert it as the first child of the `image` element.

Thus, our task is to convert this attribute into an element with the same name and insert it as the first child of the image element.

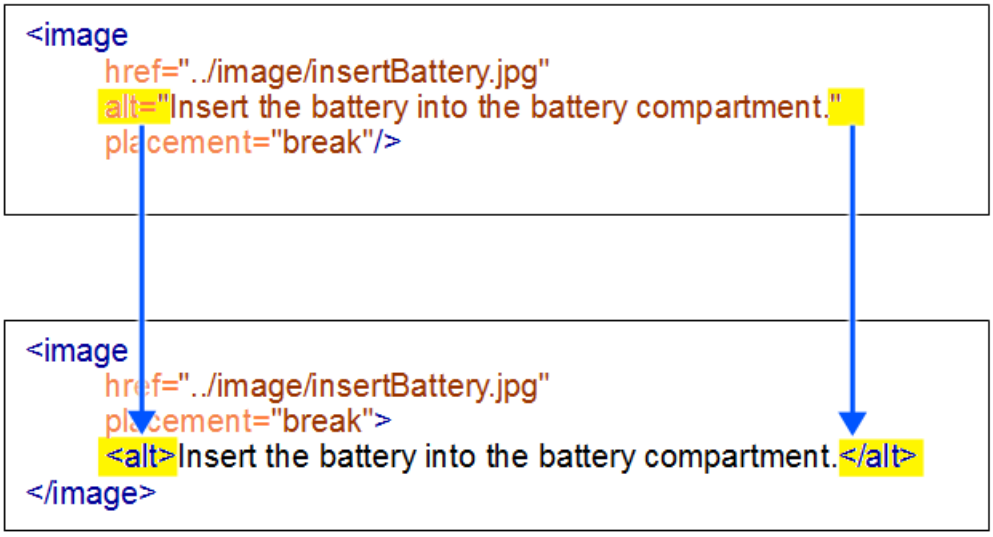


Figure 604: Example: Custom XML Refactoring Operation

We can use an XSLT stylesheet to implement the new custom XML refactoring operation. The second requirement is an *XML Refactoring operation descriptor file* that contains the path to the XSLT stylesheet.

Example of an XSLT Script for Creating a Custom Operation to *Convert an Attribute to an Element*

The XSLT stylesheet does the following:

- Iterates over all elements from the document that have the specified local name and namespace.
- Finds the attribute that will be converted to an element.
- Adds the new element as the first child of the parent element.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  exclude-result-prefixes="xs"
  xmlns:xr="http://www.oxygenxml.com/ns/xmlRefactoring"
  version="2.0">

  <xsl:import
href="http://www.oxygenxml.com/ns/xmlRefactoring/resources/commons.xsl"/>

  <xsl:param name="element_localName" as="xs:string" required="yes"/>
  <xsl:param name="element_namespace" as="xs:string" required="yes"/>
  <xsl:param name="attribute_localName" as="xs:string" required="yes"/>
  <xsl:param name="attribute_namespace" as="xs:string" required="yes"/>
  <xsl:param name="new_element_localName" as="xs:string" required="yes"/>
  <xsl:param name="new_element_namespace" as="xs:string" required="yes"/>

  <xsl:template match="node() | @*">
      <xsl:copy>
          <xsl:apply-templates select="node() | @*"/>
      </xsl:copy>
  </xsl:template>
  <xsl:template match="//*[xr:check-local-name($element_localName, ., true())
     and
        xr:check-namespace-uri($element_namespace, .)]">

      <xsl:variable name="attributeToConvert"
         select="@*[xr:check-local-name($attribute_localName, ., true())
     and
        xr:check-namespace-uri($attribute_namespace, .)]"/>

      <xsl:choose>
          <xsl:when test="empty($attributeToConvert)">
              <xsl:copy>
                  <xsl:apply-templates select="node() | @*"/>
              </xsl:copy>
          </xsl:when>
      <xsl:otherwise>
          <xsl:copy>
              <xsl:for-each select="@*[empty(. intersect $attributeToConvert)]">
                   <xsl:copy-of select="."/>
              </xsl:for-each>
                <!-- The new element namespace -->
```

```
                <xsl:variable name="nsURI" as="xs:string">
                    <xsl:choose>
            <xsl:when test="$new_element_namespace eq $xr:NO-NAMESPACE">
                            <xsl:value-of select="''"/>
                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:value-of select="$new_element_namespace"/>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:variable>
          <xsl:element name="{$new_element_localName}" namespace="{$nsURI}">
                    <xsl:value-of select="$attributeToConvert"/>
                </xsl:element>
              <xsl:apply-templates select="node()"/>
         </xsl:copy>
       </xsl:otherwise>
     </xsl:choose>
   </xsl:template>
 </xsl:stylesheet>
```

Note: The XSLT stylesheet imports a module library that contains utility functions and variables. The location of this module is resolved via an *XML Catalog* set in the XML Refactoring *framework*.

Example of an Operation Descriptor File That References the XSLT Stylesheet for Creating a Custom Operation to *Convert an Attribute to an Element*

After you have developed the XSLT stylesheet, you have to create an XML Refactoring operation descriptor. This descriptor is used by the application to load the operation details such as name, description, or parameters.

```
<?xml version="1.0" encoding="UTF-8"?>

<refactoringOperationDescriptor
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://www.oxygenxml.com/ns/xmlRefactoring"
 id="convert-attribute-to-element"
 name="Convert attribute to element">
 <description>Converts the specified attribute to an element.
           The new element will be inserted as first child of the attribute's
           parent element.</description>
 <!-- For the XSLT stylesheet option uncomment the following line and comment
           the line referring the XQuery Update script -->
 <!-- <script type="XSLT" href="convert-attribute-to-element.xsl"/> -->
 <script type="XQUERY_UPDATE" href="convert-attribute-to-element.xq"/>
  <parameters>
   <description>Specify the attribute to be converted to element.</description>
    <section label="Parent element">
     <elementParameter id="elemID">
      <localName label="Name" name="element_localName" allowsAny="true">
       <description>Local name of the parent element.</description>
        </localName>
       <namespace label="Namespace" name="element_namespace" allowsAny="true">
         <description>Local name of the parent element</description>
       </namespace>
     </elementParameter>
    </section>
    <section label="Attribute">
     <attributeParameter dependsOn="elemID">
      <localName label="Name" name="attribute_localName">
       <description>Name of the attribute to be converted.</description>
      </localName>
     <namespace label="Namespace" name="attribute_namespace" allowsAny="true">
       <description>Namespace of the attribute to be converted.</description>
     </namespace>
     </attributeParameter>
    </section>
    <section label="New element">
        <elementParameter>
           <localName label="Name" name="new_element_localName">
               <description>The name of the new element.</description>
           </localName>
           <namespace label="Namespace" name="new_element_namespace">
               <description>The namespace of the new element.</description>
           </namespace>
        </elementParameter>
    </section>
  </parameters>
</refactoringOperationDescriptor>
```

Note: If you are using an XSLT file, the line with the `script` element would look like this:

```
 <script type="XSLT" href="convert-attribute-to-element.xsl"/>
```

Results

After you have created these files, copy them into a folder *scanned by Oxygen XML Editor when it loads the custom operation*. When the XML Refactoring tool is started again, you will see the created operation.

Since various parameters can be specified, this custom operation can also be used for other similar tasks. The following image shows the parameters that can be specified in our example of the custom operation to convert an attribute to an element:

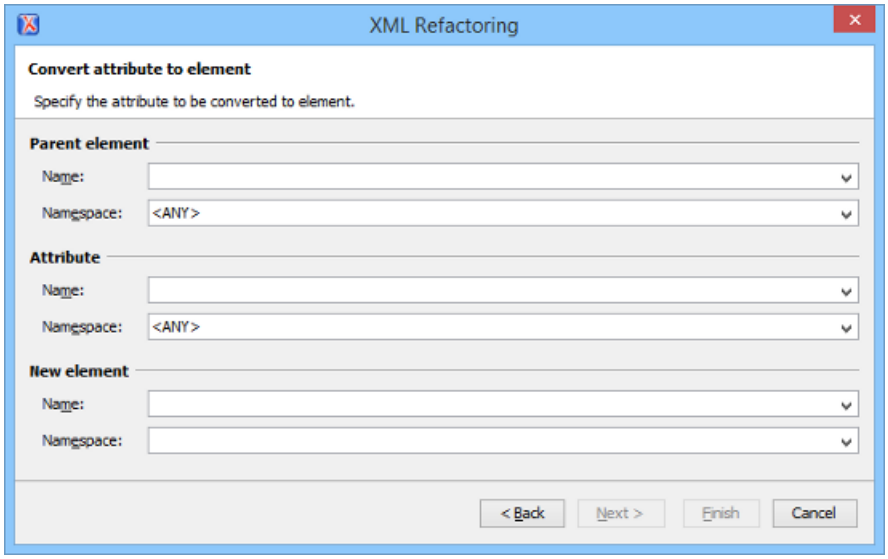


Figure 605: Example: XML Refactoring Wizard for a Custom Operation

Using Saxon Extension Functions to Allow Custom Refactoring Operations to Read and Modify Content Outside the Root Node

One advantage to using an XSLT stylesheet is that there is limitation when using an *XQuery Update script* in that refactoring operations can only be performed on *comments* or *processing instructions* that are inside the root element. Thus, using the XQuery method, *comments* or *processing instructions* that are in any node before or after the root element cannot be modified by an XML Refactoring operation.

The XSLT stylesheet method offers a work around to this limitation through the use of some Saxon extension functions.

To illustrate the use of these functions, consider the following sample XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- comment before root -->
<?pi before root ?>
<root>
    <child></child>
</root>
<!-- comment after root -->
<?pi after root ?>
```

The following Saxon extension functions can be used to read and modify content outside the root node:

Note: They belong to the `http://www.oxygenxml.com/ns/xmlRefactoring/functions` namespace.

- **get-content-after-root()** - Returns the content after root as `xs:string`.

  For the XML above, the call of this function will return the following string value:

  ```
  <!-- comment after root -->
  <?pi after root ?>
  ```

- **set-content-after-root(xs:string)** - Updates the content that will be serialized in the refactored document after the root node.

The function call `set-content-after-root('<!-- Inserted comment -->')` will result in replacing the nodes after the root element with the comment passed as string argument. The XML document will be modified as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- comment before root -->
<?pi before root ?>
<root>
    <child></child>
</root><!-- Inserted comment -->
```

- **get-content-before-root()** - Returns the content before root as `xs:string`.

  For the XML above, the call of this function will return the following string value:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- comment before root -->
<?pi before root ?>
```

- **set-content-before-root(xs:string)** - Updates the content that will be serialized in the refactored document after the root node.

  The function call `set-content-before-root('<!-- Inserted comment -->')` will result in replacing the nodes before the root element with the comment passed as string argument. The XML document will be modified as follows:

```
<!-- Inserted comment --><root>
    <child></child>
</root>
<!-- comment after root -->
<?pi after root ?>
```

XSLT Example:

To process content after the root node, the XSLT would look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" exclude-result-prefixes="xs"
    xmlns:xrf="http://www.oxygenxml.com/ns/xmlRefactoring/functions" version="3.0">
    <xsl:template match="/">
        <!-- The comment content that will be inserted after the root element -->
        <xsl:variable name="commentAsText">&lt;!-- COMMENT ADDED FROM XR OPERATION-->
        </xsl:variable>
        <!-- Retrieve the content after the root element as is -->
        <xsl:variable name="after-root-content" as="xs:string"
                            select="xrf:get-content-after-root()"/>

        <xsl:variable name="processedContent"
                            select="concat($after-root-content, $commentAsText)"/>

        <!-- Update the content after the root element -->
        <xsl:value-of select="xrf:set-content-after-root($processedContent)"/>

        <xsl:apply-templates/>
    </xsl:template>

    <xsl:template match="node() | @*">
        <xsl:copy>
            <xsl:apply-templates select="node() | @*"/>
        </xsl:copy>
    </xsl:template>
</xsl:stylesheet>
```

Note: The above XSLT retrieves the nodes after the root element as string, appends a new comment, and then sets back the updated content into the XML document.

Storing and Sharing Refactoring Operations

Oxygen XML Editor scans the following locations when looking for XML Refactoring operations to provide flexibility:

- A `refactoring` folder, created inside a directory that is associated to a *framework* you are customizing.
- A folder that you specify in the ***Load additional refactoring operations from*** *text box* in the ***XML Refactoring*** *preferences page*.

Note: If you share a project with your team, you can also share the custom operation by doing the following: Then by doing the following:

1. Save the custom operation in a folder that is part of your project.
2. Switch the **XML Refactoring** option page to *project level*:
   a. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to ***XML** > **XML Refactoring***.
   b. Select ***Project Options*** at the bottom of the dialog box.
3. In the ***Load additional refactoring operations from** text box*, use the *${pd} editor variable* so that the folder path is declared relative to the project.

- A folder specified by the *XML Refactoring Operations Plugin Extension*.
- The `refactoring` folder from the Oxygen XML Editor installation directory (*[OXYGEN_INSTALL_DIR]*/`refactoring/`).

Sharing Custom Refactoring Operations

The purpose of Oxygen XML Editor scanning multiple locations for the XML Refactoring operations is to provide more flexibility for developers who want to share the refactoring operations with the other team members. Depending on your particular use case, you can attach the custom refactoring operations to other resources, such as *framework* or projects.

After storing custom operations, you can share them with other users by sharing the resources.

Localizing XML Refactoring Operations

Oxygen XML Editor includes localization support for the XML refactoring operations.

The translation keys for the built-in refactoring operations are located in *[OXYGEN_INSTALL_DIR]*/`refactoring/i18n/translation.xml`.

The localization support is also available for custom refactoring operations. The following information can be translated:

- The operation `name`, `description`, and `category`.
- The `description` of the `parameters` element.
- The `label`, `description`, and `possibleValues` for each `parameter`.

Translated refactoring information uses the following form:

```
${i18n(translation_key)}
```

Oxygen XML Editor scans the following locations to find the `translation.xml` files that are used to load the translation keys:

- A `refactoring/i18n` folder, created inside a directory that is associated to a customized *framework*.
- A `i18n` folder, created inside a directory that is associated to a customized *framework*.
- An `i18n` folder inside any specified folder. In this case, you need to *open the **Preferences** dialog box **(Options** > **Preferences)***, go to **XML** > **XML Refactoring**, and specify the folder in the **Load additional refactoring operations from** text box.
- An `i18n` folder located in directories specified through the *XML Refactoring Operations Plugin Extension*.
- The `refactoring/i18n` folder from the Oxygen XML Editor installation directory (*[OXYGEN_INSTALL_DIR]*`/refactoring/i18n`).

Example: Refactoring Operation Descriptor File with *i18n* Support

```
<?xml version="1.0" encoding="UTF-8"?>

<refactoringOperationDescriptor
    xmlns="http://www.oxygenxml.com/ns/xmlRefactoring" id="remove_text_content"
           name="${i18n(Remove_text_content)}">
    <description>${i18n(Remove_text_content_description)}</description>
    <script type="XQUERY_UPDATE" href="remove_text_content.xq"/>
    <parameters>
        <description>${i18n(parameters_description)}</description>
        <parameter label="${i18n(Element_name)}" name="element_localName"
                      type="NC_NAME">
            <description>${i18n(Element_name_descriptor)}</description>
```

```
            <possibleValues>
                <value default="true" name="value1">${i18n(value_1)}</value>
                <value name="value2">${i18n(value_2)}</value>
            </possibleValues>
        </parameter>
    </parameters>
</refactoringOperationDescriptor>
```

Generating Sample XML Files

Oxygen XML Editor offers support to generate sample XML files both from XML schema 1.0 and XML schema 1.1, depending on the XML schema version set in ***XML Schema*** *preferences page*.

To generate sample XML files from an XML Schema, use the **Generate Sample XML Files** action from the **Tools** menu. This action is also available in the contextual menu of the schema *Design mode*. The action opens the **Generate Sample XML Files** dialog box that allows you to configure a variety of options for generating the files.

For more information about this tool, watch our video demonstration about generating sample XML files at *https://www.oxygenxml.com/demo/Generate_Sample_XML_Files.html*.

The **Generate Sample XML Files** dialog box contains three tabs with various configurable options. Default values for these options can be set in the ***Sample XML Files Generator*** *preferences page*. You can also run the tool from the command line using exported options.

Schema Tab (Generate Sample XML Files Tool)

The **Generate Sample XML Files** tool includes a dialog box that allows you to configure a variety of options for generating the XML files. The first set of options are found in the **Schema** tab.

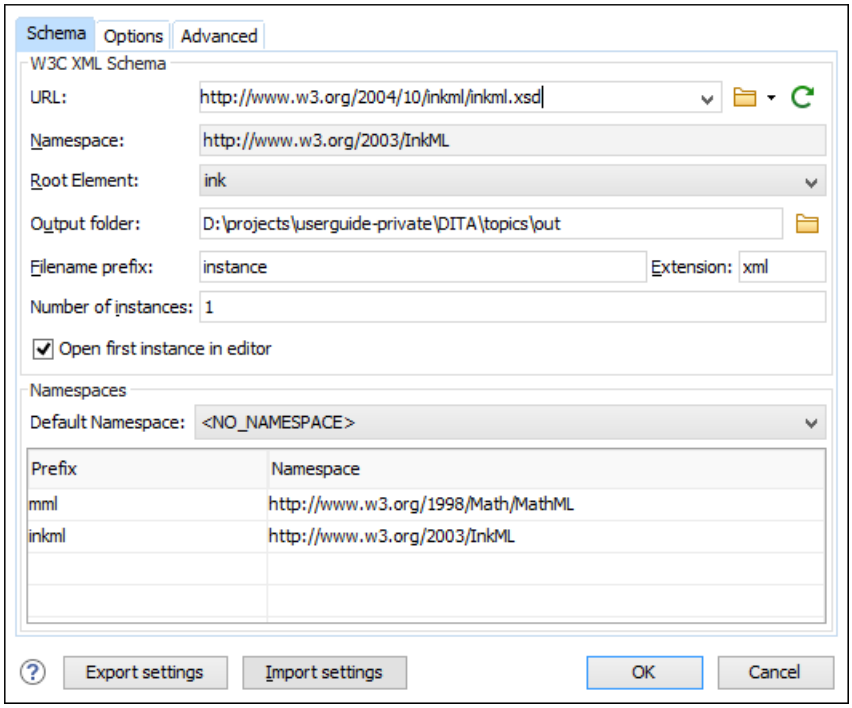


Figure 606: Generate Sample XML Files Dialog Box (Schema Tab)

This tab includes the following options:

URL

Specifies the URL of the Schema location. You can specify the path by using the text field, the history drop-down menu, or the browsing tools in the **Browse** drop-down list.

Namespace

Displays the namespace of the selected schema.

Root Element

After the schema is selected, this drop-down menu is populated with all root candidates gathered from the schema. Choose the root of the output XML documents.

Output folder

Path to the folder where the generated XML instances will be saved.

Filename prefix and Extension

You can specify the prefix and extension for the file name that will be generated. Generated file names have the following format: `prefixN.extension`, where N represents an incremental number from 0 up to the specified ***Number of instances***.

Number of instances

The number of XML files to be generated.

Open first instance in editor

When selected, the first generated XML file is opened in the editor.

Namespaces section

You can specify the **Default Namespace**, as well as the prefixes for the namespaces.

Export settings

Use this button to save the current settings for future use.

Import settings

Use this button to load previously exported settings.

You can click **OK** at any point to generate the sample XML files.

Options Tab (Generate Sample XML Files Tool)

The **Generate Sample XML Files** tool includes a dialog box that allows you to configure a variety of options for generating the XML files. The **Options** tab allows you to set specific options for namespaces and elements.

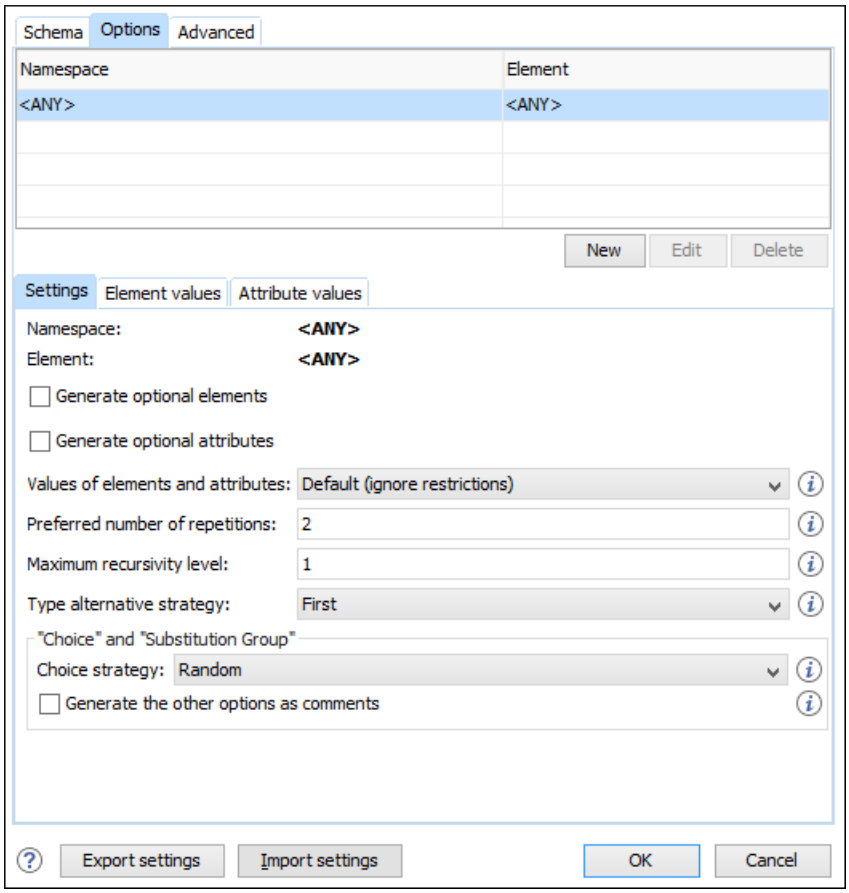


Figure 607: Generate Sample XML Files Dialog Box (Options Tab)

This tab includes the following options:

Namespace / Element table

Allows you to set a namespace for each element name that appears in an XML document instance. The following prefix-to-namespace associations are available:

- All elements from all namespaces (`<ANY>` - `<ANY>`). This is the default setting.
- All elements from a specific namespace.
- A specific element from a specific namespace.

Settings subtab

Namespace

Displays the namespace specified in the table at the top of the dialog box.

Element

Displays the element specified in the table at the top of the dialog box.

Generate optional elements

When selected, all elements are generated, including the optional ones (having the `minOccurs` attribute set to 0 in the schema).

Generate optional attributes

When selected, all attributes are generated, including the optional ones (having the `use` attribute set to `optional` in the schema).

Values of elements and attributes

Controls the content of generated attribute and element values. The following choices are available:

- **None** - No content is inserted.
- **Default** - Inserts a default value depending of data type descriptor of the particular element or attribute. The default value can be either the data type name or an incremental name of the attribute or element

(according to the global option from the **Sample XML Files Generator** preferences page). Note that type restrictions are ignored when this option is selected. For example, if an element is of a type that restricts an **xs:string** with the **xs:maxLength** facet to allow strings with a maximum length of 3, the XML instance generator tool may generate string element values longer than 3 characters.

- **Random** - Inserts a random value depending of data type descriptor of the particular element or attribute.

  Important: If all of the following are true, the **Generate Sample XML Files** tool outputs invalid values:

  - At least one of the restrictions is a `regexp`.
  - The value generated after applying the regexp does not match the restrictions imposed by one of the facets.

Preferred number of repetitions

Allows you to set the preferred number of repeating elements related to `minOccurs` and `maxOccurs` facets defined in the XML Schema.

- If the value set here is between `minOccurs` and `maxOccurs`, then that value is used.
- If the value set here is less than `minOccurs`, then the `minOccurs` value is used.
- If the value set here is greater than `maxOccurs`, then `maxOccurs` is used.

Maximum recursion level

If a recursion is found, this option controls the maximum allowed depth of the same element.

Type alternative strategy

Used for the `xs:alternative` element from XML Schema 1.1. The possible strategies are:

- **First** - The first valid alternative type is always used.
- **Random** - A random alternative type is used.

Choice strategy

Used for `xs:choice` or `substitutionGroup` elements. The possible strategies are:

- **First** - The first branch of `xs:choice` or the head element of `substitutionGroup` is always used.
- **Random** - A random branch of `xs:choice` or a substitute element or the head element of a `substitutionGroup` is used.

Generate the other options as comments

If selected, generates the other possible choices or substitutions (for `xs:choice` and `substitutionGroup`). These alternatives are generated inside comments groups so you can uncomment and use them later. Use this option with care (for example, on a restricted namespace and element) as it may generate large result files.

Element values subtab

Allows you to add values that are used to generate the content of elements. If there are multiple values, then the values are used in a random order.

Attribute values subtab

Allows you to add values that are used to generate the content of attributes. If there are multiple values, then the values are used in a random order.

Export settings

Use this button to save the current settings for future use.

Import settings

Use this button to load previously exported settings.

You can click **OK** at any point to generate the sample XML files.

Advanced Tab (Generate Sample XML Files Tool)

The **Generate Sample XML Files** tool includes a dialog box that allows you to configure a variety of options for generating the XML files. The **Advanced** tab allows you to set some options in regards to output values and performance.

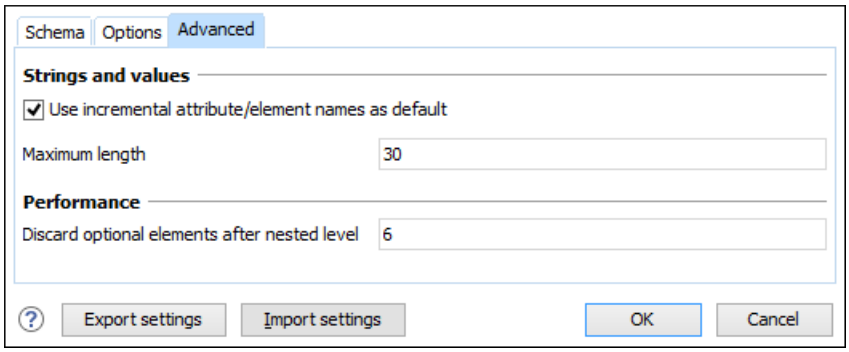


Figure 608: Generate Sample XML Files Dialog Box (Advanced Tab)

This tab includes the following options:

Use incremental attribute / element names as default

If selected, the value of an element or attribute starts with the name of that element or attribute. For example, for an a element the generated values are: a1, a2, a3, and so on. If not selected, the value is the name of the type of that element / attribute (for example: `string`, `decimal`, etc.)

Maximum length

The maximum length of string values generated for elements and attributes.

Discard optional elements after nested level

The optional elements that exceed the specified nested level are discarded. This option is useful for limiting deeply nested element definitions that can quickly result in very large XML documents.

Export settings

Use this button to save the current settings for future use.

Import settings

Use this button to load previously exported settings.

Running the Generate Sample XML Files Tool from the Command Line

The **Generate Sample XML Files** tool can be also used from command line by running the script called `xmlGenerator.bat` (on Windows) / `xmlGenerator.sh` (on Mac OS X / Unix / Linux) located in the Oxygen XML Editor installation folder. The parameters can be set in the dialog box, exported to an XML file on disk with the **Export settings** button, and then reused from command line. With the exported settings file, you can generate the same XML instances from the command line as from the dialog box. For example:

```
xmlGenerator.bat path_of_CFG_file
```

The script can be integrated in an external batch process launched from the command line. The command line parameter of the script is the relative path to the exported XML settings file. The files specified with relative paths in the exported XML settings will be made absolute relative to the folder where the script is run.

The following example shows such an XML configuration file:

```
<settings>
    <schemaSystemId>http://www.w3.org/2001/XMLSchema.xsd</schemaSystemId>
    <documentRoot>schema</documentRoot>
    <outputFolder>D:\projects\output</outputFolder>
    <filenamePrefix>instance</filenamePrefix>
    <filenameExtension>xml</filenameExtension>
    <noOfInstances>1</noOfInstances>
    <openFirstInstance>true</openFirstInstance>
    <defaultNamespace>&lt;NO_NAMESPACE></defaultNamespace>
    <element namespace="&lt;ANY>" name="&lt;ANY>">
        <generateOptionalElements>false</generateOptionalElements>
        <generateOptionalAttributes>false</generateOptionalAttributes>
        <valuesForContentType>DEFAULT</valuesForContentType>
        <preferredNumberOfRepetitions>2</preferredNumberOfRepetitions>
        <maximumRecursivityLevel>1</maximumRecursivityLevel>
        <choicesAndSubstitutions strategy="RANDOM"
                generateOthersAsComments="false"/>
        <attribute namespace="&lt;ANY>"
                name="&lt;ANY>">
            <attributeValue>attrValue1</attributeValue>
            <attributeValue>attrValue2</attributeValue>
        </attribute>
```

```
    </element>
    <element namespace="&lt;NO_NAMESPACE>"
            name="&lt;ANY>">
        <generateOptionalElements>true</generateOptionalElements>
        <generateOptionalAttributes>true</generateOptionalAttributes>
        <valuesForContentType>DEFAULT</valuesForContentType>
        <preferredNumberOfRepetitions>2</preferredNumberOfRepetitions>
        <maximumRecursivityLevel>1</maximumRecursivityLevel>
        <choicesAndSubstitutions strategy="RANDOM"
                generateOthersAsComments="true"/>
        <elementValue>value1</elementValue>
        <elementValue>value2</elementValue>
        <attribute namespace="&lt;ANY>"
                name="&lt;ANY>">
            <attributeValue>attrValue1</attributeValue>
            <attributeValue>attrValue2</attributeValue>
        </attribute>
    </element>
</settings>
```

Converting Schema to Another Schema Language

The **Generate/Convert Schema** tool allows you to convert a DTD or Relax NG (full or compact syntax) schema or a set of XML files to an equivalent XML Schema, DTD or Relax NG (full or compact syntax) schema. Where perfect equivalence is not possible due to limitations of the target language, Oxygen XML Editor generates an approximation of the source schema. Oxygen XML Editor uses *Trang multiple format converter* to perform the actual schema conversions.

To use this tool, select the **Generate/Convert Schema (Alt + Shift + C (Command + Alt + C on OS X))** action from the **Tools** menu or from the **Open with** submenu when invoking the contextual menu in the ***Project** view*. This action opens the **Generate/Convert Schema** dialog box that allows you to configure various options for conversion.

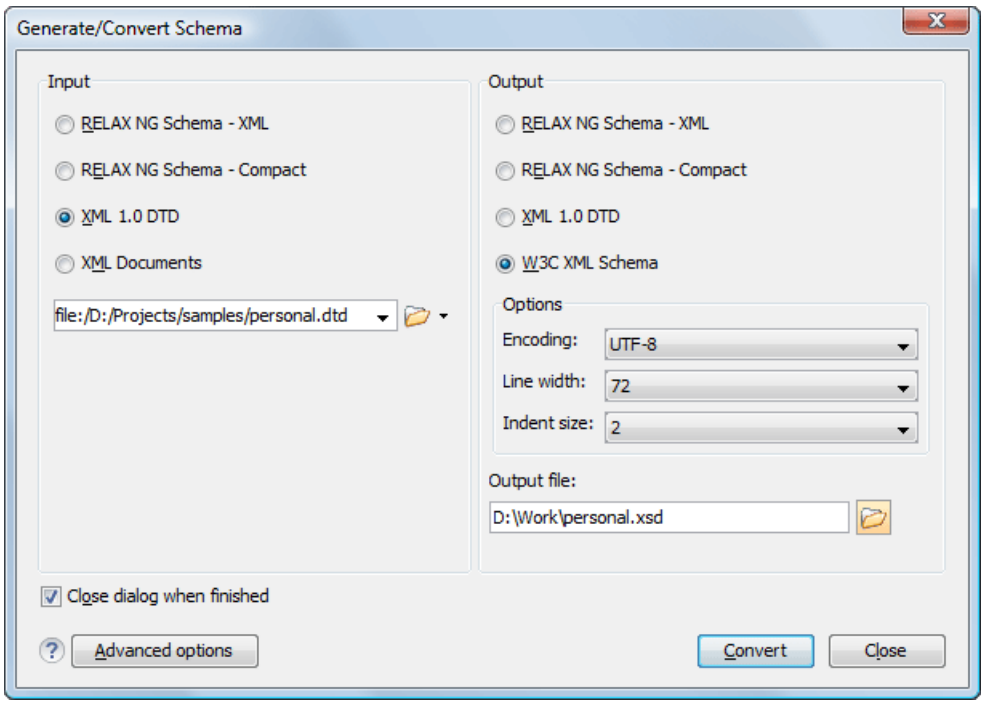


Figure 609: Generate/Convert Schema Dialog Box

The **Generate/Convert Schema** dialog box includes the following options:

Input section

Allows you to select the language of the source schema. If the conversion is based on a set of XML files, rather than just a single XML file, select the **XML Documents** option and use the file selector to add the XML files involved in the conversion.

Output section

Allows you to select the language of the target schema.

Options

You can choose the **Encoding**, the maximum **Line width**, and the **Indent size** (in number of spaces) for one level of indentation.

Output file

Specifies the path for the output file that will be generated.

Close dialog when finished

If you deselect this option, the dialog box will remain opened after the conversion so that you can easily continue to convert more files.

Advanced options

If you select **XML 1.0 DTD** for the input, you can click this button to access more advance options to further fine-tune the conversion. The following advanced options are available:

XML 1.0 DTD Input section

These options apply to the source DTD:

- **xmlns** - Specifies the default namespace, that is the namespace used for unqualified element names.
- **attlist-define** - Specifies how to construct the name of the definition representing an attribute list declaration from the name of the element. The specified value must contain exactly one percent character. This percent character is replaced by the name of element (after colon replacement) and the result is used as the name of the definition.
- **colon-replacement** - Replaces colons in element names with the specified chars when constructing the names of definitions used to represent the element declarations and attribute list declarations in the DTD.
- **any-name** - Specifies the name of the definition generated for the content of elements declared in the DTD as having a content model of ANY.
- **element-define** - Specifies how to construct the name of the definition representing an element declaration from the name of the element. The specified value must contain exactly one percent character. This percent character is replaced by the name of element (after colon replacement) and the result is used as the name of the definition.
- **annotation-prefix** - Default values are represented using an annotation attribute `prefix:defaultValue` where prefix is the specified value and is bound to `http://relaxng.org/ns/compatibility/annotations/1.0` as defined by the RELAX NG DTD Compatibility Committee Specification. By default, the conversion engine will use a for prefix unless that conflicts with a prefix used in the DTD.
- **inline-attlist** - Instructs the application not to generate definitions for attribute list declarations, but instead move attributes declared in attribute list declarations into the definitions generated for element declarations. This is the default behavior when the output language is XSD.
- **strict-any** - Preserves the exact semantics of *ANY* content models by using an explicit choice of references to all declared elements. By default, the conversion engine uses a wildcard that allows any element
- **generate-start** - Specifies whether or not the conversion engine should generate a start element. DTD's do not indicate what elements are allowed as document elements. The conversion engine assumes that all elements that are defined but never referenced are allowed as document elements.
- **xmlns mappings** table - Each row specifies the prefix used for a namespace in the input schema.

W3C XML Schema Output section

This section is available if you select **W3C XML Schema** for the output.

- **disable-abstract-elements** - Disables the use of abstract elements and substitution groups in the generated XML Schema. This can also be controlled using an annotation attribute.
- **any-process-contents** - One of the values: strict, lax, skip. Specifies the value for the `processContents` attribute of any elements. The default is skip (corresponding to RELAX NG semantics) unless the input format is DTD, in which case the default is strict (corresponding to DTD semantics).

- **any-attribute-process-contents** - Specifies the value for the `processContents` attribute of `anyAttribute` elements. The default is skip (corresponding to RELAX NG semantics).

Converting Database to XML Schema

Oxygen XML Editor includes a tool that allows you to create an XML Schema from the structure of a database.

To convert a database structure to an XML Schema, use the following procedure:

1. Select the **Convert DB Structure to XML Schema** action from the **Tools** menu.

   Result: The **Convert DB Structure to XML Schema** dialog box is opened and your current database connections are displayed in the **Connections** section.
2. If the database source is not listed, click the **Configure Database Sources** button to open the ***Data Sources*** *preferences page* where you can configure data sources and connections.
3. In the **Format for generated schema** section, select one of the following formats:
   - **Flat schema** - A flat structure that resembles a tree-like view of the database without references to elements.
   - **Hierarchical schema** - Display the table dependencies visually, in a type of tree view where dependent tables are shown as indented child elements in the content model. Select this option if you want to configure the database columns of the tables to be converted.
4. Click **Connect**.

   Result: The database structure is listed in the **Select database tables** section according to the format you chose.
5. Select the database tables that you want to be included in the XML Schema.
6. If you selected **Hierarchical schema** for the format, you can configure the database columns.
   a. Select the database column you want to configure.
   b. In the **Criterion** section you can choose to convert the selected database column as an **Element**, **Attribute**, or to be **Skipped** in the resulting XML Schema.
   c. You can also change the name of the selected database column by changing it in the **Name** text field.
7. Click **Generate XML Schema**.

Result: The database structure is converted to an XML Schema and it is opened for viewing and editing.

Flatten an XML Schema

You can organize an XML schema linked by `xs:include` and `xs:import` statements on several levels. In some cases, working on such a schema as if it were a single file is more convenient than working on multiple files separately. The **Flatten Schema** operation allows you to flatten an entire hierarchy of XML schemas. Starting with the main XML schema, Oxygen XML Editor calculates its hierarchy by processing the `xs:include` and `xs:import` statements.

The **Flatten Schema** action is available from the **Tools** menu or the contextual menu in **Text** mode. The action opens the **Flatten Schema** dialog box that allows you to configure the operation.

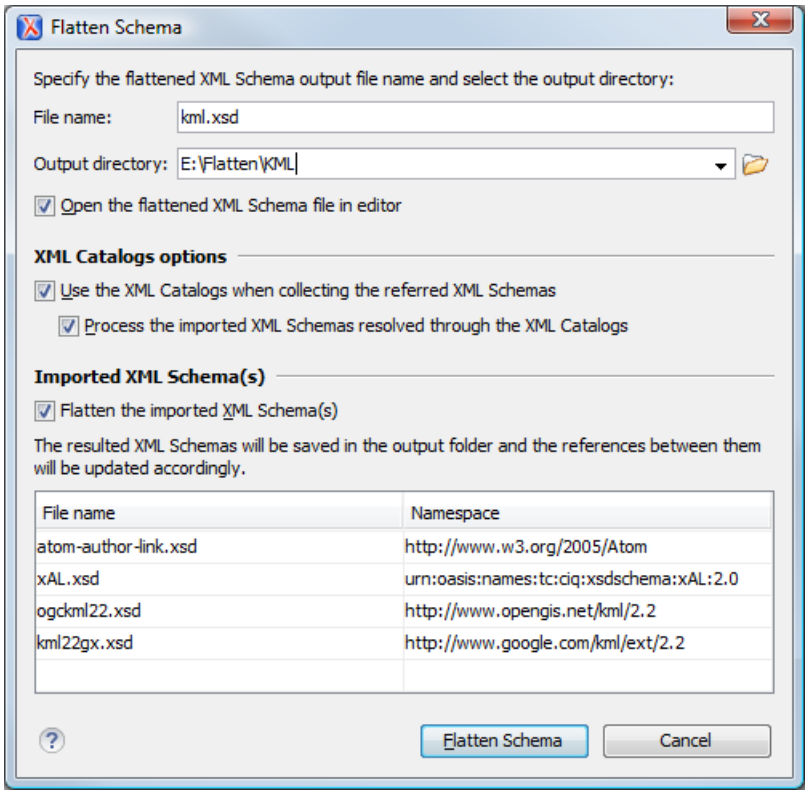


Figure 610: Flatten Schema Dialog Box

For the main schema file and for each imported schema, a new flattened schema is generated in the specified output folder. These schemas have the same name as the original ones.

Note: If necessary, the operation renames the resulted schemas to avoid duplicated file names.

A flattened XML schema is obtained by recursively adding the components of the included schemas into the main one. This means Oxygen XML Editor replaces the `xs:include`, `xs:redefine`, and `xs:override` elements with the ones coming from the included files.

Options in the Flatten Schema Dialog Box

The following options are available in the **Flatten Schema** dialog box:

File name

The name of the output file.

Output directory

The path of the output directory where the flattened schema file will be saved.

Open the flattened XML Schema file in editor

Opens the main flattened schema in the editing area after the operation completes.

Use the XML Catalogs when collecting the referenced XML Schemas

Enables the imported and included schemas to be resolved through the available *XML Catalogs*.

Note: Changing this option triggers the recalculation of the dependencies graph for the main schema.

Process the imported XML Schemas resolved through the XML Catalogs

Specifies whether or not the imported schemas that were resolved through an *XML Catalog* are also processed.

Flatten the imported XML Schema(s)

Specifies whether or not the imported schemas are flattened.

Note: For the schemas skipped by the flatten operation, no files are created in the output folder and the corresponding import statements remain unchanged.

Flatten Schema from the Command Line

The **Flatten Schema** tool can be also ran from command line by using the following command:

- `flattenSchema.bat` on Windows
- `sh flattenSchemaMac.sh` on OS X
- `sh flattenSchema.sh` on Unix/Linux

The command line accepts the following parameters:

- `-in:inputSchemaURL` - The input schema URL.
- `-outDir:outputDirectory` - The directory where the flattened schemas should be saved.
- `-flattenImports:<boolean_value>` - Controls whether or not the imported XML Schemas should be flattened. The default value `true`.
- `-useCatalogs:<boolean_value>` - Controls if the references to other XML Schemas should be resolved through the available *XML Catalogs*. The default value `false`.
- `-flattenCatalogResolvedImports:<boolean_value>` - Controls whether or not the imported schemas that were resolved through the *XML Catalogs* should be flattened. The default value is `true`.

  Note: This option is used only when `-useCatalogs` is set to `true`.
- `-verbose` - Provides information about the current flatten XML Schema operation.
- `--help | -help | --h | -h` - Prints the available parameters for the operation.

Example: Command Line Example for Windows

```
flattenSchema.bat -in:http://www.w3.org/MarkUp/SCHEMA/xhtml11.xsd
   -outDir:mySchemas/flattened/xhtml -flattenImports:true -useCatalogs:true
      -flattenCatalogResolvedImports:true -verbose
```

Example: Command Line Example for OS X

```
sh flattenSchemaMac.sh -in:http://www.w3.org/MarkUp/SCHEMA/xhtml11.xsd
    -outDir:mySchemas/flattened/xhtml -flattenImports:true -useCatalogs:true
       -flattenCatalogResolvedImports:true -verbose
```

Example: Command Line Example for Unix/Linux

```
sh flattenSchema.sh -in:http://www.w3.org/MarkUp/SCHEMA/xhtml11.xsd
     -outDir:mySchemas/flattened/xhtml -flattenImports:true -useCatalogs:true
        -flattenCatalogResolvedImports:true -verbose
```

XML to JSON Converter

Oxygen XML Editor includes a useful and simple tool for converting XML files to JSON. It can be found in the **Tools** menu.

To convert an XML document to JSON, follow these steps:

1. Select the **XML to JSON** action from the **Tools** menu.

   The **XML to JSON** dialog box is displayed:

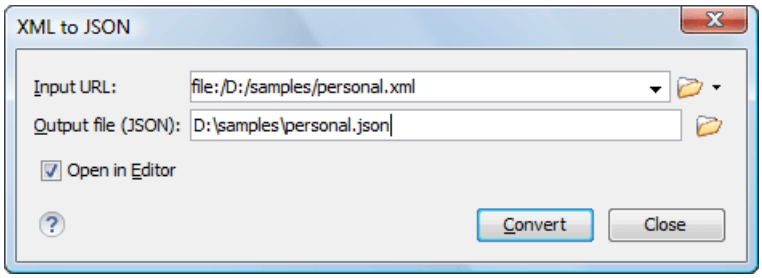


   Figure 611: XML to JSON Dialog Box

2. Choose or enter the **Input URL** of the XML document.
3. Choose the path of the **Output file** that will contain the conversion JSON result.

4. Select the **Open in Editor** option to open the JSON result of the conversion in the Oxygen XML Editor JSON Editor.
5. Click the **Convert** button.

The original XML document is now converted to a JSON document.

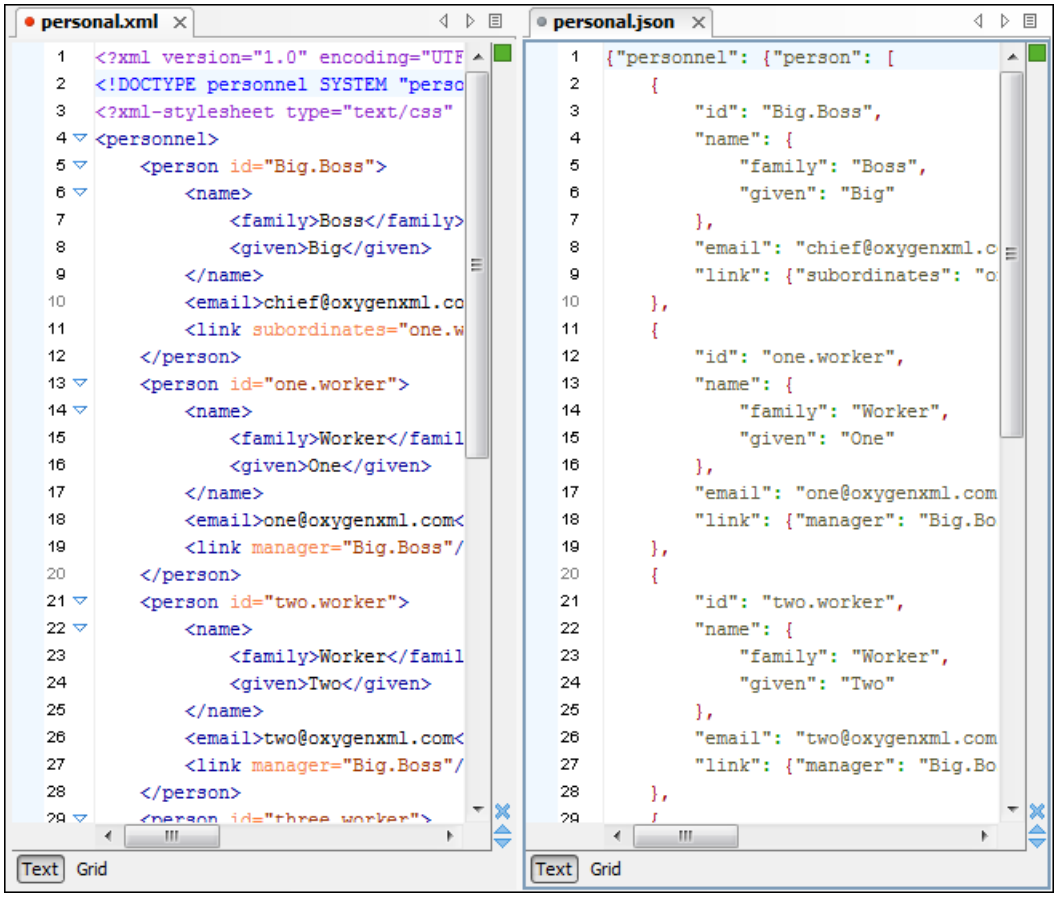


Figure 612: Example: XML to JSON Operation Result

Compiling an XSL Stylesheet for Saxon

As of Saxon 9.7, it is possible to export a compiled form of a stylesheet as an XML file (called a *stylesheet export file*). Oxygen XML Editor includes a simple tool called **Compile XSL Stylesheet for Saxon** that does this for you. A *stylesheet export file* is also needed if you want to use the Saxon-JS product to run transformations in a browser, as in the following example:

```
<script type="text/javascript" src="SaxonJS/SaxonJS.min.js"></script>
<script>
    window.onload = function() {
      SaxonJS.transform({
         stylesheetLocation: "books.sef",
         sourceLocation: "books.xml"
         });
    }
</script>
```

The **Compile XSL Stylesheet for Saxon** tool can be found in the **Tools** menu and it compiles a specified stylesheet as an XML file (*stylesheet export file* with a file extension of `.sef`).

Selecting this tool opens the **Compile XSL Stylesheet for Saxon** dialog box that allows you to configure some options for conversion.

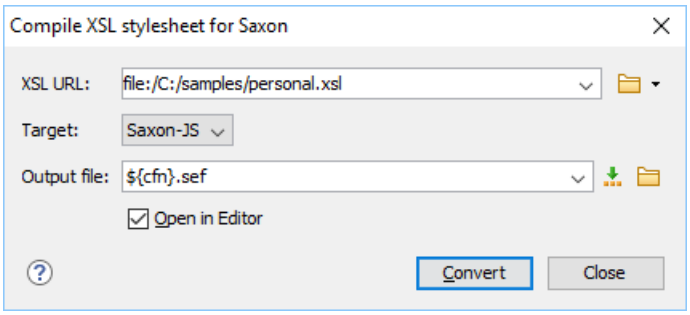


Figure 613: Compile XSL Stylesheet for Saxon Dialog Box

This dialog box includes the following options:

XSL URL

Allows you to select URL of the source XSL stylesheet. You can specify the URL by using the text field, the history drop-down, or the browsing tools in the **Browse** drop-down list.

Target

Allows you to select the type of Saxon product that the export file will be used with. You can choose **Saxon-JS**, **Saxon-EE**, **Saxon-PE**, or **Saxon-HE**.

Output file

You can specify the path where the output file will be saved by entering it in the text field, using the **Insert Editor Variables** button, or using the browsing tool button.

Open in Editor

Select this option to open the resulting *stylesheet export file* in the main Oxygen XML Editor editing pane.

Convert

Use this button to generate the *stylesheet export file* according the options selected in this dialog box.

Format and Indent (Pretty-Print) Multiple Files

Oxygen XML Editor provides support for formatting and indenting (*pretty-print*) multiple files at once. This action is available for any document in XML format, as well as for XQuery, CSS, JavaScript, and JSON documents.

To format and indent multiple files, use the **Format and Indent Files** action that is available in the contextual menu of the ***Project*** *view* or from the **Tools** menu. This opens the **Format and Indent Files** dialog box that allows you to configure options for the action.

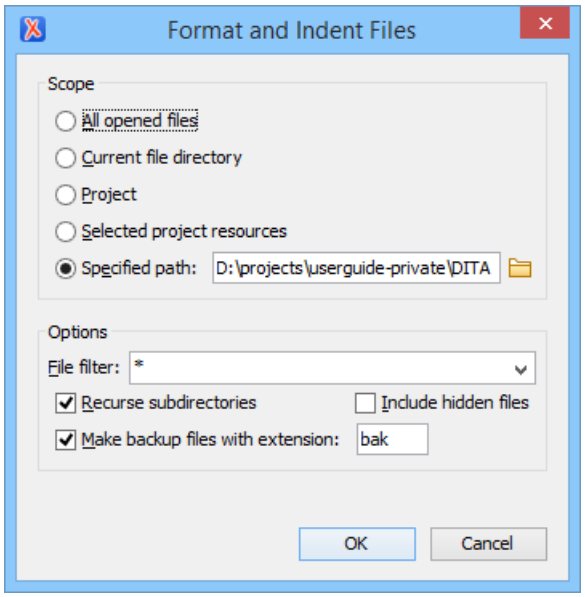


Figure 614: Format and Indent Files Dialog Box

The **Scope** section allows you choose from the following scopes:

- **All opened files** - The *pretty-print* is performed in all opened files.
- **Directory of the current file** - All the files in the folder of the current edited file.
- **Project files** - All files from the current project.
- **Selected project files** - The selected files from the current project.
- **Specified path** - the *pretty-print* is performed in the files located at a specified path.

The **Options** section includes the following options:

- **File filter** - Allow you to filter the files from the selected scope.
- **Recurse subdirectories** - When selected, the *pretty-print* is performed recursively for the specified scope. The one exception is that this option is ignored if the scope is set to **All opened files**.
- **Include hidden files** - When selected, the *pretty-print* is also performed in the hidden files.
- **Make backup files with extension** - When selected, Oxygen XML Editor makes backup files of the modified files. The default extension is `.bak`, but you can change the extension as you prefer.

Generate Documentation

Oxygen XML Editor includes a tool for generating documentation for XSLT, XML Schema, XQuery, and WSDL documents.

Generating Documentation for an XML Schema

Oxygen XML Editor can generate detailed documentation for the components of an XML Schema in HTML, PDF, DocBook, or other custom formats. You can select the components and the level of detail. The components are hyperlinked in both HTML and DocBook documents.

Note: You can generate documentation for both XML Schema version 1.0 and 1.1.

To generate documentation for an XML Schema document, select **XML Schema Documentation** from the **Tools** > **Generate Documentation** menu or from the **Generate Documentation** submenu in the contextual menu of the ***Project*** *view*.

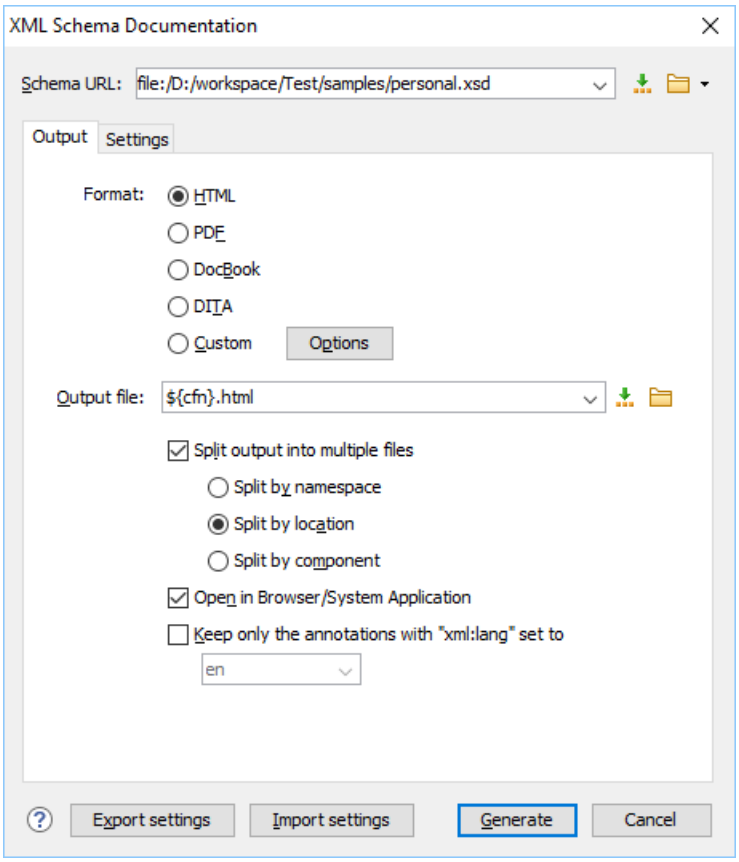


Figure 615: XML Schema Documentation Dialog Box

The **Schema URL** field of the dialog box must contain the full path to the XML Schema (XSD) file for which you want to generate documentation. The schema may be a local or a remote file. You can specify the path to the schema by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.

Output Tab

The following options are available in the **Output** tab:

- **Format** - Allows you to choose between the following formats:
  - **HTML** - The documentation is generated in *HTML output format*.
  - **PDF** - The documentation is generated in *PDF output format*.
  - **DocBook** - The documentation is generated in *DocBook output format*.
  - **DITA** - The documentation is generated in *DITA output format*.
  - **Custom** - The documentation is generated in a *custom output format*, allowing you to control the output. Click the **Options** button to open a **Custom format options** dialog box where you can specify a custom stylesheet for creating the output. There is also an option to **Copy additional resources to the output folder** and you can select the path to the additional **Resources** that you want to copy. You can also choose to keep the intermediate XML files created during the documentation process by deselecting the **Delete intermediate XML file** option.
- **Output file** - You can specify the path of the output file by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.
- **Split output into multiple files** - Instructs the application to split the output into multiple files. You can choose to split them by namespace, location, or component name.
- **Open in Browser/System Application** - Opens the result in the system application associated with the output file type. For DITA and DocBook documents, this option appears as **Open in Editor** and the result will be opened in Oxygen XML Editor (in the current editor).

Note: To set the browser or system application that will be used, *open the **Preferences** dialog box (**Options > Preferences**)*, go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.

- **Keep only the annotations with xml:lang set to** - The generated output will contain only the annotations with the `xml:lang` attribute set to the selected language. If you choose a primary language code (for example, **en** for English), this includes all its possible variations (**en-us**, **en-uk**, etc.).

Settings Tab

When you generate documentation for an XML schema you can choose what components to include in the output and the details to be included in the documentation.

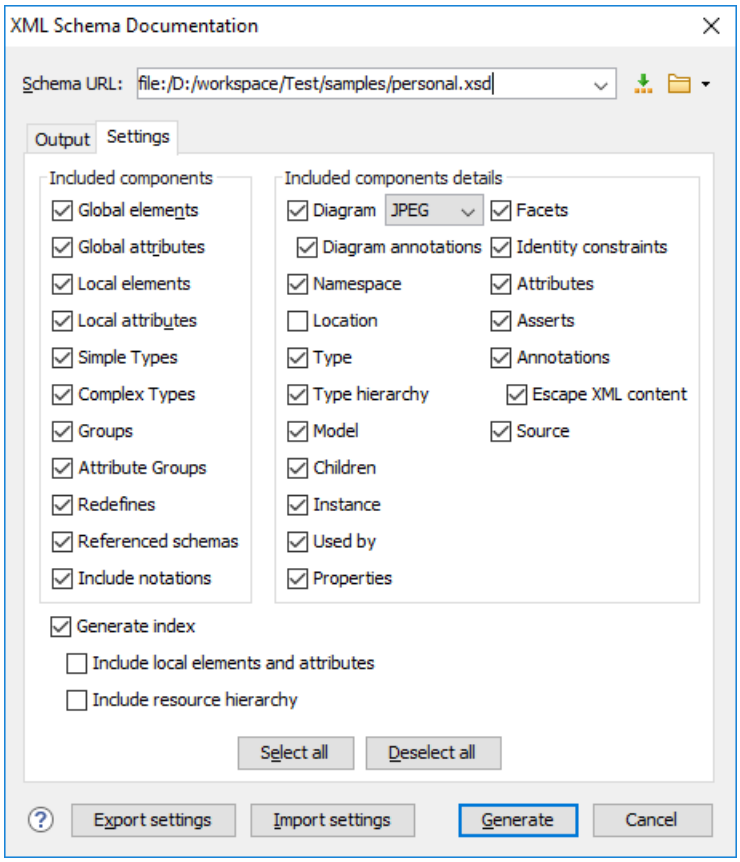


Figure 616: Settings Tab of the XML Schema Documentation Dialog Box

The **Settings** tab allows you to choose whether or not to include the following components: **Global elements**, **Global attributes**, **Local elements**, **Local attributes**, **Simple Types**, **Complex Types**, **Groups**, **Attribute Groups**, **Redefines**, **Referenced schemas**, **Include notations**.

You can choose whether or not to include the following other details:

- **Diagram** - Displays the diagram for each component. You can choose the image format (JPEG, PNG, GIF, SVG) to use for the diagram section. The generated diagrams are dependent on the options from the *Schema Design Properties* page.
  - **Diagram annotations** - This option controls whether or not the annotations of the components presented in the diagram sections are included.
- **Namespace** - Displays the namespace for each component.
- **Location** - Displays the schema location for each component.
- **Type** - Displays the component type if it is not an anonymous one.
- **Type hierarchy** - Displays the types hierarchy.
- **Model** - Displays the model (sequence, choice, all) presented in BNF form. The separator characters that are used depend upon the information item used:

    - `xs:all` - Its children will be separated by space characters.
    - `xs:sequence` - Its children will be separated by comma characters.
    - `xs:choice` - Its children will be separated by / characters.
- **Children** - Displays the list of component's children.
- **Instance** - Displays an XML instance generated based on each schema element.
- **Used by** - Displays the list of all the components that reference the current one. The list is sorted by component type and name.
- **Properties** - Displays some of the component's properties.
- **Facets** - Displays the facets for each simple type
- **Identity constraints** - Displays the identity constraints for each element. For each constraint there are presented the name, type (unique, key, keyref), reference attribute, selector and field(s).
- **Attributes** - Displays the attributes for the component. For each attribute there are presented the name, type, fixed or default value, usage and annotation.
- **Asserts** - Displays the **assert** elements defined in a complex type. The test, XPath default namespace, and annotation are presented for each assert.
- **Annotations** - Displays the annotations for the component. If you choose **Escape XML Content**, the XML tags are present in the annotations.
- **Source** - Displays the text schema source for each component.
- **Generate index** - Displays an index with the components included in the documentation.
    - **Include local elements and attributes** - If selected, local elements and attributes are included in the documentation index.
    - **Include resource hierarchy** - Specifies whether or not the resource hierarchy for an XML Schema documentation is generated. It is deselected by default.

Export settings - Save the current settings in a settings file for further use (for example, with the exported settings file you can generate the same *documentation from the command line interface*.)

Import settings - Reloads the settings from the exported file.

Generate - Use this button to generate the XML Schema documentation.

Related Information:

Customizing the PDF Output of Generated XML Schema Documentation on page 608

Generating Documentation for an XSLT Stylesheet

You can use Oxygen XML Editor to generate detailed documentation in HTML format for the elements (top-level elements whose names are in the XSLT namespace) of an XSLT stylesheet. You can select what XSLT elements to include in the generated documentation and also the level of details to present for each of them. The elements are hyperlinked. To generate documentation in a *custom output format*, you can edit the XSLT stylesheet used to generate the documentation, or create your own stylesheet.

To open the **XSLT Stylesheet Documentation** dialog box, select **XSLT Stylesheet Documentation** from the **Tools** > **Generate Documentation** menu or from the **Generate Documentation** submenu in the contextual menu of the ***Project*** *view*.

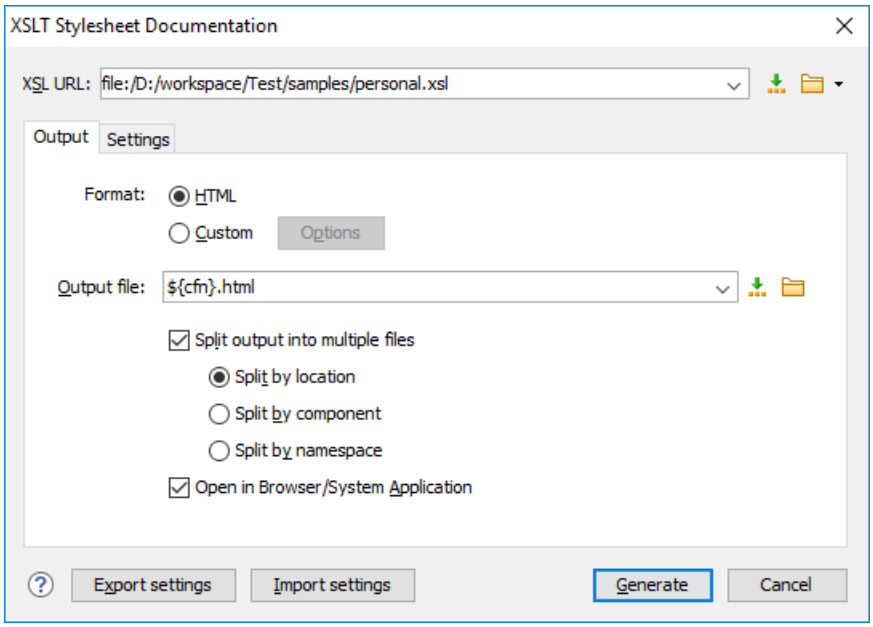


Figure 617: XSLT Stylesheet Documentation Dialog Box

The **XSL URL** field of the dialog box must contain the full path to the XSL Stylesheet file you want to generate documentation for. The stylesheet may be a local or a remote file. You can specify the path to the stylesheet by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.

Output Tab

The following options are available in the **Output** tab:

- **Format** - Allows you to choose between the following formats:
  - **HTML** - The documentation is generated in *HTML output format*.
  - **Custom** - The documentation is generated in a *custom output format*, allowing you to control the output. Click the **Options** button to open a **Custom format options** dialog box where you can specify a custom stylesheet for creating the output. There is also an option to **Copy additional resources to the output folder** and you can select the path to the additional **Resources** that you want to copy. You can also choose to keep the intermediate XML files created during the documentation process by deselecting the **Delete intermediate XML file** option.
- **Output file** - You can specify the path of the output file by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.
- **Split output into multiple files** - Instructs the application to split the output into multiple files. For large XSLT stylesheets, choosing another split criterion may generate smaller output files, providing faster documentation browsing. You can choose to split them by namespace, location, or component name.
- **Open in Browser/System Application** - Opens the result in the system application associated with the output file type.

  Note: To set the browser or system application that will be used, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.

Settings Tab

When you generate documentation for an XSLT stylesheet you can choose what XSLT elements to include in the output (templates, functions, global parameters, global variables, attribute sets, character maps, keys, decimal formats, output formats, XSLT elements from referenced stylesheets) and the details to include in the documentation.

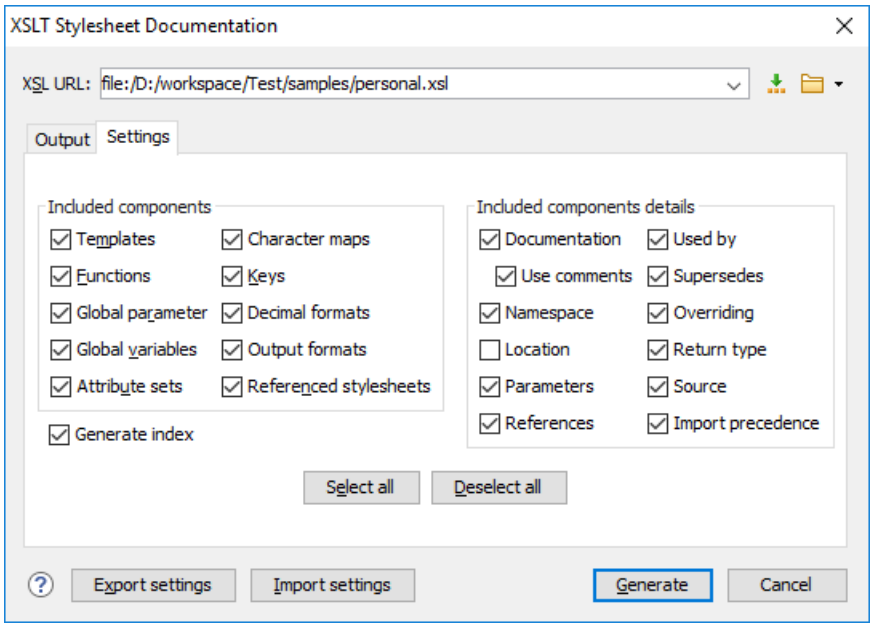


Figure 618: Settings Tab of the XSLT Stylesheet Documentation Dialog Box

The **Settings** tab allows you to choose whether or not to include the following components: **Templates**, **Functions**, **Global parameters**, **Global variables**, **Attribute sets**, **Character maps**, **Keys**, **Decimal formats**, **Output formats**, **Referenced stylesheets**.

You can choose whether or not to include the following other details:

- **Documentation** - Shows the documentation for each XSLT element. For HTML format, the user-defined data elements that are recognized and transformed in documentation blocks of the XSLT elements they precede, are the ones from the following schemas:
  - Oxygen XML Editor built-in XSLT documentation schema.
  - A subset of DocBook 5 elements. The recognized elements are: `section`, `sect1` to `sect5`, `emphasis`, `title`, `ulink`, `programlisting`, `para`, `orderedlist`, `itemizedlist`.
  - A subset of DITA elements. The recognized elements are: `concept`, `topic`, `task`, `codeblock`, p, b, `i`, `ul`, `ol`, `pre`, `sl`, `sli`, `step`, `steps`, `li`, `title`, `xref`.
  - Full XHTML 1.0 support.
  - XSLStyle documentation environment. XSLStyle uses DocBook or DITA languages inside its own user-defined data elements. The supported DocBook and DITA elements are the ones mentioned above.
  - DOXSL documentation *framework*. Supported elements are : `codefrag`, `description`, `para`, `docContent`, `documentation`, `parameter`, `function`, `docSchema`, `link`, `list`, `listitem`, `module`, `parameter`, `template`, `attribute-set`;

    Other XSLT documentation blocks that are not recognized will just be serialized inside an HTML `pre` element. You can change this behavior by using a *custom format* instead of the built-in *HTML format* and providing your own XSLT stylesheets.
- **Use comments** - Controls whether or not the comments that precede an XSLT element is treated as documentation for the element they precede. Comments that precede or succeed the `xsl:stylesheet` element, are treated as documentation for the whole stylesheet. Note that comments that precede an import or include directive are not collected as documentation for the imported/included module. Also, comments from within the body of the XSLT elements are not collected at all.
- **Namespace** - Shows the namespace for named XSLT elements.
- **Location** - Shows the stylesheet location for each XSLT element.
- **Parameters** - Shows parameters of templates and functions.
- **References** - Shows the named XSLT elements that are referenced from within an element.
- **Used by** - Shows the list of all the XSLT elements that reference the current named element.
- **Supersedes** - Shows the list of all the XSLT elements that are superseded the current element.

- **Overriding** - Shows the list of all the XSLT elements that override the current element.
- **Return type** - Shows the return type of the function.
- **Source** - Shows the text stylesheet source for each XSLT element.
- **Import precedence** - Shows the computed import precedence as declared in the XSL transformation specifications.
- **Generate index** - Creates an index with all the XSLT elements included in the documentation.

Export settings - Save the current settings in a settings file for further use (for example, with the exported settings file you can generate the same *documentation from the command-line interface*.)

Import settings - Reloads the settings from the exported file.

Generate - Use this button to generate the XSLT documentation.

Related Information:

XSLT Stylesheet Component Documentation Support on page 530

Generating HTML Documentation for an XQuery Document

To generate HTML documentation for an XQuery document, use the **XQuery Documentation** dialog box. It is opened with the **XQuery Documentation** action that is available from the **Tools** > **Generate Documentation** menu or from the **Generate Documentation** submenu in the contextual menu of the ***Project*** *view*.

The dialog box allows you to configure a set of parameters for the process of generating the HTML documentation.

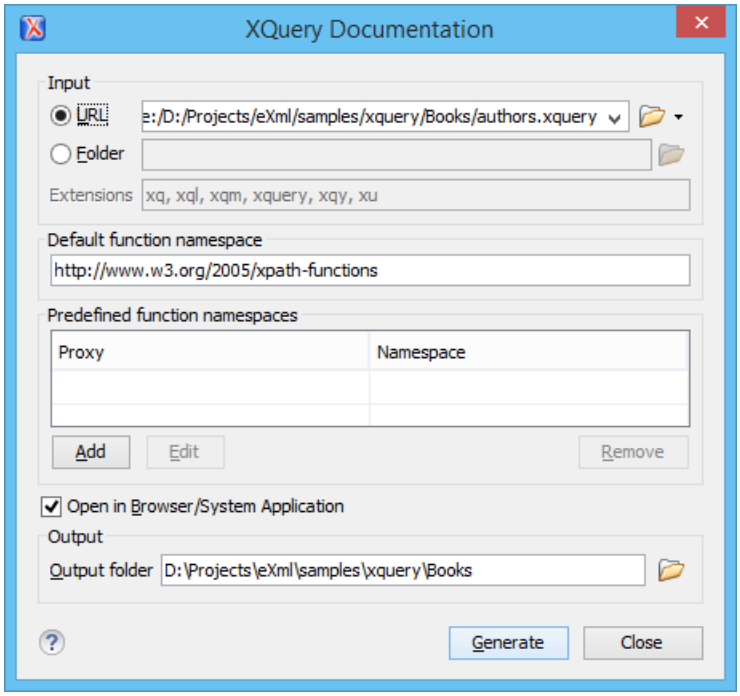


Figure 619: XQuery Documentation Dialog Box

The following options are available:

- **Input** - The full path to the XQuery file must be specified in one of the two fields in this section:
  - **URLFile** - The URL of the file in which you want to generate the documentation.
  - **Folder** - The directory that contains the files for which you want to generate the documentation. You can also specify the XQuery file extensions to be searched for in the specified directory.
- **Default function namespace** - Optional URI for the default namespace for the submitted XQuery.
- **Predefined function namespaces** - Optional, engine-dependent, predefined namespaces that the submitted XQuery refers to. They allow the conversion to generate annotation information to support the presentation

component hypertext linking (only if the predefined modules have been loaded into the local xqDoc XML repository).

- **Open in Browser/System Application** - Select this option if you want the result to be opened in the system application associated with that file type.

  Note: To set the browser or system application that will be used, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.
- **Output** - Allows you to specify where the generated documentation is saved on disk.

Generating Documentation for WSDL Documents

You can use Oxygen XML Editor to generate detailed documentation for the components of a WSDL document in HTML format. You can select the WSDL components to include in your output and the level of details to present for each of them. Also, the components are hyperlinked. You can also generate the documentation in a *custom output format* by using a custom stylesheet.

Note: The WSDL documentation includes the XML Schema components that belong to the internal or imported XML schemas.

To generate documentation for a WSDL document, select **WSDL Documentation** from the **Tools** > **Generate Documentation** menu or from the **Generate Documentation** submenu in the contextual menu of the ***Project** view*.

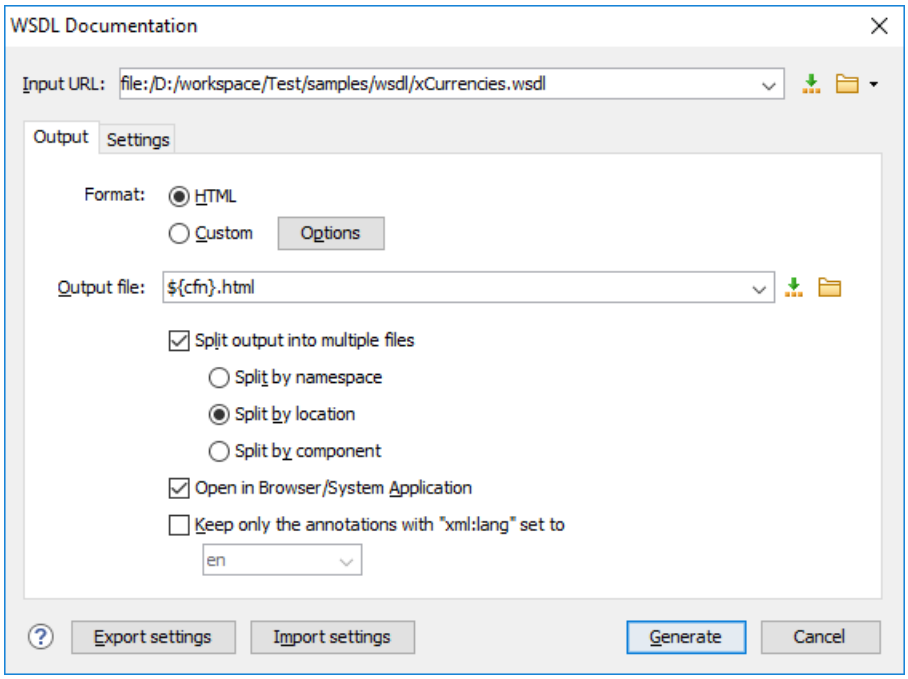


Figure 620: WSDL Documentation Dialog Box

The **Input URL** field of the dialog box must contain the full path to the WSDL document that you want to generate documentation for. The WSDL document may be a local or a remote file. You can specify the path to the WSDL file by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.

Output Tab

The following options are available in the **Output** tab:

- **Format** - Allows you to choose between the following formats:
  - **HTML** - The documentation is generated in *HTML output format*.
  - **Custom** - The documentation is generated in a *custom output format*, allowing you to control the output. Click the **Options** button to open a **Custom format options** dialog box where you can specify a custom

stylesheet for creating the output. There is also an option to **Copy additional resources to the output folder** and you can select the path to the additional **Resources** that you want to copy. You can also choose to keep the intermediate XML files created during the documentation process by deselecting the **Delete intermediate XML file** option.

- **Output file** - You can specify the path of the output file by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.
- **Split output into multiple files** - Instructs the application to split the output into multiple files. For large WSDL documents, choosing a different split criterion may generate smaller output files providing a faster documentation browsing. You can choose to split them by namespace, location, or component name.
- **Open in Browser/System Application** - Opens the result in the system application associated with the output file type.

  Note: To set the browser or system application that will be used, *open the **Preferences** dialog box **(Options > Preferences)***, go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.
- **Keep only the annotations with xml:lang set to** - The generated output will contain only the annotations with the `xml:lang` attribute set to the selected language. If you choose a primary language code (for example, **en** for English), this includes all its possible variations (**en-us**, **en-uk**, etc.).

Setting Tab

When you generate documentation for a WSDL document, you can choose what components to include in the output and the details to be included in the documentation.

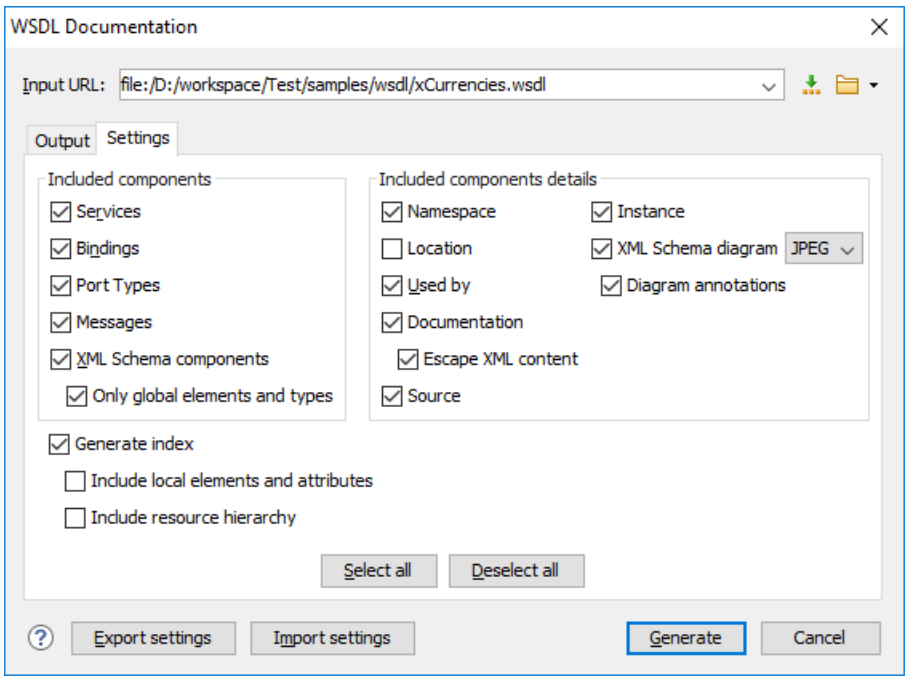


Figure 621: Settings Tab of the WSDL Documentation Dialog Box

The **Settings** tab allows you to choose whether or not to include the following:

- **Components**
  - **Services** - Specifies whether or not the generated documentation includes the WSDL services.
  - **Bindings** - Specifies whether or not the generated documentation includes the WSDL bindings.
  - **Port Types** - Specifies whether or not the generated documentation includes the WSDL port types.
  - **Messages** - Specifies whether or not the generated documentation includes the WSDL messages.
  - **XML Schema Components** - Specifies whether or not the generated documentation includes the XML Schema components.

    - **Only global elements and types** - Specifies whether or not the generated documentation includes only global elements and types.
- **Component Details**
  - **Namespace** - Presents the namespace information for WSDL or XML Schema components.
  - **Location** - Presents the location information for each WSDL or XML Schema component.
  - **Used by** - Presents the list of components that reference the current one.
  - **Documentation** - Presents the component documentation. If you choose **Escape XML Content**, the XML tags are presented in the documentation.
  - **Source** - Presents the XML fragment that defines the current component.
  - **Instance** - Generates a sample XML instance for the current component.

    Note: This option applies to the XML Schema components only.
  - **XML Schema Diagram** - Displays the diagram for each XML Schema component. You can choose the image format (JPEG, PNG, GIF, SVG) to use for the diagram section.
    - **Diagram annotations** - Specifies whether or not the annotations of the components presented in the diagram sections are included.
- **Generate index** - Displays an index with the components included in the documentation.
  - **Include local elements and attributes** - If selected, local elements and attributes are included in the documentation index.
  - **Include resource hierarchy** - Specifies whether or not the resource hierarchy for an XML Schema documentation is generated. It is deselected by default.

Export settings - Save the current settings in a settings file for further use (for example, with the exported settings file you can generate the same *documentation from the command-line interface*.)

Import settings - Reloads the settings from the exported file.

Generate - Use this button to generate the WSDL documentation.

Canonicalizing Files

You can select the *canonicalization* algorithm to be used for a document from the dialog box that is displayed by using the **Canonicalize** action that is available from the **Source** submenu when invoking the contextual menu in **Text** mode or from the **Tools** menu.

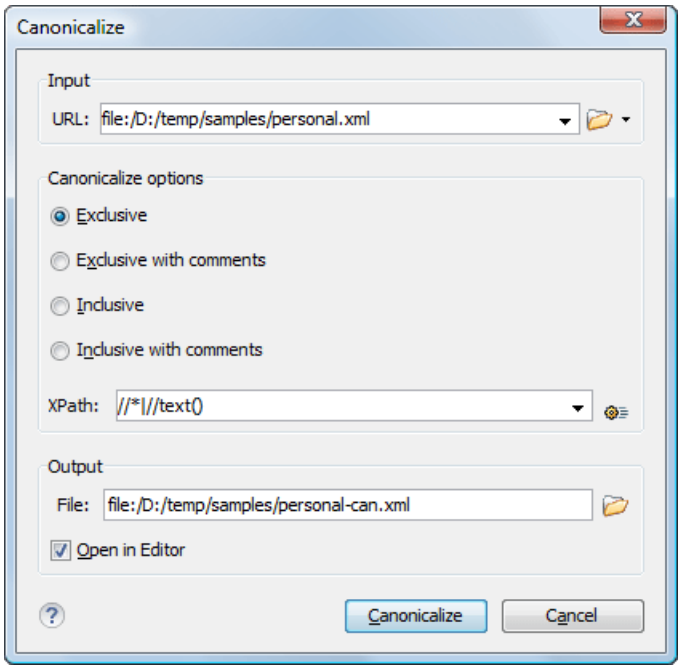


Figure 622: Canonicalization Settings Dialog Box

The **Canonicalize** dialog box allows you to set the following options:

- **Input URL** - Available if the **Canonicalize** action was selected from the **Tools** menu. It allows you to specify the location of the input file.
- **Exclusive** - If selected, the exclusive (uncommented) *canonicalization* method is used.

  Note: *Exclusive Canonicalization* just copies the namespaces you are actually using (the ones that are a part of the XML syntax). It does not look into attribute values or element content, so the namespace declarations required to process these are not copied. This is useful if you have a signed XML document that you want to insert into other XML documents (or you need self-signed structures that support placement within various XML contexts), as it will ensure the signature is verified correctly each time.
- **Exclusive with comments** - If selected, the exclusive with comments *canonicalization* method is used.
- **Inclusive** - If selected, the inclusive (uncommented) *canonicalization* method is used.

  Note: *Inclusive Canonicalization* copies all the declarations, even if they are defined outside of the scope of the signature, and all the declarations you might use will be unambiguously specified. Inclusive *Canonicalization* is useful when it is less likely that the signed data will be inserted in other XML document and it is the safer method from the security standpoint because it requires no knowledge of the data that are to be secured to safely sign them. A problem may occur if the signed document is moved into another XML document that has other declarations because the Inclusive *Canonicalization* will copy them and the signature will be invalid.
- **Inclusive with comments** - If selected, the inclusive with comments *canonicalization* method is used.
- **XPath** - The XPath expression provides the fragments of the XML document to be signed.
- **Output** - Available if the **Canonicalize** action was selected from the **Tools** menu. It allows you to specify the output file path where the signed XML document will be saved.
- **Open in editor** - If selected, the output file will be opened in the editor.

Related Information:

Digital Signatures Overview on page 736

Signing Files

You can select the type of signature to be used for documents from a signature settings dialog box. To open this dialog box, select the **Sign** action from the **Source** submenu when invoking the contextual menu in **Text** mode or from the **Tools** menu.

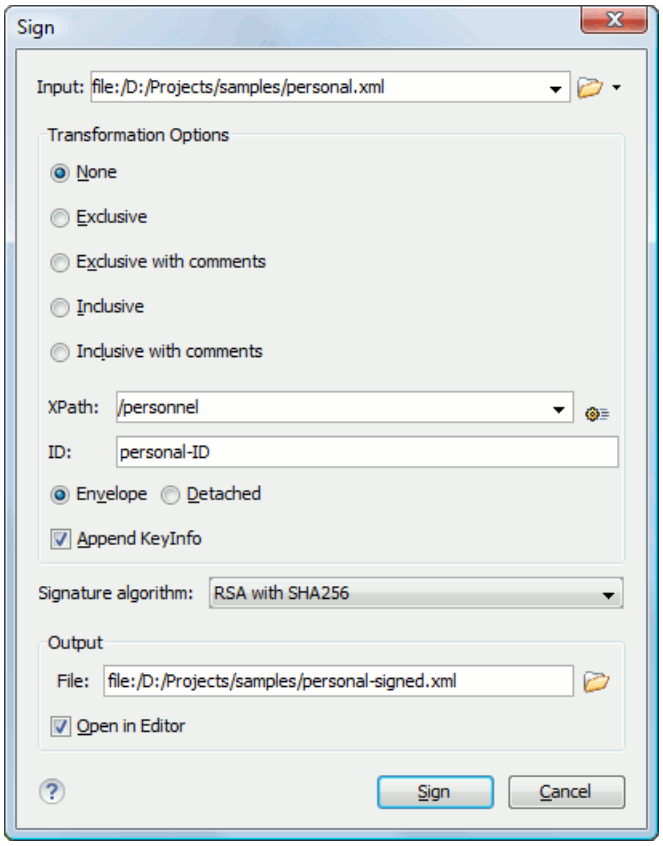


Figure 623: Signature Settings Dialog Box

The following options are available:

Note: If Oxygen XML Editor could not find a valid certificate, a link is provided at the top of the dialog box that opens the *XML Signing Certificates preferences page* where you can configure a valid certificate.

- **Input** - Available if the **Sign** action was selected from the **Tools** menu. Specifies the location of the input URL.
- **Transformation Options** - See the *Digital Signature Overview* section for more information about these options.
  - **None** - If selected, no *canonicalization* algorithm is used.
  - **Exclusive** - If selected, the exclusive (uncommented) *canonicalization* method is used.

    Note: *Exclusive Canonicalization* just copies the namespaces you are actually using (the ones that are a part of the XML syntax). It does not look into attribute values or element content, so the namespace declarations required to process these are not copied. This is useful if you have a signed XML document that you want to insert into other XML documents (or you need self-signed structures that support placement within various XML contexts), as it will ensure the signature is verified correctly each time.
  - **Exclusive with comments** - If selected, the exclusive with comments *canonicalization* method is used.
  - **Inclusive** - If selected, the inclusive (uncommented) *canonicalization* method is used.

    Note: *Inclusive Canonicalization* copies all the declarations, even if they are defined outside of the scope of the signature, and all the declarations you might use will be unambiguously specified. Inclusive *Canonicalization* is useful when it is less likely that the signed data will be inserted in other XML document and it is the safer method from the security standpoint because it requires no knowledge of the data that are to be secured to safely sign them. A problem may occur if the signed document is moved into another XML document that has other declarations because the Inclusive *Canonicalization* will copy them and the signature will be invalid.
  - **Inclusive with comments** - If selected, the inclusive with comments *canonicalization* method is used.
- **XPath** - The XPath expression provides the fragments of the XML document to be signed.

- **ID** - Provides ID of the XML element to be signed.
- **Envelope** - If selected, the *enveloped* signature is used. See the *Digital Signature Overview* for more information.
- **Detached** - If selected, the *detached* signature is used. See the *Digital Signature Overview* for more information.
- **Append KeyInfo** - If this option is selected, the `ds:KeyInfo` element will be added in the signed document.
- **Signature algorithm** - The algorithm used for signing the document. The following options are available: **RSA with SHA1**, **RSA with SHA256**, **RSA with SHA384**, and **RSA with SHA512**.
- **Output** - Available if the **Sign** action was selected from the **Tools** menu. Specifies the path of the output file where the signed XML document will be saved.
- **Open in editor** - If selected, the output file will be opened in Oxygen XML Editor.

Related Information:

Digital Signatures Overview on page 736
Verifying Signature on page 740
Example of How to Digitally Sign XML Files or Content on page 741

Verifying Signature

You can verify the signature of a file by selecting the **Verify Signature** action from the **Source** submenu when invoking the contextual menu in **Text** mode or from the **Tools** menu. The **Verify Signature** dialog box then allows you to specify the location of the file whose signature is verified.

If the signature is valid, a dialog box displays the name of the signer. Otherwise, an error shows details about the problem.

Related Information:

Digital Signatures Overview on page 736
Signing Files on page 739
Example of How to Digitally Sign XML Files or Content on page 741

WSDL SOAP Analyzer

WSDL SOAP Analyzer is a tool that helps you test if the messages defined in a Web Service Descriptor (WSDL) are accepted by a Web Services server.

After you edit and validate your Web service descriptor against a mix of the XML Schemas for WSDL and SOAP, it is easy to check if the defined SOAP messages are accepted by the remote Web Services server by using the integrated **WSDL SOAP Analyzer** tool (available from the toolbar or **Tools** menu).

Oxygen XML Editor provides two ways of testing, one for the currently edited WSDL document and another for the remote WSDL documents that are published on a web server. To open the **WSDL SOAP Analyzer** tool for the currently edited WSDL document do one of the following:

- Click the **WSDL SOAP Analyzer** toolbar button.
- Use the **WSDL SOAP Analyzer** action from the **Tools** menu.
- Go to **Open with** > **WSDL SOAP Analyzer** in the contextual menu of the ***Project*** view.

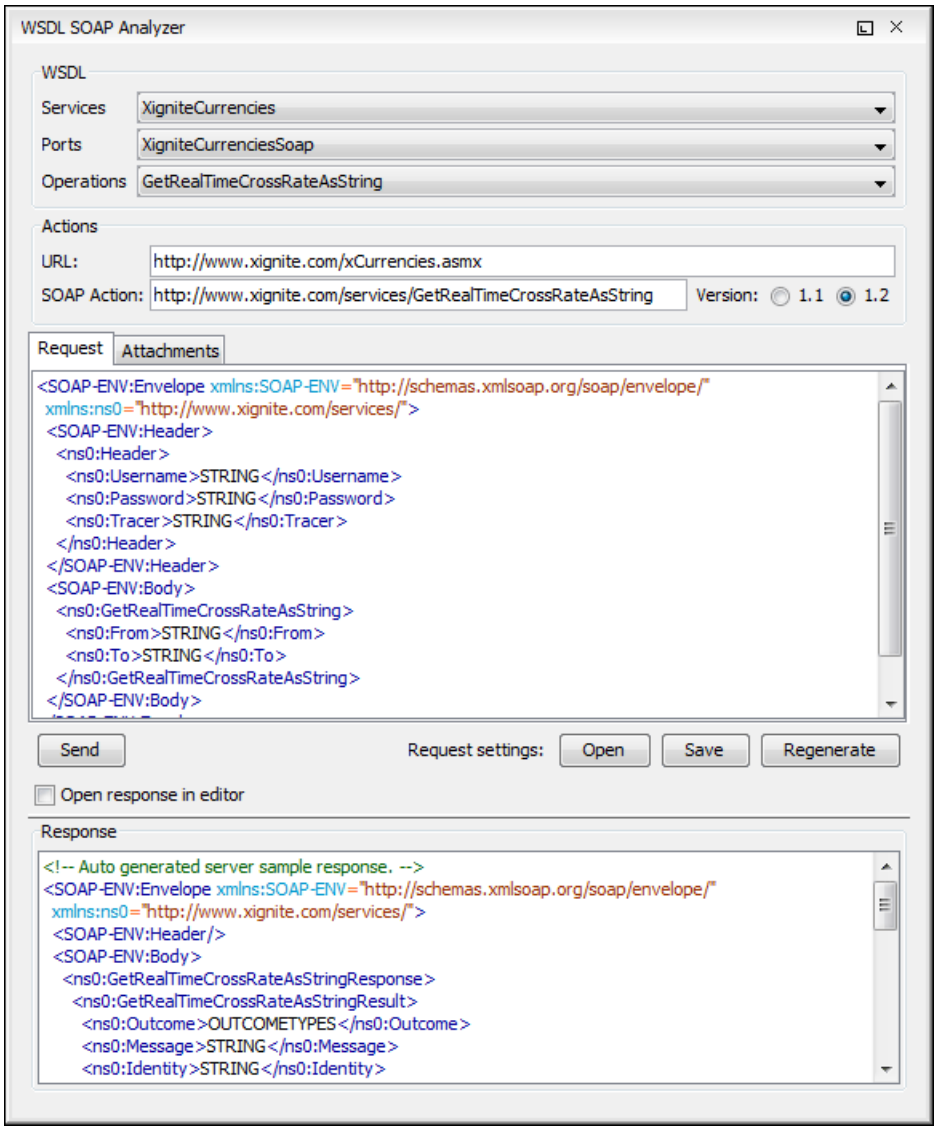


Figure 624: WSDL SOAP Analyzer Dialog Box

This tool contains a SOAP analyzer and sender for Web Services Description Language file types. The analyzer fields are as follows:

- **Services** - The list of services defined by the WSDL file.
- **Ports** - The ports for the selected service.
- **Operations** - The list of available operations for the selected service.
- **Action URL** - The script that serves the operation.
- **SOAP Action** - Identifies the action performed by the script.
- **Version** - Choose between 1.1 and 1.2. The SOAP version is selected automatically depending on the selected port.
- **Request Editor** - It allows you to compose the web service request. When an action is selected, Oxygen XML Editor tries to generate as much content as possible for the SOAP request. The envelope of the SOAP request has the correct namespace for the selected SOAP version, that is *http://schemas.xmlsoap.org/soap/envelope/* for SOAP 1.1 or *http://www.w3.org/2003/05/soap-envelope* for SOAP 1.2. Usually you just have to change a few values for the request to be valid. The *Content Completion Assistant* is available for this editor and is driven by the schema that defines the type of the current message. While selecting various operations, Oxygen XML Editor remembers the modified request for each one. You can press the **Regenerate** button to overwrite your modifications for the current request with the initial generated content.
- **Attachments List** - You can define a list of file URLs to be attached to the request.

- **Response Area** - Initially it displays an auto generated server sample response so you can have an idea about how the response looks like. After pressing the **Send** button, it presents the message received from the server in response to the Web Service request. It may show also error messages. If the response message contains attachments, Oxygen XML Editor prompts you to save them, then tries to open them with the associated system application.
- **Errors List** - There may be situations where the WSDL file is respecting the WSDL XML Schema, but it fails to be valid (for example, in the case of a message that is defined by means of an element that is not found in the types section of the WSDL). In such a case, the errors are listed here. This list is presented only when there are errors.
- **Send Button** - Executes the request. A status dialog box is displayed when Oxygen XML Editor is connecting to the server.

The testing of a WSDL file is straight-forward: click the WSDL analysis button, then select the service, the port, and the operation. The editor generates the skeleton for the SOAP request. You can edit the request, eventually attach files to it and send it to the server. Watch the server response in the response area. You can find more details in the *Testing Remote WSDL Files* section.

Note: SOAP requests and responses are automatically validated in the **WSDL SOAP Analyzer** using the XML Schemas specified in the WSDL file.

Once defined, a request derived from a Web Service descriptor can be saved with the **Save** button to a Web Service SOAP Call (WSSC) file for later reuse. In this way, you save time in configuring the URLs and parameters.

You can open the result of a Web Service call in an editor panel using the **Open** button.

Testing Remote WSDL Files

To open and test a remote WSDL file the steps are the following:

1. Go to **Tools** > **WSDL SOAP Analyzer** .
2. On the **WSDL File** tab enter the URL of the remote WSDL file.
   You enter the URL:
   - by typing
   - by browsing the local file system
   - by browsing a remote file system
   - by browsing *a UDDI Registry*
3. Press the **OK** button.
   This will open the ***WSDL SOAP Analyzer** tool*. In the **Saved SOAP Request** tab you can open directly a previously saved Web Service SOAP Call (WSSC) file, thus skipping the analysis phase.

UDDI Registry Browser

Pressing the button in the **WSDL File Opener** dialog box (menu **Tools** > **WSDL SOAP Analyzer**) opens the **UDDI Registry Browser** dialog box.

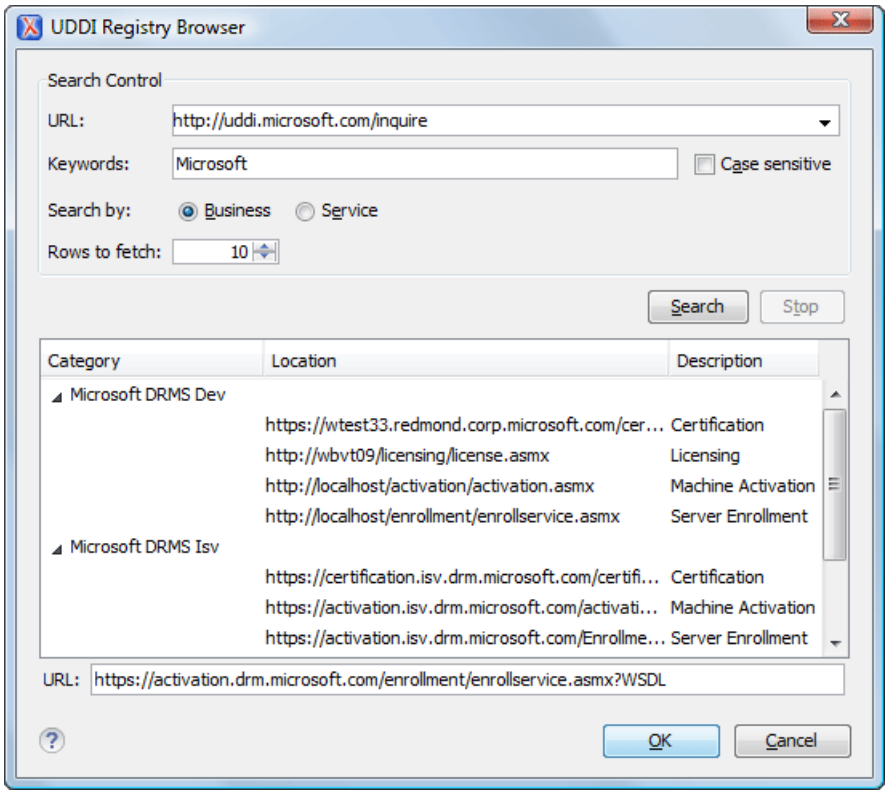


Figure 625: UDDI Registry Browser Dialog Box

The fields of the dialog box are as follows:

- **URL** - Type the URL of an UDDI registry or choose one from the default list.
- **Keywords** - Enter the string you want to be used when searching the selected UDDI registry for available Web services.
- **Rows to fetch** - The maximum number of rows to be displayed in the result list.
- **Search by** - You can choose to search either by company or by provided service.
- **Case sensitive** - When selected, the search takes into account the keyword case.
- **Search** - The WSDL files that matched the search criteria are added in the result list.

When you select a WSDL from the list and click the **OK** button, the **UDDI Registry Browser** dialog box is closed and you are returned to the **WSDL File Opener** dialog box.

XML Schema Regular Expressions Builder

The XML Schema regular expressions builder allows you to test regular expressions on a fragment of text as they are applied to an XML instance document. Start the tool by selecting **XML Schema Regular Expressions Builder** from the **Tools** menu.

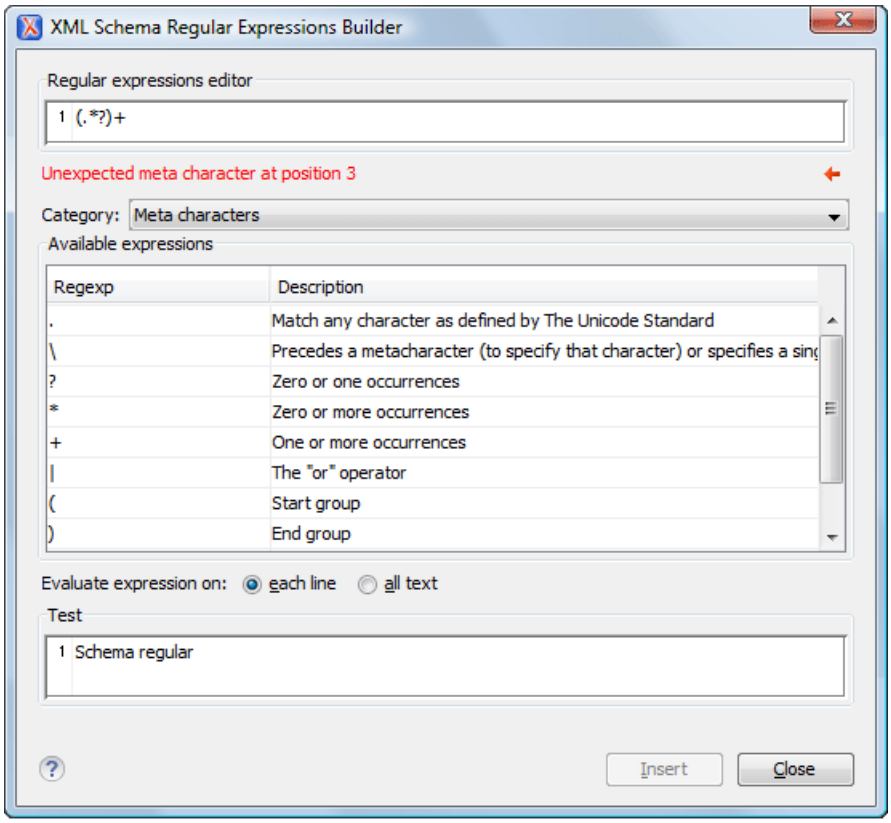


Figure 626: XML Schema Regular Expressions Builder Dialog Box

The dialog box contains the following:

Regular expressions editor

Allows you to edit the regular expression to be tested and used. Content completion is available and presents a list with all the predefined expressions. It is triggered by pressing **Ctrl + Space (Command + Space on OS X)**.

Error display area

If the edited regular expression is incorrect, an error message will be displayed here. The message contains the description and the exact location of the error. Also, clicking the quick navigation button (←) highlights the error inside the regular expression.

Category

You can choose from several categories of predefined expressions. The selected category influences the displayed expressions in the **Available expressions** table.

Available expressions

This table includes the available regular expressions and a short description for each of them. The set of expressions depends on the category selected in the previous **Category** combo box. You can add an expression in the **Regular expressions editor** by double-clicking the expression row in the table. You will notice that in the case of **Character categories** and **Block names**, the expressions are also listed in complementary format.

Evaluate expression on

You can choose between two options:

- **Evaluate expression on each line** - The edited expression will be applied on each line in the **Test** area.
- **Evaluate expression on all text** - The edited expression will be applied on the whole text.

Test

A text editor that allows you to enter a text sample for which the regular expression will be applied. All matches of the edited regular expression will be highlighted.

After editing and testing your regular expression you can insert it in the current editor. The **Insert** button will become active when an editor is opened in the background and there is an expression in the **Regular expressions editor**.

The regular expression builder cannot be used to insert regular expressions in the ***Grid*** *mode* or *schema* ***Design*** *mode*. Accordingly, the **Insert** button will be not available if the current document is edited in these modes.

Note: Some regular expressions may indefinitely block the Java Regular Expressions engine. If the execution of the regular expression does not end in about five seconds, the application displays a dialog box that allows you to interrupt the operation.

Large File Viewer

XML files tend to become larger and larger mostly because they are frequently used as a format for database export or for porting between multiple database formats. Traditional XML text editors simply cannot handle opening these huge export files, some having sizes exceeding one gigabyte, because all the file content must be loaded in memory before the user can actually view it.

The best performance of the viewer is obtained for encodings that use a fixed number of bytes per character (such as UTF-16 or ASCII). The performance for UTF-8 is very good for documents that use mostly characters of the European languages. For the same encoding, the rendering performance is higher for files consisting of long lines (up to few thousands characters) and may degrade for short lines. In fact, the maximum size of a file that can be rendered in the Large File Viewer decreases when the total number of the text lines of the file increases. Trying to open a very large file (for example, a file of 4 GB) with a very high number of short lines (100 or 200 characters per line) may produce an *out of memory* error (**OutOfMemoryError**) that would require either increasing the Java heap memory with the -Xmx startup parameter or decreasing the total number of lines in the file.

The powerful **Large File Viewer** is available from the **Tools** menu or as a standalone application. You can also right-click a file in your project and choose to open it with the viewer. It uses an efficient structure for indexing the opened document. No information from the file is stored in the main memory, just a list of indexes in the file. In this way the viewer can open very large files, up to 10 gigabytes. If the opened file is XML, the encoding used to display the text is detected from the XML prolog of the file. For other file types, the encoding is taken from the Oxygen XML Editor options. See *Encoding for non XML files*.

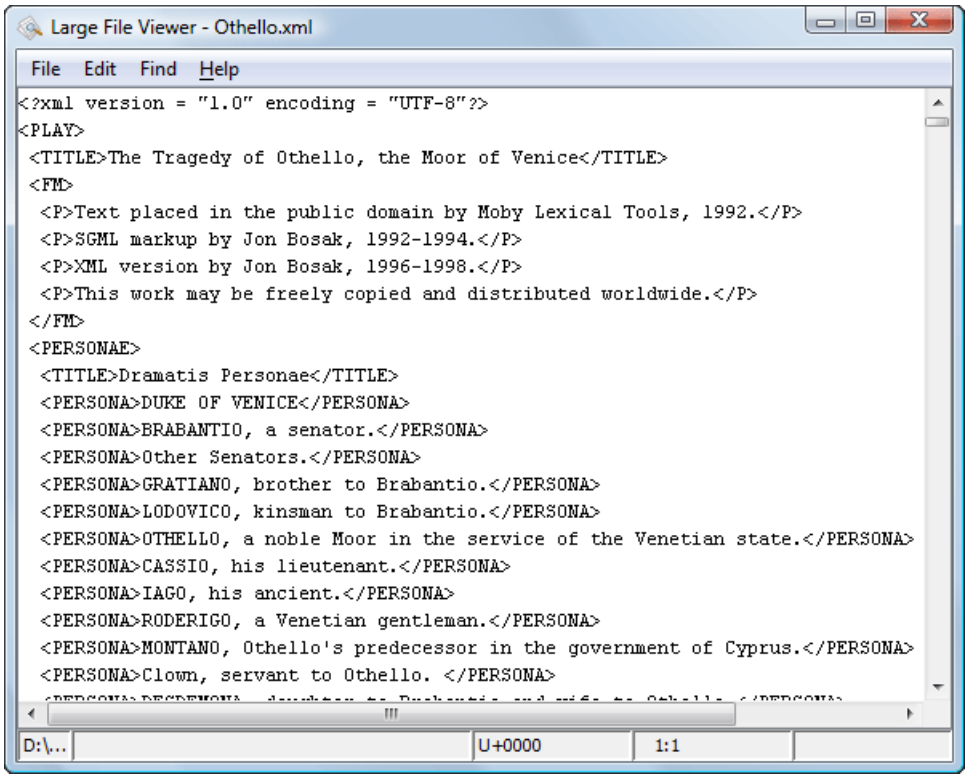


Figure 627: Large File Viewer

Large File Viewer components:

- The menu bar provides menu driven access to all the features and functions that are available in **Large File Viewer**:

  File > Open

  Opens files in the viewer (also available in the contextual menu).

  File > Exit

  Closes the viewer.

  Edit > Copy

  Copies the selected text to clipboard (also available in the contextual menu).

  Find > Find

  Opens a reduced **Find** dialog box that provides some basic search options, such as:

  - **Case sensitive** - When selected, operations are case-sensitive.
  - **Regular Expression** - When selected, allows you to use any regular expression in *Perl-like syntax*.
  - **Wrap around** - Continues the find operation from the start/end of the document after reaching the end/, depending on whether the search is in forward or backward direction.

  Help > Help

  Provides access to the User Manual.

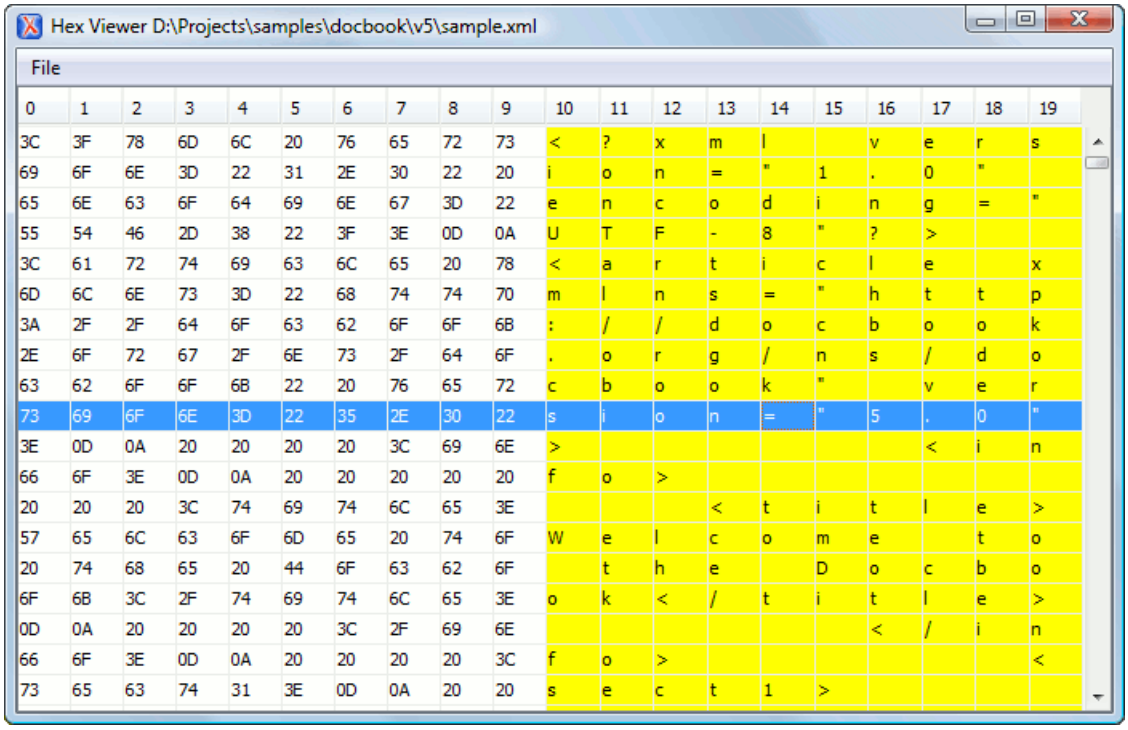
- The status bar provides information about the current opened file path, the Unicode representation of the character at cursor position and the line and column in the opened document where the cursor is located.

Attention: For faster computation the **Large File Viewer** uses a fixed font (plain, monospace font of size 12) to display characters. The font is *not* configurable from the ***Preferences*** *page*.

Tip: The best performance of the viewer is accomplished for encodings that use a fixed number of bytes per character (such as UTF-16 or ASCII). The performance for UTF-8 is very good for documents that use mostly characters of the European languages. For the same encoding the rendering performance is high for files consisting of short lines (up to a few thousand characters) and may degrade for long lines.

Hex Viewer

When the Unicode characters that are visible in a text viewer or editor are not enough and you need to see the byte values of each character of a document, you can start the **Hex Viewer** that is available on the **Tools** menu. It has two panels: the characters are rendered in the right panel and the bytes of each character are displayed in the left panel. There is a 1:1 correspondence between the characters and their byte representation: the byte representation of a character is displayed in the same matrix position of the left panel as the character in the matrix of the right panel.



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
3C	3F	78	6D	6C	20	76	65	72	73	<	?	x	m	l		v	e	r	s
69	6F	6E	3D	22	31	2E	30	22	20	i	o	n	=	"	1	.	0	"	
65	6E	63	6F	64	69	6E	67	3D	22	e	n	c	o	d	i	n	g	=	"
55	54	46	2D	38	22	3F	3E	0D	0A	U	T	F	-	8	"	?	>		
3C	61	72	74	69	63	6C	65	20	78	<	a	r	t	i	c	l	e		x
6D	6C	6E	73	3D	22	68	74	74	70	m	l	n	s	=	"	h	t	t	p
3A	2F	2F	64	6F	63	62	6F	6F	6B	:	/	/	d	o	c	b	o	o	k
2E	6F	72	67	2F	6E	73	2F	64	6F	.	o	r	g	/	n	s	/	d	o
63	62	6F	6F	6B	22	20	76	65	72	c	b	o	o	k	"		v	e	r
73	69	6F	6E	3D	22	35	2E	30	22	s	i	o	n	=	"	5	.	0	"
3E	0D	0A	20	20	20	20	3C	69	6E	>							<	i	n
66	6F	3E	0D	0A	20	20	20	20	20	f	o	>							
20	20	20	3C	74	69	74	6C	65	3E				<	t	i	t	l	e	>
57	65	6C	63	6F	6D	65	20	74	6F	W	e	l	c	o	m	e		t	o
20	74	68	65	20	44	6F	63	62	6F		t	h	e		D	o	c	b	o
6F	6B	3C	2F	74	69	74	6C	65	3E	o	k	<	/	t	i	t	l	e	>
0D	0A	20	20	20	20	3C	2F	69	6E							<	/	i	n
66	6F	3E	0D	0A	20	20	20	20	3C	f	o	>							<
73	65	63	74	31	3E	0D	0A	20	20	s	e	c	t	1	>				

Figure 628: Hex Viewer

To open a file in **Hex Viewer** use the **File** > **Open** action. Alternatively, you can drag a file and drop it in the **Hex Viewer** panel.

Standalone SVG Viewer

Oxygen XML Editor includes a simple **SVG Viewer** that allows you to work with SVG images.

To open the viewer, select **SVG Viewer** from the **Tools** menu.

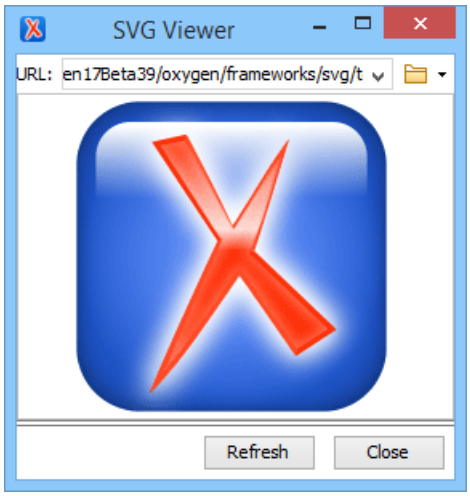


Figure 629: SVG Viewer

You can browse for and open any SVG file that has the `.svg` or `.svgz` extension.

If the file is included in the current project, you can open it in the viewer by right-clicking the image file in the ***Project** view* and selecting **Open with** > **SVG Viewer**.

Actions Available in the SVG Viewer

The following actions are available in the **SVG Viewer**:

Zoom in

To zoom in on an image, use any of the following methods:

- Scroll **forward** with the mouse wheel.
- Select **Zoom in** from the contextual menu.
- Use the **Ctrl + I (Command + I on OS X)** keyboard shortcut.

Zoom out

To zoom in on an image, use any of the following methods:

- Scroll **backward** with the mouse wheel.
- Use the **Ctrl + O (Command + O on OS X)** keyboard shortcut.
- Select **Zoom out** from the contextual menu.

Rotate

To rotate an image, use either of the following methods:

- Use the **Ctrl + Right-Click + Drag (Command + Right-Click + Drag on OS X)** shortcut.
- Select **Rotate** from the contextual menu. This rotates the image exactly 90 degrees clockwise.

Refresh

To refresh (or reset) an image, use either of the following methods:

- Use the **Ctrl + T (Command + T on OS X)** keyboard shortcut.
- Select **Refresh** from the contextual menu.

Move

To move an image, use either of the following methods:

- Use the **Arrow Keys** on your keyboard.
- Use the **Shift + Left-Click + Drag** shortcut.

Pan

To pan an image, **click and drag** the image with your mouse.

Related Information:

Editing SVG Files on page 704

Tree Editor

The **Tree Editor** (**Tools** > **Tree Editor**) is used for editing the content of a document displayed as an XML tree. The workspace offers the following functional areas:

- Main menu - Provides access to all the features and functions available in Oxygen XML Editor **Tree Editor**.
- Toolbar - Provides easy access to common and frequently used functions. Each icon is a button that acts as a shortcut to a related function.
- Editor panel - Easy editing of structured mark-up documents. Each token has an associated icon for easier visual identification.
- Message panel - Displays messages returned from user operations.
- Model view - Shows the detailed information about the attribute or element that you are working on.
- All Elements panel - Presents a list of all defined elements that can be inserted within your document.

The tree editor does not offer entity support. Entities are not presented with a special type of node in the tree and new entity nodes cannot be inserted in the document.

Compare Files

The **Compare Files** tool can be used to compare files or XML file fragments. The tool provides a mechanism for comparing two files or fragments, as well as the mechanism for a three-way comparison. The utility is available from the **Tools** menu or can be opened as a stand-alone application from the Oxygen XML Editor installation folder (`diffFiles.exe`).

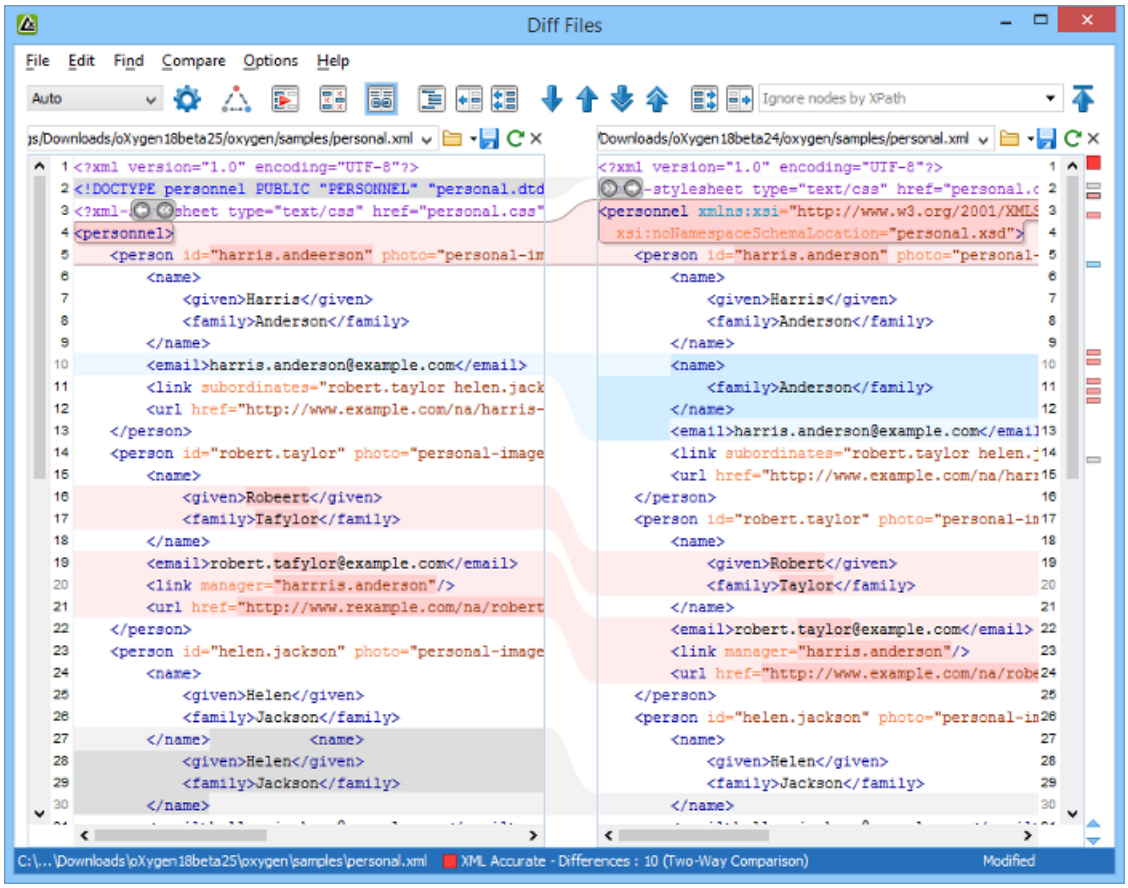


Figure 630: Compare Files Tool

Starting the Tool from a Command Line

The file comparison tool can also be started by using command-line arguments. In the installation folder there is an executable shell (`diffFiles.bat` on Windows, `diffFiles.sh` on Unix/Linux, `diffFilesMac.sh` on OS X). To specify the files to compare, you can pass command-line arguments using the following construct: `diffFiles.bat/diffFiles.sh/diffFilesMac.sh [path to left file] [path to right file] [path to 3-way base file]`.

If three files are specified, the tool will start in the *3-way comparison mode*. If only two files are specified, the tool will start in the *2-way comparison mode*. The first specified file will be added to the left panel in the comparison tool, the second file to the right panel, and the optional third file will be the base (ancestor) file used for a 3-way comparison. If you pass only one argument, you are prompted to manually choose another file.

If you want to launch the file comparison tool from an external application with specified files and you want the file browsing tools at the top of both panels to be hidden, you should use the `-ext` argument as the first command. There are some additional arguments that are allowed and to see all the details for the command line construct, type `diffFiles.bat --help` in the command line.

For example, to do a 3-way comparison on Windows, the command line might look like this:

```
diffFiles.bat "c:\docs\file 1" "c:\docs\file 2" c:\docs\basefile
```

Tip: If there are spaces in the path names, surround the paths with quotes.

Two-Way Comparisons

The **Compare Files** tool can be used to compare the differences between two files or XML fragments.

Compare Files

To perform a two-way comparison, follow these steps:

1. Open a file in the left panel and the file you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the browsing tools in the **Browse** drop-down menu.

   Step Result: The selected files are opened in the two side-by-side editors. A text editing mode is used to offer a better view of the differences.
2. To highlight the differences between the two files, click the **Perform File Differencing** button from the toolbar.
3. You can use the drop-down menu on the left side of the toolbar to change the *algorithm* for the operation.
4. You can also use the **Diff Options** button to access the **Files Comparison** preferences page where you can choose to ignore certain types of markup and configure various options.
5. If you are comparing XML documents using the **XML Fast** or **XML Accurate** algorithms, you can enter an XPath 2.0 expression in the **Ignore nodes by XPath** text field to ignore certain nodes from the comparison.

The resulting comparison will show you differences between the two files. The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes. This layout allows you to easily identify and focus on a group of related changes.

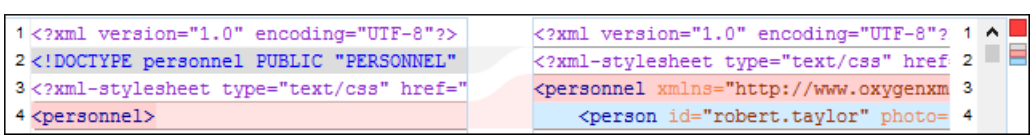


Figure 631: Two-Way Differences

Highlighting Colors

The differences are also highlighted in several colors, depending on the type of change, and dynamic lines connect the compared fragments in the middle section between the two panes. The highlighting colors can be customized in the ***Files Comparison / Appearance*** *preferences page*, but the default colors and their shades mean the following:

- **Pink** - Identifies modifications on either side.
- **Gray** - Identifies an addition of a node in the left side (your outgoing changes).
- **Blue** - Identifies an addition of a node in the right side (incoming changes).
- **Lighter Shade** - Identifies blocks of changes that can be merged in their entirety.
- **Darker Shade** - Identifies specific changes within the blocks that can be merged more precisely.

Compare Fragments

To compare XML file fragments, you need to copy and paste the fragments you want to compare into each side, without selecting a file. If a file is already selected, you need to close it using the **Close (Ctrl + W (Command + W on OS X))** button, before pasting the fragments. If you save modified fragments, a dialog box opens that allows you to save the changes as a new document.

Navigate Differences

To navigate through differences, do one of the following:

- Use the navigation buttons on the toolbar (or in the **Compare** menu).
- Select a block of differences by clicking its small colored marker in the overview ruler located in the right-most part of the window. At the top of the overview ruler there is a success indicator that turns green where there are no differences, or red if differences are found.
- Click a colored area in between the two text editors.

Editing Actions

You can edit the files directly in either editing pane. The two editors are constantly synchronized and the differences are refreshed when you save the modified document or when you click the **Perform File Differencing** button.

A variety of actions are available on the *toolbar* and in the *various menus* (these same actions are also available in the contextual menu in both editing panes). The tool also includes some inline actions to help you merge, copy, or remove changes. When you select a change, the following inline action widgets are available, depending on the type of change:

Append left change to right and Append right change to left

Copies the content of the selected change from one side and appends it on the other, according to the content of the corresponding change. As a result, the side where the arrow points to will contain the changes from both sides.

Copy change from left to right and Copy change from right to left

Replaces the content of a change from one side with the content of the corresponding change from the other side.

Remove change

Rejects the change on the particular side and preserves the particular content on the other side.

Two-Way Diff Algorithms

Oxygen XML Editor offers the following two-way diff algorithms to compare files or fragments:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Characters** - Computes the differences at character level, meaning that it compares two files or fragments looking for identical characters.
- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words.
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **Syntax Aware** - Computes differences for known file types or fragments. This algorithm splits the files or fragments into sequences of *tokens* and computes the differences between them. The meaning of a *token* depends on the type of compared files or fragments.

Known file types include those listed in the **New** dialog box, such as XML file types (XSLT files, XSL-FO files, XSD files, RNG files, NVDL files, etc.), XQuery file types (`.xquery`, `.xq`, `.xqy`, `.xqm` extensions), DTD file types (`.dtd`, `.ent`, `.mod` extensions), TEXT file type (`.txt` extension), or PHP file type (`.php` extension).

For example:

- When comparing XML files or fragments, a token can be one of the following:
  - The name of an XML tag
  - The < character
  - The /> sequence of characters
  - The name of an attribute inside an XML tag
  - The = sign
  - The " character
  - An attribute value
  - The text string between the start tag and the end tag (a text node that is a child of the XML element corresponding to the XML tag that encloses the text string)
- When comparing plain text, a token can be any continuous sequence of characters or any continuous sequence of whitespaces, including a new line character.

- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Three-Way Comparisons

Oxygen XML Editor also includes a three-way comparison feature to help you solve conflicts and merge changes between multiple modifications. It is especially helpful for teams who have multiple authors editing and committing the same documents. It provides a comparison between a local change, another change, and the original base revision. Some additional advantages include:

- Visualize and merge content that was modified by you and another member of your team.
- Marks differences correctly even when the document structure is rearranged.
- Allows you to merge XML-relevant modifications.

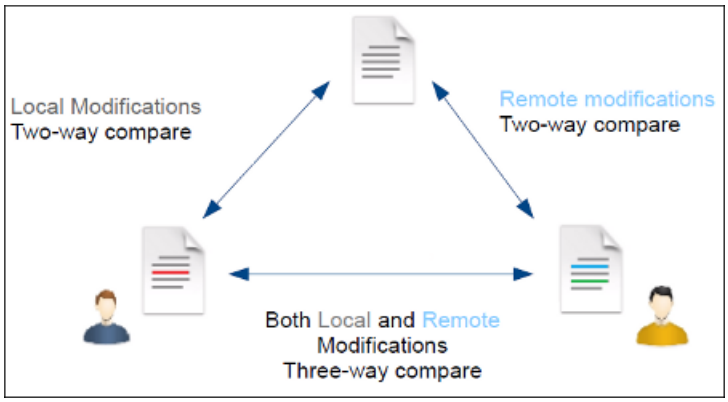


Figure 632: Three-Way Comparison

Compare Files

To perform a three-way comparison, follow these steps:

1. Open a file in the left panel and the file you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the browsing tools in the **Browse** drop-down menu.

   Step Result: The selected files are opened in the two side-by-side editors. A text editing mode is used to offer a better view of the differences.

2. Click the **Three-Way Comparison** button on the toolbar and select the base file in the **Base** field. You can specify the path by using the text field, the history drop-down, or the browsing tools in the **Browse** drop-down menu.
3. To highlight the differences, click the **Perform File Differencing** button on the toolbar.
4. You can use the drop-down menu on the left side of the toolbar to change the *algorithm* for the operation.
5. You can also use the **Diff Options** button to access the **Files Comparison** preferences page where you can choose to ignore certain types of markup and configure various options.

The resulting comparison will show you differences between the two files, as well as differences between either of them and the base (ancestor) file. The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes.

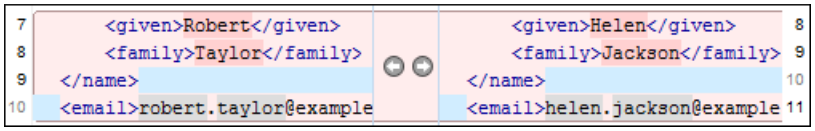


Figure 633: Three-Way Differences

Highlighting Colors

The differences are also highlighted in several colors, depending on the type of change, and dynamic lines connect the compared fragments in the middle section between the two panes. The highlighting colors can be customized in the ***Files Comparison / Appearance*** *preferences page*, but the default colors and their shades mean the following:

- **Pink** - Identifies blocks of changes that include conflicts.
- **Gray** - Identifies your outgoing changes that do not include conflicts.
- **Blue** - Identifies incoming changes that do not include conflicts.
- **Lighter Shade** - Identifies blocks of changes that can be merged in their entirety.
- **Darker Shade** - Identifies specific changes within the blocks that can be merged more precisely.

Navigate Differences

To navigate through differences, do one of the following:

- Use the navigation buttons on the toolbar (or in the **Compare** menu).
- Select a block of differences by clicking its small colored marker in the overview ruler located in the right-most part of the window. At the top of the overview ruler there is a success indicator that turns green where there are no differences, or red if differences are found.
- Click a colored area in between the two text editors.

Editing Actions

You can edit the files directly in either editing pane. The two editors are constantly synchronized and the differences are refreshed when you save the modified document or when you click the **Perform File Differencing** button.

A variety of actions are available on the *toolbar* and in the *various menus* (these same actions are also available in the contextual menu in both editing panes). The tool also includes some inline actions to help you merge, copy, or remove changes. When you select a change, the following inline action widgets are available, depending on the type of change:

Append left change to right and Append right change to left

Copies the content of the selected change from one side and appends it on the other, according to the content of the corresponding change. As a result, the side where the arrow points to will contain the changes from both sides.

Copy change from left to right and **Copy change from right to left**

Replaces the content of a change from one side with the content of the corresponding change from the other side.

Remove change

Rejects the change on the particular side and preserves the particular content on the other side.

Three-Way Diff Algorithms

Oxygen XML Editor offers the following three-way diff algorithms to compare files:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Second Level Comparisons

For both two-way and three-way comparisons, Oxygen XML Editor automatically performs a second level comparison for the **Lines**, **XML Fast**, and **XML Accurate** algorithms. After the first comparison is finished, the second level comparisons for the **Lines** algorithm is processed on text nodes using a word level comparison, meaning that it looks for identical words. For the **XML Fast** and **XML Accurate** algorithms, the second level comparison is processed using a *syntax-aware comparison*, meaning that it looks for identical *tokens*. This second level comparison makes it easier to spot precise differences and you can merge or reject the precise modifications.

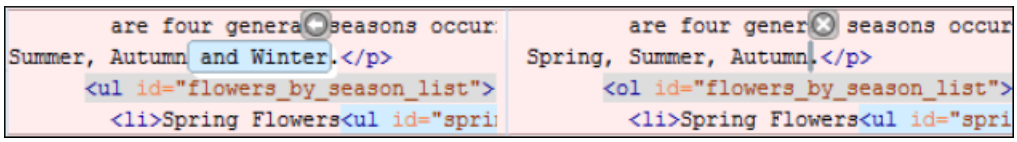


Figure 634: Second Level Diff Comparison

Note: If a modified text fragment contains XML markup (such as processing instructions, XML comments, CData, or elements), the second level comparison will not automatically be performed. In this case you can manually select a second level comparison by doing a word level or character level comparison.

To do a word level comparison, select **Show word level details** from the contextual menu or **Compare** menu.

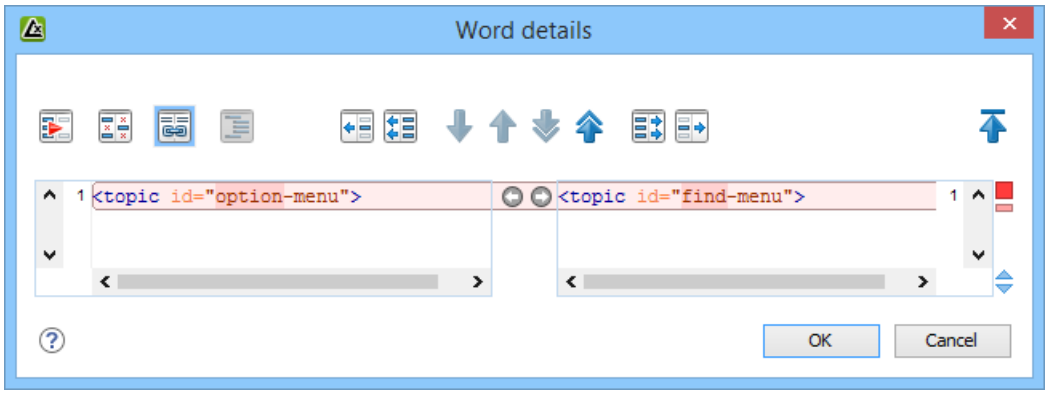


Figure 635: Word Level Comparison

To do a character level comparison, select **Show Character Level details** from the contextual menu or **Compare** menu.

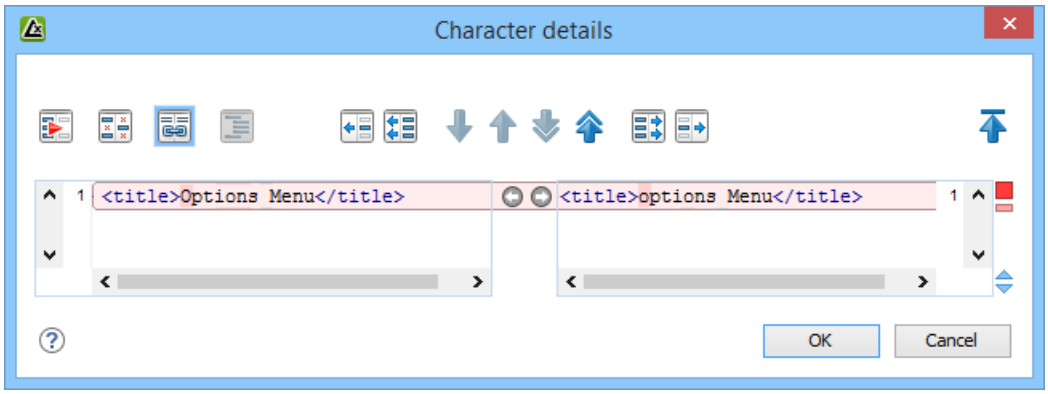


Figure 636: Character Level Comparison

Related Information:

Files Comparison Preferences Page on page 145
Compare Directories on page 753

Toolbar and Contextual Menu Actions of the Compare Files Tool

The toolbar of the **Compare Files** tool contains operations that can be performed on the source and target files or XML fragments. Many of the actions are also available in the contextual menu.

Figure 637: Compare Toolbar

The following actions are available:

Algorithm

This drop-down menu allows you to select one of the following diff algorithms (depending on whether it is a two-way or three-way comparison):

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Characters** - Computes the differences at character level, meaning that it compares two files or fragments looking for identical characters.
- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words.
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **Syntax Aware** - Computes differences for the file types or fragments known by Oxygen XML Editor, taking the syntax (the specific types of tokens) into consideration.
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Diff Options

Opens the ***Files Comparison*** *preferences page* where you can configure various options.

Three-Way Comparison

Toggle action that allows you to perform a three-way comparison between the two files displayed in the two editing panes and a base (ancestor) file.

Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.

Ignore Whitespaces

Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces.

Synchronized scrolling

Toggles synchronized scrolling on or off so that a selected difference can be seen on both sides of the application window. This option is on by default.

Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on OS X))

Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.

Note: When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.

Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.

Copy All Changes from Right to Left

Copies all changes from the file in the right panel to the file in the left panel.

Next Block of Changes (Ctrl + Period (Command + Period on OS X))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

Note: A change block groups one or more consecutive lines that contain at least one change.

Previous Block of Changes (Ctrl + Comma (Command + Comma on OS X))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

Next Change (Ctrl + Shift + Period (Command + Shift + Period on OS X))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

Previous Change (Ctrl + Shift + Comma (Command + Shift + M on OS X))

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

Copy All Changes from Left to Right

Copies all changes from the file in the left panel to the file in the right panel.

Copy Change from Left to Right

Copies the selected difference from the file in the left panel to the file in the right panel.

Ignore Nodes by XPath

You can use this text field to enter an *XPath expression* to ignore certain nodes from the comparison. It will be processed as XPath version 2.0. You can also enter the name of the node to ignore all nodes with the specified name (for example, if you want to ignore all ID attributes from the document, you could simply enter `@id`). This field is only available when comparing XML documents using the **XML Fast** or **XML Accurate** algorithms.

Note: If an XPath expression is specified in the ***Ignore nodes by XPath*** *option* in the **Diff / File Comparison** preferences page, that one is used as a default when the application is started. If you then enter an

expression in this field on the toolbar, this one will be used instead of the default. If you delete the expression from this field, neither will be used.

First Change (Ctrl + B (Command + B on OS X))

Jumps to the first change.

Base

Available for *three-way comparisons*. It is the base file that will be compared with the files opened in the left and right editors. You can specify the path to the file by using the text field, its history drop-down, or the browsing tools in the **Browse** drop-down menu.

Left-Side (Source) File

You can specify the path to the file to be compared on the left side (source) by using the text field, its history drop-down, or the browsing tools in the **Browse** drop-down menu.

Save

Saves the changes made in the source (left-side) file.

Reload

Reloads the source (left-side) file.

Close

Closes the source (left-side) file.

Right-Side (Target) File

You can specify the path to the file to be compared on the right side (target) by using the text field, its history drop-down, or the browsing tools in the **Browse** drop-down menu.

Save

Saves the target (right-side) file.

Reload

Reloads the target (right-side) file.

Close

Closes the target (right-side) file.

Compare Files Tool Menus

The menus in the **Compare Files** tool contain some of the same actions that are on the toolbar, as well as some common actions that are identical to the same actions in the Oxygen XML Editor menus. The menu actions include:

File Menu

Source > Open

Browses for a file that will be displayed in the left panel.

Source > Open URL

Browses for a remote file that will be displayed in the left panel.

Source > Open File from Archive

Browses an archive for a file that will be displayed in the left panel.

Source > Reload

Reloads the file in the left panel.

Source > Save

Saves the changes made to the file in the left panel.

Source > Save As

Allows you to choose a destination to save the file in the left panel.

Source > Close

Closes the file in the left panel.

Target > Open

Browses for a file that will be displayed in the right panel.

Target > Open URL

Browses for a remote file that will be displayed in the right panel.

Target > Open File from Archive

Browses an archive for a file that will be displayed in the right panel.

Target > Reload

Reloads the file in the right panel.

Target > Save

Saves the changes made to the file in the right panel.

Target > Save As

Allows you to choose a destination to save the file in the right panel.

Target > Close

Closes the file in the right panel.

Base > Open

Browses for a file that will be compared with both files in a *three-way comparison*.

Base > Open URL

Browses for a remote file that will be compared with both files in a *three-way comparison*.

Base > Open File from Archive

Browses an archive for a file that will be compared with both files in a *three-way comparison*.

Close (Ctrl + W (Command + W on OS X))

Closes the application.

Edit Menu

Cut

Cut the selection from the currently focused editor panel to the clipboard.

Copy

Copy the selection from the currently focused editor panel to the clipboard.

Paste

Paste content from the clipboard into the currently focused editor panel.

Select all

Selects all content in the currently focused editor panel.

Undo

Undo changes in the currently focused editor panel.

Redo

Redo changes in the currently focused editor panel.

Find Menu

Find/Replace

Perform *find/replace* operations in the currently focused editor panel.

Find Next

Go to the next match using the same options as the last *find* operation. This action runs in both editor panels.

Find Previous

Go to the previous match using the same options as the last *find* operation. This action runs in both editor panels.

Compare Menu

Three-Way Comparison

Toggle action that allows you to perform a three-way comparison between the two files displayed in the two editing panes and a base (ancestor) file.

Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.

Next Block of Changes (Ctrl + Period (Command + Period on OS X))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

Note: A change block groups one or more consecutive lines that contain at least one change.

Previous Block of Changes (Ctrl + Comma (Command + Comma on OS X))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

Next Change (Ctrl + Shift + Period (Command + Shift + Period on OS X))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

Previous Change (Ctrl + Shift + Comma (Command + Shift + M on OS X))

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

Last Change (Ctrl + E (Command + E on OS X))

Jumps to the last change.

First Change (Ctrl + B (Command + B on OS X))

Jumps to the first change.

Copy All Changes from Right to Left

Copies all changes from the file in the right panel to the file in the left panel.

Copy All Changes from Left to Right

Copies all changes from the file in the left panel to the file in the right panel.

Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.

Copy Change from Left to Right

Copies the selected difference from the file in the left panel to the file in the right panel.

Show Word Level Details

Provides a word-level comparison of the selected change.

Show Character Level Details

Provides a character-level comparison of the selected change.

Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on OS X))

Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.

Note: When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.

Options Menu

Preferences

Opens the preferences dialog box that includes numerous pages of options that can be configured.

Menu Shortcut Keys

Opens the **Menu Shortcut Keys** option page where you can configure keyboard shortcuts available for menu items.

Reset Global Options

Resets options to their default values. Note that this option appears only when the tool is executed as a stand-alone application.

Import Global Options

Allows you to import an options set that you have previously exported.

Export Global Options

Allows you to export the current options set to a file.

Help Menu

Help (F1)

Opens a **Help** dialog box that displays the User Manual at a section that is appropriate for the context of the current cursor position.

Use Online Help

If this option is selected, when you select Help or press F1 while hovering over any part of the interface, Oxygen XML Editor attempts to open the help documentation in online mode. If this option is not selected or an internet connection fails, the help documentation is opened in offline mode.

Report problem

Opens a dialog box that allows the user to write the description of a problem that was encountered while using the application. You can change the URL where the reported problem is sent by using the `com.oxygenxml.report.problems.url` system property. The report is sent in XML format through the `report` parameter of the POST HTTP method.

Support Center

Opens the Oxygen XML Editor Support Center web page in a browser.

Compare Directories

The **Compare Directories** tool can be used to compare and manage changes to files and folders within the structure of your directories. The utility is available from the **Tools** menu or can be opened as a stand-alone application from the Oxygen XML Editor installation folder (`diffDirs.exe`).

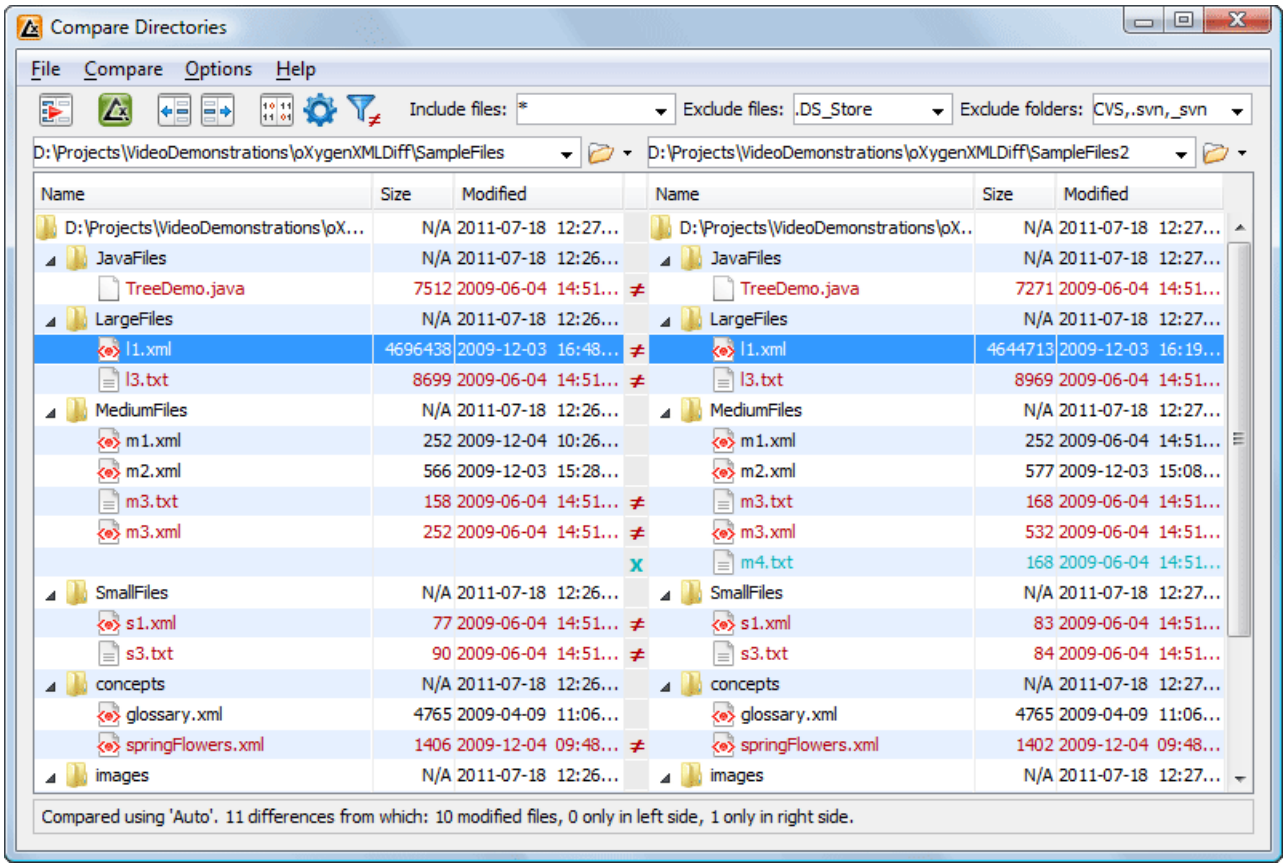


Figure 638: Compare Directories Tool

Starting the Tool from a Command Line

The directory comparison tool can also be started by using command-line arguments. In the installation folder there is an executable shell (`diffDirs.bat` on Windows, `diffDirs.sh` on Unix/Linux, `diffDirsMac.sh` on OS X). To specify the directories to compare, you can pass command-line arguments using the following construct: `diffDirs.bat/diffDirs.sh/diffDirsMac.sh [directory path 1] [directory path 2]`.

If you pass only one argument, you are prompted to manually choose the second directory or archive.

For example, to start a comparison between two Windows directories, the command line would look like this:

```
diffDirs.bat "c:\documents new" "c:\documents old"
```

Tip: If there are spaces in the path names, surround the paths with quotes.

Directory Comparisons

To perform a directory comparison, follow these steps:

1. Select a folder in the left panel and the folder you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the **Browse for local directory** action in the **Browse** drop-down menu.

   Step Result: The selected directory structures are opened in the two side-by-side panels.
2. To highlight the differences between the two folders, click the **Perform Directories Differencing** button from the toolbar.
3. You can also use the **Diff Options** button to access the ***Directories Comparison*** *preferences page* where you can configure various options.

To compare the content of two archives, follow these steps:

1. Use the **Browse for archive file** action in the **Browse** drop-down menu to select the archives in the left and right panels.
2. By default, the supported archives are not treated as directories and the comparison is not performed on the files inside them. To make Oxygen XML Editor treat supported archives as directories, select the ***Look in archives*** *option* in the **Directories Comparison** preferences page.
3. To highlight the differences, click the **Perform Directories Differencing** button from the toolbar.

The directory comparison results are presented using two tree-like structures showing the files and folders, including their name, size, and modification date. A column that contains graphic symbols separates the two tree-like structures. The graphic symbols can be one of the following:

- An **X** symbol, when a file or a folder exists in only one of the compared directories.
- A ≠symbol, when a file exists in both directories but the content differs. The same sign appears when a collapsed folder contains differing files.

The color used for the symbol and the directory or file name can be customized in the ***Directories Comparison / Appearance*** *preferences page*. You can double-click lines marked with the ≠ symbol to open a **Compare Files** window, which shows the differences between the two files.

The directories that contain files that differ are expanded automatically so that you can focus directly on the differences. You can merge the contents of the directories by using the copy actions. If you double-click (or press **Enter**) on a line with a pair of files, Oxygen XML Editor starts a *file comparison* between the two files, using the **Compare Files** tool.

Related Information:

Compare Files on page 742

Toolbar and Contextual Menu Actions of the Compare Directories Tool

The toolbar of the **Compare Directories** tool contains operations that can be performed on the compared directory structure. Some of the toolbar actions are also available in the contextual menu.

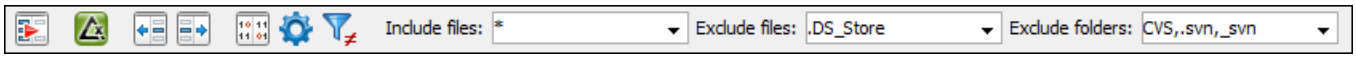


Figure 639: Compare toolbar

Toolbar Actions

Perform Directories Differencing

Looks for differences between the two directories displayed in the left and right side of the application window.

Perform Files Differencing

Opens the ***Compare Files*** *tool* that allows you to compare the currently selected files.

Copy Change from Right to Left

Copies the selected change from the right side to the left side (if there is no file/folder in the right side, the left file/folder is deleted).

Copy Change from Left to Right

Copies the selected change from the left side to the right side (if there is no file/folder in the left side, the right file/folder is deleted).

Binary Compare

Performs a byte-level comparison on the selected files.

Diff Options

Opens the ***Directory Comparison*** *preferences page* where you can configure various options.

Show Only Modifications

Displays a more uncluttered file structure by hiding all identical files.

File and folder filters

Differences can be filtered using three combo boxes: **Include files**, **Exclude files**, and **Exclude folders**. They come with predefined values and are editable to allow custom values. All of them accept multiple comma-separated values and the ***** and **?** wildcards. For example, to filter out all JPEG and GIF image files, edit the **Exclude files** filter box to read **`*.jpeg`**, **`*.png`**. Each filter includes a drop-down menu with the latest 15 filters applied.

Contextual Menu Actions

Perform Files Differencing

Opens the ***Compare Files*** *tool* that allows you to compare the currently selected files.

Binary Compare

Performs a byte-level comparison on the selected files.

Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.

Copy Change from Left to Right

Copies the selected difference from the file in the left panel to the file in the right panel.

Open

If the action is invoked on a file, the selected file is opened in Oxygen XML Editor. If the action is invoked on a directory, the selected directory is opened in the default file browser for your particular operating system.

Open in System Application

Opens the selected file in the system application that is associated with that type of file. The action is available when launching the **Compare Directories** tool from the **Tools** menu in Oxygen XML Editor.

Show in Explorer

Opens the default file browser for your particular operating system with the selected file highlighted.

Compare Directories Tool Menus

The menus in the **Compare Directories** tool contain some of the same actions that are on the toolbar, as well as some common actions that are identical to the same actions in the Oxygen XML Editor menus. The menu actions include:

File Menu

Close (Ctrl + W (Command + W on OS X))

Closes the application.

Compare Menu

Perform Directories Differencing

Looks for differences between the two directories displayed in the left and right side of the application window.

Perform Files Differencing

Opens the ***Compare Files*** *tool* that allows you to compare the currently selected files.

Copy Change from Right to Left

Copies the selected change from the right side to the left side (if there is no file/folder in the right side, the left file/folder is deleted).

Copy Change from Left to Right

Copies the selected change from the left side to the right side (if there is no file/folder in the left side, the right file/folder is deleted).

Options Menu

Preferences

Opens the preferences dialog box that includes numerous pages of options that can be configured.

Menu Shortcut Keys

Opens the **Menu Shortcut Keys** option page where you can configure keyboard shortcuts available for menu items.

Reset Global Options

Resets options to their default values. Note that this option appears only when the tool is executed as a stand-alone application.

Import Global Options

Allows you to import an options set that you have previously exported.

Export Global Options

Allows you to export the current options set to a file.

Help Menu

Help (F1)

Opens a **Help** dialog box that displays the User Manual at a section that is appropriate for the context of the current cursor position.

Use Online Help

If this option is selected, when you select Help or press F1 while hovering over any part of the interface, Oxygen XML Editor attempts to open the help documentation in online mode. If this option is not selected or an internet connection fails, the help documentation is opened in offline mode.

Report problem

Opens a dialog box that allows the user to write the description of a problem that was encountered while using the application. You can change the URL where the reported problem is sent by using the `com.oxygenxml.report.problems.url` system property. The report is sent in XML format through the `report` parameter of the POST HTTP method.

Support Center

Opens the Oxygen XML Editor Support Center web page in a browser.

Compare Images

You can use the **Compare Directories** tool to compare images. If you double-click a line that contains two different images, the **Compare images** window is displayed. This dialog box presents the images in the left and right sides, scaled to fit the available view area. You can use the contextual menu actions to scale the images to their original size or scale them down to fit in the view area.

The supported image types are: *GIF*, *JPG*, *JPEG*, *PNG*, and *BMP*.

Compare Directories Against a Base (3-Way)

The **Compare Directories Against a Base (3-way)** tool allows you to perform three-way comparisons on directories to help you identify and merge changes between multiple modifications of the same directory structure. It is especially helpful for teams that have multiple authors contributing documents to the same directory system. It offers information about conflicts and changes, and includes actions to easily merge, accept, overwrite, or ignore changes to the directory system.

How to Perform 3-Way Directory Comparisons

To perform a 3-way directories comparison, follow these steps:

1. Select **Compare Directories Against a Base (3-way)** from the **Tools** menu.

   Step Result: This opens a dialog box that allows you to select the 3 file sets that will be used for the comparison.

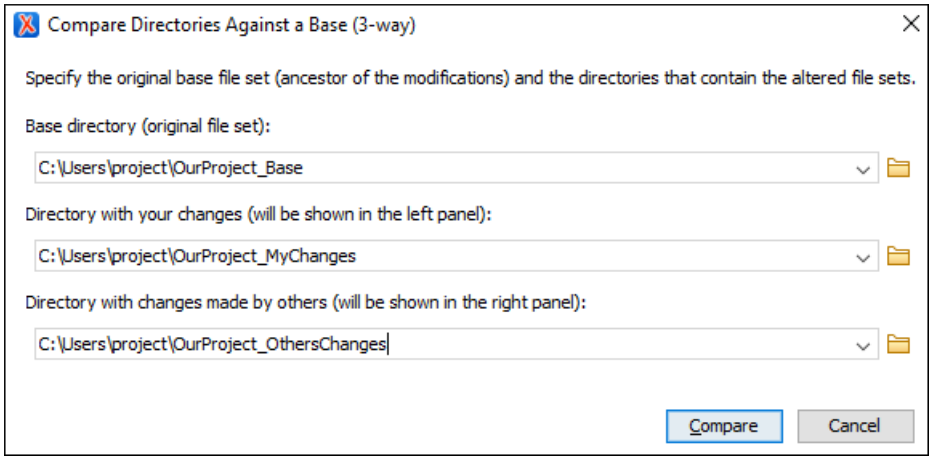


   Figure 640: Compare Directories Against a Base File Set Chooser

2. Select the file sets to be compared:
   - **Base directory** - This is the original (base) file set before any modifications were made by your or others.
   - **Directory with your changes** - This is the file set with changes that you have made. This file set will be displayed in the left panel in the comparison tool.
   - **Directory with changes made by others** - This is the file set with changes made by others that you want to merge with your changes. This file set will be displayed in the right panel in the comparison tool.
3. Click the **Compare** button to compare the file sets and open the comparison and merge tool.
4. Use the features and actions described in the next section to identify and merge the changes.

3-Way Directory Comparison and Merge Tool

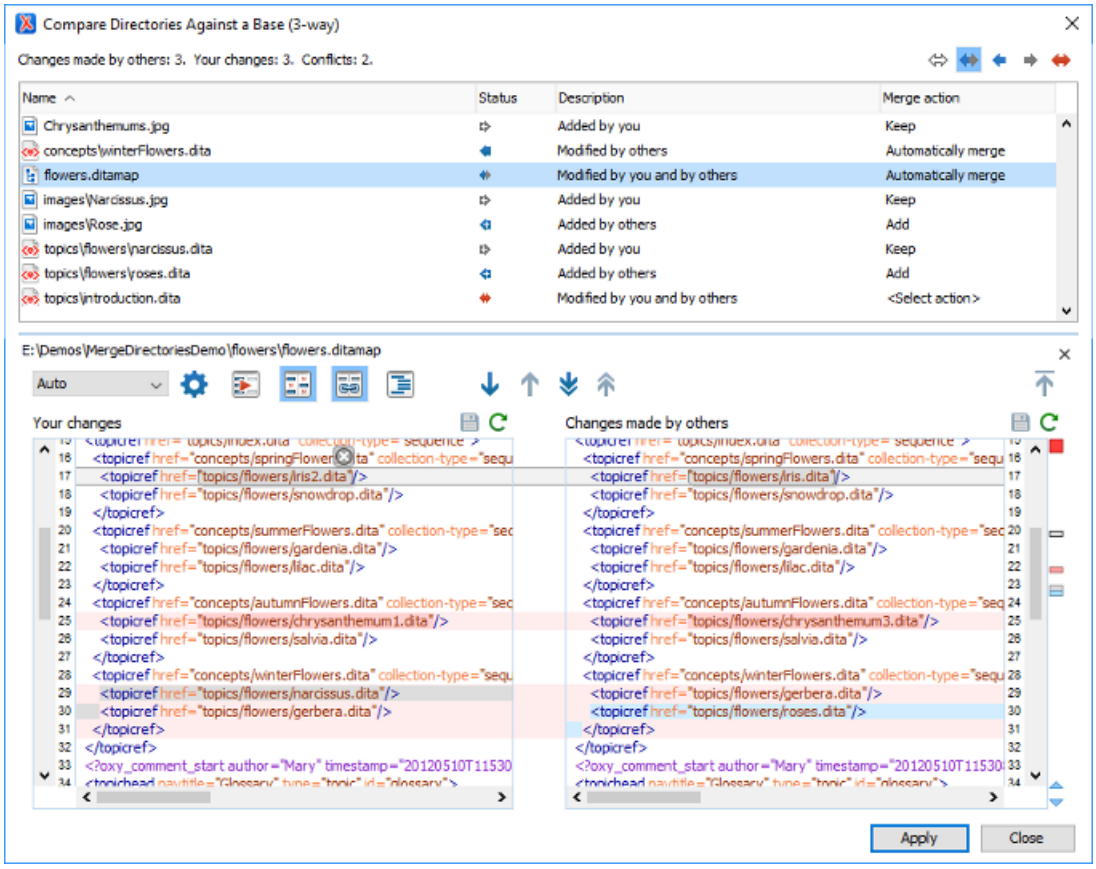


Figure 641: Comparison and Merge Tool

The 3-way directory comparison and merge tool includes the following information, features, and actions:

Number of Changes and Conflicts

The first thing you see in top-left corner of the tool is grand total of all the changes made by others, changes made by you, and the number of conflicts.

Filter Buttons

In the top-right corner you can use the toggle buttons to filter the list of modifications:

Show all files

Use this button to show all modified and unmodified files, as well as conflicts.

Show only files modified by you and others

Filters the list to show all files that have been modified, including conflicts.

Show only files modified by others

Filters the list to only show the files that were modified by others.

Show only files modified by you

Filters the list to only show the files that were modified by you.

Show only conflicting files

Filters the list to only show files that contain conflicts.

List of Files Panel

This panel shows the list of files in the compared file sets based upon the filter button that is selected. This panel includes the following sortable columns:

- **Name** - The file names.

- **Status** - An icon that represents the file status. Red icons indicate some sort of conflict. Gray icons indicate modifications made by you. Blue icons indicate modifications made by others.
- **Description** - A description of the file status.
- **Merge Action** - This column provides a drop-down menu for each file that allows you to choose some merge actions depending upon its status. A default action is always set to automatically merge the changes made by others with your changes. If there is a conflict, the default is **<Select action>** and you are required to make a selection. Click this column to access the drop-down menu where you can make a selection. The same actions are available in the contextual menu.

You can double-click any non-binary file (or select **Show modifications** from the contextual menu) to open it in the file comparison panel (the file from your file set is shown in the left panel while the file from the file set with changes made by others is shown in the right panel).

File Comparison Panels

If you double-click any non-binary file in the top panel, the file is opened in this file comparison section. The file from your file set is shown in the left panel and the file from the other file set is shown in the right panel.

Note: If Oxygen XML Editor does not recognize the file type, a dialog box will be displayed that allows you to select the type of editor you want it to be associated with for this comparison (if you want Oxygen XML Editor to remember this association, you can select the **Associate file type with editor** option at the bottom of the dialog box).

This panel includes the following information and toolbar actions:

File Path

The first thing you see in this panel is the file path where merge actions will be applied if you make changes.

Close

Closes the file comparison panel.

Algorithm Drop-Down Menu

This drop-down menu allows you to select one of the following diff algorithms to be used for file comparisons:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Diff Options

Opens the ***Files Comparison*** *preferences page* where you can configure various options.

Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.

Ignore Whitespaces

Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces.

Synchronized scrolling

Toggles synchronized scrolling. When toggled on, a selected difference can be seen in both panels.

Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on OS X))

Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.

Note: When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.

Next Block of Changes (Ctrl + Period (Command + Period on OS X))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

Note: A change block groups one or more consecutive lines that contain at least one change.

Previous Block of Changes (Ctrl + Comma (Command + Comma on OS X))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

Next Change (Ctrl + Shift + Period (Command + Shift + Period on OS X))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

Previous Change (Ctrl + Shift + Comma (Command + Shift + M on OS X))

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

First Change (Ctrl + B (Command + B on OS X))

Jumps to the first change.

Left-Side File (Your changes)

Above the panel you can see the file path and the following two buttons:

Save

Saves changes made to the file.

Reload

Reloads the file.

Right-Side File (Changes made by others)

Above the panel you can see the file path and the following two buttons:

Reload

Reloads the file.

Displaying Changes in the File Comparison Panels

The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes.

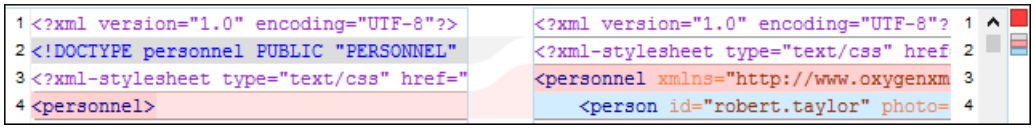


Figure 642: File Comparison Panels

The differences are also highlighted in several colors, depending on the type of change, and dynamic lines connect the compared fragments in the middle section between the two panes. The highlighting colors can be customized in the ***Files Comparison / Appearance*** *preferences page*, but the default colors and their shades mean the following:

- **Pink** - Identifies modifications on either side.
- **Gray** - Identifies an addition of a node in the left side (your outgoing changes).
- **Blue** - Identifies an addition of a node in the right side (incoming changes).
- **Lighter Shade** - Identifies blocks of changes that can be merged in their entirety.

- **Darker Shade** - Identifies specific changes within the blocks that can be merged more precisely.

Direct Editing Actions in the File Comparison Panels

In addition to selecting merge actions from the drop-down menus in the **Merge Action** column in the top panel, you can also edit the files directly in the left pane (your local changes). The two editors are constantly synchronized and the differences are refreshed when you save the modified document (**Save** button or **Ctrl +S**) or when you click the **Perform File Differencing** button.

A variety of actions are available in the contextual menu in both editing panes. The tool also includes some inline actions to help you merge, copy, or remove changes. When you select a change, the following inline action widgets are available, depending on the type of change:

Append right change to left

Copies the content of the selected change from the right side and appends it on the left side.

Copy change from right to left

Replaces the content of a change in the left side with the content of the change in the right side.

Remove change

Removes the change from the left side.

Any time you save manual changes (**Save** button or **Ctrl+S**), the selection in the **Merge Action** column in the top panel automatically changes to **Use merged** and a copy of the original file is kept so that you can revert to the original file if necessary. To discard your manual changes and revert to your original changes, select a different action in the **Merge Action** drop-down menu.

Applying Changes

When you click the **Apply** button, all the merge actions you have selected and the changes you have made will be processed.

If there are unresolved conflicts (conflicts where no merge action is selected in the **Merge Action** drop-down menu), a dialog box will be displayed that allows you to choose how to solve the conflicts. You can choose between the following:

- **Keep your changes** - If you select this option and then click **Apply**, your local changes will be preserved for the unresolved conflicts.
- **Overwrite your changes** - If you select this option and then click **Apply**, your local changes will be overwritten with the changes made by others, for the unresolved conflicts.
- **Cancel** - You can click the **Cancel** button to go back to the merge tool to resolve the conflicts individually.

Cancelling Changes

If you click the **Cancel** button at the bottom of the merge tool, all the changes you made in the tool will be lost.

Related Information:

Compare Directories on page 753
Compare Files on page 742

Syncro SVN Client

The Syncro SVN Client is a client application for the Apache Subversion™ version control system, compatible with Subversion 1.6, 1.7, and 1.8 servers. It manages files and directories that change over time and are stored in a central repository. The version control repository is much like an ordinary file server, except that it remembers every change ever made to your files and directories. This allows you to access older versions of your files and examine the history of how and when your data changed.

To start Syncro SVN Client, go to **Tools** > **SVN Client**.

Main Window

This section explains the main window of Syncro SVN Client.

Views

The main window consists of the following views:

- ***Repositories*** *view* - Allows you to define and manage Apache Subversion™ repository locations.
- ***Working Copy*** *view* - Allows you to manage with ease the content of the working copy.
- ***History*** *view* - Displays information (author name, revision number, commit message) about the changes made to a resource during a specified period of time.
- ***Editor*** *view* - Allows you to edit various types of text files, with full syntax-highlight.
- ***Annotations*** *view* - Displays a list with information regarding the structure of a document (author and revision for each line of text).
- ***Compare*** *view* - Displays the differences between two revisions of a text file from the working copy.
- ***Image Preview*** *panel* - Allows you to preview standard image files supported by Syncro SVN Client: JPG, GIF and PNG.
- ***Compare Images*** *view* - Displays two images side by side.
- ***Properties*** *view* - Displays the SVN properties of a resource under version control.
- ***Console*** *view* - Displays information about the currently running operation, similar with the output of the Subversion command line client.
- ***Dynamic Help*** *view* - Shows information about the currently selected view.

The main window's status bar presents in the left side the operation in progress or the final result of the last performed action. In the right side there is a progress bar for the running operation and a stop button to cancel the operation.

SVN Main Menu

The main menu of the Syncro SVN Client is composed of the following menus:

File Menu

New submenu:

New File

This operation creates a new file as a child of the selected folder from the ***Repositories*** *view* tree or the ***Working Copy*** *view* tree, depending on the view that was last used. Note that for the ***Working Copy*** *view*, the file is added to *version control* only if the selected folder is under *version control*.

New Folder(Ctrl (Command on OS X) + Shift + F)

This operation creates a new folder as a child of the selected folder from the ***Repositories*** *view* tree or the ***Working Copy*** *view* tree, depending on the view that was last used. Note that for the ***Working Copy*** *view*, the file is added to *version control* only if the selected folder is under *version control*.

New External Folder (Ctrl (Command on OS X) + Shift + W)

This operation allows you to add a new external definition on the selected folder. An external definition is a mapping of a local directory to *a URL of a versioned directory*, and ideally a particular revision, stored in the `svn:externals` property of the selected folder.

Tip: You can specify a particular revision of the external item by using a *peg revision* at the end of the URL (for example, URL`@rev1234`). You can also use peg revisions to access external items that were deleted, moved, or replaced.

The URL used in the external definition format can be relative. You can specify the repository URL that the external folder points to by using one of the following relative formats:

- **../** - Relative to the URL of the directory that the `svn:externals` property is set.
- **^/** - Relative to the root of the repository in which the `svn:externals` property is versioned.
- **//** - Relative to the scheme of the URL of the directory that the `svn:externals` property is set.
- **/** - Relative to the root URL of the server in which the `svn:externals` property is versioned.

Important: To change the target URL of an external definition, or to delete an external item, do the following:

1. Modify or delete the item definition found in the `svn:externals` property that is set on the parent folder.

2. For the change to take effect, use the **Update** operation on the parent folder of the external item.

Note: Syncro SVN Client does not support definitions of local relative external items.

Open (Ctrl (Command on OS X) + O)

This action opens the selected file in an editor where you can modify it. The action is active only when a single item is selected. The action opens a file with the internal editor or the external application associated with that file type. This action works on any file selection from the ***Repositories*** *view*, ***Working Copy*** *view*, ***History*** *view*, or ***Directory Change Set*** *view*, depending on the view that was last used to invoke it. In the case of a folder, the action opens the selected folder with the system application for folders (for example, Windows Explorer on Windows or Finder on OS X, etc). Note that opening folders is available only for folders selected in the ***Working Copy*** *view*.

Open with(Ctrl (Command on OS X) + Shift + O)

Displays the **Open with** dialog box for specifying the editor in which the selected file is opened. If multiple files are selected only external applications can be used to open the files. This action works on any file selection from ***Repositories*** *view*, ***Working Copy*** *view*, ***History*** *view*, or ***Directory Change Set*** *view*, depending on the view that was last used to invoke it.

Show in Explorer/Show in Finder

Opens the parent directory of the selected working copy file and selects the file.

Save (Ctrl (Command on OS X) + S)

Saves the local file currently opened in the editor or the **Compare** view.

Save as

Saves any file selected in the **Repositories**, **History**, or **Directory Change Set** view.

Copy URL Location (Ctrl (Command on OS X) + Alt + U)

Copies the URL location of the resource currently selected in the **Repositories** view to clipboard.

Copy to

Copies the currently selected resource, either in **Repositories** or **Working copy** view, to a specified location.

Note: This action can also be used from **History** and **Directory Change Set** views to recover older versions of a repository item.

Move to(Ctrl (Command on OS X) + M)

Moves the currently selected resource, either in **Repositories** or **Working copy** view, to a specified location.

Rename(F2)

Renames the resource currently selected, either in **Repositories** or **Working copy** view.

Delete (Delete)

Deletes the resource currently selected either, in **Repositories** or **Working copy** view.

Locking:

- **Scan for locks (Ctrl (Command on OS X) + L)** - Contacts the repository and recursively obtains the list of locks for the selected resources. A dialog box containing the locked files and the lock description will be displayed. This is only active for resources under *version control*. For more details see *Scanning for locks*.
- **Lock (Ctrl (Command on OS X) + K)** - Allows you to lock certain files that need exclusive access. You can write a comment describing the reason for the lock and you can also force (*steal*) the lock. This action is active only on files under *version control*. For more details on the use of this action see *Locking a file*.
- **Unlock (Ctrl (Command on OS X) + Alt + K)** - Releases the exclusive access to a file from the repository. You can also choose to unlock it by force (*break the lock*).

Show SVN Properties (Ctrl (Command on OS X) + P)

Opens the ***Properties*** *view* and displays the SVN properties for a selected resource from ***Repositories*** *view* or ***Working Copy*** *view*, depending on the view that was last used to invoke it.

Show SVN Information (Ctrl (Command on OS X) + I)

Provides additional information for a selected resource. For more details, go to *Obtain information for a resource*.

Exit (Ctrl (Command on OS X) + Q)

Closes the application.

Edit Menu

Undo (Ctrl (Command on OS X) + Z)

Undo edit changes in the local file that is currently opened in the editor or the **Compare** view.

Redo (Ctrl (Command on OS X) + Y)

Redo edit changes in the local file that is currently opened in the editor or the **Compare** view.

Cut (Ctrl (Command on OS X) + X)

Cut selection from the local file that is currently opened in the editor view or the **Compare** view to clipboard.

Copy (Ctrl (Command on OS X) + C)

Copy selection from the local file that is currently opened in the editor or the **Compare** view to clipboard.

Paste (Ctrl (Command on OS X) + V)

Paste selection from clipboard into the local file that is currently opened in editor or the **Compare** view.

Find/Replace (Ctrl (Command on OS X) + F)

Perform find and replace operations in the local file that is currently opened in the editor or the **Compare** view.

Find Next (F3)

Go to the next match using the same find options of the last find operation. This action runs in the editor panel and in any non-editable text area (for example, the **Console** view).

Find Previous (Shift + F3)

Go to the previous match using the same find options of the last find operation. This action runs in the editor panel and in any non-editable text area (for example, the **Console** view).

Repository Menu

New Repository Location (Ctrl + Alt + N (Command + Alt + N on OS X))

Displays the **Add SVN Repository** dialog box. This dialog box allows you to define a new repository location.

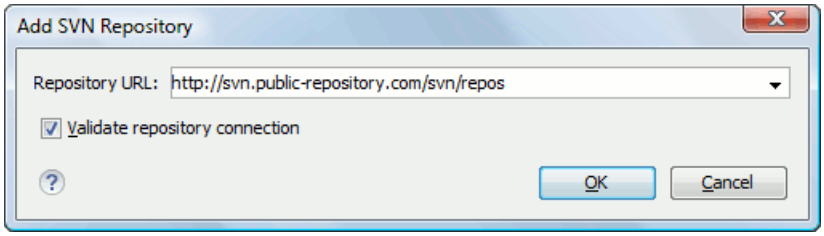


Figure 643: Add SVN Repository Dialog Box

If the **Validate repository connection** option is selected, the URL connection is validated before being added to the **Repositories** view.

Edit Repository Location (Ctrl + Alt + E (Command + Alt + E on OS X))

Context-dependent action that allows you to edit the selected repository location using the **Edit SVN Repository** dialog box. It is active only when a repository location root is selected.

Change the Revision to Browse (Ctrl + Alt + B (Command + Alt + B on OS X))

Context-dependent action that allows you to change the selected repository revision using the **Change the Revision to Browse** dialog box. It is active only when a repository location root is selected.

Remove Repository Location (Ctrl + Alt + R (Command + Alt + R on OS X))

Allows you to remove the selected repository location from the view. It shows you a confirmation dialog box before removal. It is active only when a repository location root is selected.

Refresh (F5)

Refreshes the resource selected in the **Repositories** view.

Check out (Ctrl + Alt + O (Command + Alt + O on OS X))

Allows you to create a working copy from a repository directory, on your local file system. To read more about this operation, see the section *Check out a working copy*.

Export

Opens *the **Export** dialog box* that allows you to configure options for exporting a folder from the repository to the local file system.

Import:

Import folder (Ctrl + Shift + L (Command + Shift + L on OS X))

Allows you to import the contents of a specified folder from the file system into the selected folder in a repository. To read more about this operation, see the section *Importing resources into a repository*.

Note: The difference between the **Import folder** and **Share project** actions is that the latter also converts the selected directory into a working copy.

Import Files (Ctrl + Shift + I (Command + Shift + I on OS X))

Imports the files selected from the files system into the selected folder in the repository.

Working Copy Menu

(on OS X)Working Copies Manager

Opens a dialog box with a list of working copies that the Apache Subversion™ client is aware of. In this dialog box you can add existing working copies or remove those that are no longer needed.

Switch to

Selects one of the following view modes: **All Files**, **Modified**, **Incoming**, **Outgoing**, or **Conflicts**.

Refresh (F5)

Refreshes the state of the selected resources or of the entire working copy (if there is no selection).

Synchronize (Ctrl (Command on OS X) + Shift + S)

Connects to the repository and determines the working copy and repository changes made to the selected resources. The application switches to **Modified** view mode if the ***Always switch to 'Modified' mode** option* is selected.

Update (Ctrl (Command on OS X) + U)

Updates all the selected resources that have incoming changes to the HEAD revision. If one of the selected resources is a directory then the update for that resource will be recursive.

Update to revision/depth

Allows you to update the selected resources from the working copy to an earlier revision from the repository. You can also select the update *depth* for the current folder. You can find out more about the *depth* term in the *sparse checkouts* section.

Commit

Collects the outgoing changes from the selected resources in the working copy and allows you to choose exactly what resources to commit. A directory will always be committed recursively. Unversioned resources will be deselected by default. In the **Commit** dialog box you can also enter a comment before sending your changes to the repository.

Update all(Ctrl (Command on OS X) + Shift + U)

Updates all resources from the working copy that have incoming changes. It performs a recursive update on the synchronized resources.

Commit all

Commits all the resources with outgoing changes. It is disabled when **Incoming** mode is selected or the synchronization result does not contain resources with outgoing changes. It performs a recursive commit on the synchronized resources.

Revert (Ctrl (Command on OS X) + Shift + V)

Undoes all local changes for the selected resources. It does not contact the repository and the files are obtained from Apache Subversion™ pristine copy. It is available only for modified resources. See *Revert your changes* for more information.

Edit conflict (Ctrl (Command on OS X) + E)

Opens the **Compare** editor, allowing you to modify the content of the currently conflicting resources. For more information about editing conflicts, see *Edit conflicts*.

Mark Resolved (Ctrl (Command on OS X) + Shift + R)

Instructs the Subversion system that you resolved a conflicting resource. For more information, see *Merge conflicts*.

Mark as Merged (Ctrl (Command on OS X) + Shift + M)

Instructs the Subversion system that you resolved the pseudo-conflict by merging the changes and you want to commit the resource. Read the *Merge conflicts* section for more information about how you can solve the pseudo-conflicts.

Override and Update

Drops any outgoing change and replaces the local resource with the HEAD revision. This action is available on resources with outgoing changes, including conflicting ones. See the *Revert your changes* section.

Override and Commit

Drops any incoming changes and sends your local version of the resource to the repository. This action is available on conflicting resources. For more information see *Drop incoming modifications*.

Mark as copied

You can use this action to mark an item from the working copy as a copy of an other item under *version control*, when the copy operation was performed outside of an SVN client. The **Mark as copied** action is available when you select two items (both the new item and source item), and it depends on the state of the source item.

Mark as moved

You can use this action to mark an item from the working copy as being moved from another location of the working copy, when the move operation was performed outside of an SVN client. The **Mark as moved** action is available when you select two items from different locations (both the new item and the source item that is usually reported as *missing*), and it depends on the state of the source item.

Mark as renamed

You can use this action to mark an item from the working copy as being renamed outside of an SVN client. The **Mark as renamed** action is available when you select two items from the same directory (both the new item and the source item that is usually reported as *missing*), and it depends on the state of the source item.

Add to "svn:ignore" (Ctrl (Command on OS X) + Alt + I)

Allows you to add files that should not participate in the *version control* operations inside your working copy . This action can only be performed on resources not under *version control*. It actually modifies the value of the `svn:ignore` property in the parent directory of the resource. Read more about this in the *Ignore Resources Not Under Version Control* section.

Add to version control (Ctrl (Command on OS X) + Alt + V)

Allows you to add resources that are not under *version control*. For further details, see *Add Resources to Version Control* section.

Remove from version control

Schedules selected items for deletion from repository upon the next commit. The items are not removed from the file system after committing.

Clean up (Ctrl (Command on OS X) + Shift + C)

Performs a maintenance cleanup operation on the selected resources from the working copy. This operation removes the Subversion maintenance locks that were left behind. This is useful when you already know where the problem originated and want to fix it as quickly as possible. It is only active for resources under *version control*.

Expand All (Ctrl (Command on OS X) + Alt + X)

Displays all descendants of the selected folder. The same behavior is obtained by double-clicking a collapsed folder.

Collapse all (Ctrl (Command on OS X) + Alt + Z)

Collapses all descendants of the selected folder. The same behavior is obtained by double-clicking a expanded folder.

Compare Menu

Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.

Next Block of Changes (Ctrl + Period (Command + Period on OS X))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

Note: A change block groups one or more consecutive lines that contain at least one change.

Previous Block of Changes (Ctrl + Comma (Command + Comma on OS X))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

Next Change (Ctrl + Shift + Period (Command + Shift + Period on OS X))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

Previous Change (Ctrl + Shift + Comma (Command + Shift + M on OS X))

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

Last Change (Ctrl + E (Command + E on OS X))

Jumps to the last change.

First Change (Ctrl + B (Command + B on OS X))

Jumps to the first change.

Copy All Changes from Right to Left

Copies all changes from the file in the right panel to the file in the left panel.

Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.

Show Word Level Details

Provides a word-level comparison of the selected change.

Show Character Level Details

Provides a character-level comparison of the selected change.

Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on OS X))

Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.

Note: When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.

Ignore Whitespaces

Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces.

History Menu

Show History(Ctrl (Command on OS X) + H)

Displays the history for a SVN resource at a given revision. The resource can be one selected from the **Repositories** view, **Working Copy** view, or from the **Affected Paths** table from the **History** view, depending on which view was last focused when this action was invoked.

Show Annotation (Ctrl + Shift + A (Command + Shift + A on OS X))

Opens the **Show Annotation** dialog box that computes *the annotations for a file and displays them in the **Annotations** view*, along with the history of the file in the **History** view.

Repositories

This operation is available for any resource selected from view, **Working Copy** view, **History** view or **Directory Change Sets** view, depending on which view was last focused when this action was invoked.

Revision Graph (Ctrl (Command on OS X) + G)

This action allows you to see the graphical representation of a resource's history. For more details about a resource's revision graph see the section *Revision Graph*. This operation is available for any resource selected in the **Repositories** view or **Working Copy** view.

Tools Menu

Share project

Allows you to *share a new project* using an SVN repository. The local project is automatically converted into an SVN working copy.

Branch / Tag

Allows you to copy the selected resource from the **Repositories** view or **Working Copy** view to a branch or tag into the repository. To read more about this operation, see the section *Creating a Branch / Tag*.

Merge (Ctrl (Command on OS X) + J)

Allows you to merge the changes made on one branch back into the trunk, or vice versa, using the selected resource from the working copy. To read more about this operation, see the section *Merging*.

Switch (Ctrl (Command on OS X) + Alt + W)

Allows you to change the repository location of a working copy, or only of a versioned item of the working copy, within the same repository. It is available when the selected item of the working copy is a versioned resource, except for *external* items. To read more about this action, see the *Switching the Repository Location* section.

Relocate

Allows you to change the base URL of the root folder of the working copy to a new URL when the base URL of the repository changed. For example, if the repository itself was moved to a different server. This operation is only available for the root item of the working copy. To read more about this operation, see the *Relocate a Working Copy* section.

Create patch (Ctrl (Command on OS X) + Alt + P)

Allows you to create a file containing all the differences between two resources, based on the `svn diff` command. To read more about creating patches, see *the section about patches*.

Working copy format

This submenu contains the following two operations:

Upgrade

Upgrades the format of the currently loaded working copy to the newest one known by Syncro SVN Client. This allows you to benefit of all the new features of the client.

Downgrade

Downgrades the format of the currently loaded working copy to SVN 1.7 format. This is useful if you want to use older SVN clients with the current working copy, or, by mistake, you have upgraded the format of an older working copy to SVN 1.8.

Note: SVN 1.7 working copies cannot be downgraded to older formats.

See the section *Working Copy Format* to read more about this subject.

Options Menu

Preferences

Opens the **Preferences** dialog box.

Menu Shortcut Keys

Opens the ***Menu Shortcut Keys*** *preferences page*, where users can configure in one place the keyboard shortcuts available for menu items available in Syncro SVN Client.

Global Run-Time Configuration

Allows you to configure SVN general options, that should be used by all the SVN clients you may use:

- **Edit 'config' file** - In this file you can configure various SVN client-side behaviors.
- **Edit 'servers' file** - In this file you can configure various server-specific protocol parameters, including HTTP proxy information and HTTP timeout settings.

Export Options

Allows you to export the current options to an XML file.

Import Options

Allows you to import options you have previously exported.

Reset Options

Resets all your options to the default ones.

Reset Authentication

Resets the Subversion authentication information.

Window Menu

Show View

Allows you to select the view you want to bring to front.

Show Toolbar

Allows you to select the toolbar you want to be visible.

Enable flexible layout

Toggles between a fixed and a flexible layout. When the flexible layout is enabled, you can move and dock the internal views to adapt the application to various viewing conditions and personal requirements.

Reset Layout

Resets all the views to their default position.

Help Menu

Help (F1)

Opens the **Help** dialog box.

Use online help (selected by default)

If this option is selected, when you select **Help** or press **F1** while hovering over any part of the interface, Oxygen XML Editor attempts to open the help documentation in online mode. If this option is not selected or an internet connection fails, the help documentation is opened in offline mode.

Show Dynamic Help view

Displays the **Dynamic Help** view.

Report Problem

Opens a dialog box that allows you to write the description of a problem that was encountered while using the application.

Support Center

Opens the Support Center web page in a browser.

About

Opens the **About** dialog box.

SVN Main Toolbar

The toolbar of the Syncro SVN Client SVN Repositories window contains the following actions:

Check out

Checks out a working copy from a repository. The repository URL and the working copy format must be specified.

Synchronize

Synchronizes the current working copy with the repository.

Update All

Updates all resources of the working copy that have an older revision that repository.

Commit All

Commits all resources of working copy that have a newer version compared to that of the repository.

Refresh

Refreshes the whole content of the current working copy from disk starting from the root folder. At the end of the operation, the modified files and folders that were not committed to repository yet, are displayed in the **Working Copy** view.

Compare

The selected resource is compared with:

- The *BASE* revision, when the selected resource is:
  - Locally modified and the **All Files** view mode is currently selected (no matter if there are incoming changes).
  - Locally modified and there are no incoming changes when any other view mode is selected.
- The remote version of the same resource, when remote information is available after a **Synchronize** operation (only when one of **Modified**, **Incoming**, **Outgoing** and **Conflicts** view modes is selected).
- The working copy revision, when the selected resource is from the **History** view.

Show History

Displays the history of the selected resource (from the **Working Copy** or **Repository** views) in the **History** view.

Show Annotation

Displays the annotations of the selected resource. The selected resource can be in the **Working Copy** or the **History** views.

Revision Graph

Displays the revision graph of the selected resource. The selected resource can be in the **Working Copy** or the **Repositories** views.

Enable/Disable flexible layout

Toggles between a fixed and a flexible layout. When the flexible layout is enabled, you can move and dock the internal views to adapt the application to various viewing conditions and personal requirements.

Status Bar

The status bar of the Syncro SVN Client window displays important details of the current status of the application. This information is available only in the **Working Copy** view.

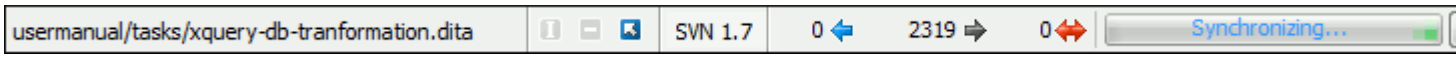


Figure 644: Status bar

The status bar is composed of the following areas:

- The path of the currently processed file from the current working copy (during an operation such as **Check out** or **Synchronize**) or the result of the last operation.
- The current status of the following working copy options:
  - Show ignored files ( ).
  - Show deleted files ( ).
  - Process `svn:externals` definitions ( ).

  The options for ignored and deleted files are switched on and off from *the **Settings** menu* of the **Working Copy** panel:
- The format of the currently loaded working copy.
- The current numbers of incoming changes ( ), outgoing changes ( ) and conflicting changes ( ).
- A progress bar for the currently running SVN operation and a button ( ) that allows you to stop it.

Getting Started

This section explains the basic operations that can be done in Syncro SVN Client.

SVN Repository Location

This section explains how to add and edit the repository locations in Syncro SVN Client.

Add / Edit / Remove Repository Locations

Usually, team members do all of their work separately, in their own working copy, and then must share their work by committing their changes. This is done using an Apache Subversion™ repository. Oxygen XML Editor supports versions 1.4, 1.5, 1.6, 1.7, and 1.8 of the SVN repository format.

Before you can begin working with a Subversion repository, you must define a repository location in the ***Repositories** view*.

To create a repository location, use the **New Repository Location** action that is available in the **Repository** menu, the **Repositories** view toolbar, and in the contextual menu. This action opens the **New Repository Location** dialog box, which prompts you for the *URL of the repository* you want to connect to. You can also *use peg revisions at the end of the URLs* (for example, URL`@rev1234`) to browse only that specific revision. No authentication information is requested at the time the location is defined. It is left to the Subversion client to request the user and password information when it is needed. The main benefit of allowing Subversion to manage your password is that it prompts you for a new password only when your password changes.

Once you enter the repository URL, Oxygen XML Editor tries to contact the server to get the content of the repository for displaying it in the ***Repositories** view*. If the server does not respond in the timeout interval set in the preferences, an error is displayed. If you do not want to wait until the timeout expires, you can use the **Stop** button from the toolbar of the view.

To edit a repository location, use the **Edit Repository Location** action that is available in the **Repository** menu and in the contextual menu. This action opens the **Edit Repository Location** dialog box, which prompts you for the *URL of the repository* you want to connect to. You can also *use peg revisions at the end of the URLs* (for example, URL`@rev1234`) to browse only that specific revision.

To remove a repository location, use the **Remove Repository Location** action that is available in the **Repository** menu and in the contextual menu. A confirmation dialog box is displayed to make sure that you do not accidentally remove the wrong locations.

The order of the repositories can be changed in the **Repositories** view at any time with the **Up** arrow and **Down** arrow buttons on the toolbar of the view. For example, pressing the up arrow once moves the selected repository in the list up one position.

To set the reference revision number of an SVN repository use the **Change the Revision to Browse** action that is available in the **Repository** menu and in the contextual menu. The revision number of the repository is used for displaying the contents of the repository when it is viewed in the ***Repositories** view*. Only the files and folders that were present in the repository at the moment when this revision number was generated in the repository are displayed as contents of the repository tree. Also, this revision number is used for all the operations executed directly from the ***Repositories** view*.

Authentication

Five protocols are supported: *HTTP*, *HTTPS*, *SVN*, *SVN + SSH* and *FILE*. If the repository that you are trying to access is password protected, the **Enter authentication data** dialog box requests a user name and a password. If the **Store authentication data** checkbox is selected, the credentials are stored in Apache Subversion™ default directory:

- Windows - `%HOME%\Application Data\Subversion\auth`. Example: `C:\Documents and Settings\John\Application Data\Subversion\auth`
- Linux and OS X - `$HOME/.subversion/auth`. Example: `/home/John/.subversion/auth`

There is one file for each server that you access. If you want to make Subversion forget your credentials, you can use the **Reset authentication** command from the **Options** menu. This causes Subversion to forget all your credentials. When you reset the authentication data, restart Oxygen XML Editor for the change to take effect.

Tip: The *FILE* protocol is recommended if the SVN repository and Oxygen XML Editor are located on the same computer as it ensures faster access to the SVN repository compared with other protocols.

For HTTPS connections where client authentication is required by your SSL server, you must choose the certificate file and enter the corresponding certificate password that is used to protect your certificate.

When using a secure HTTP (HTTPS) protocol for accessing a repository, a **Certificate Information** dialog box is displayed and asks you whether you want to accept the certificate permanently, temporarily, or simply deny it.

If the repository has SVN+SSH protocol, the SSH authentication can also be made with a private key and a pass phrase.

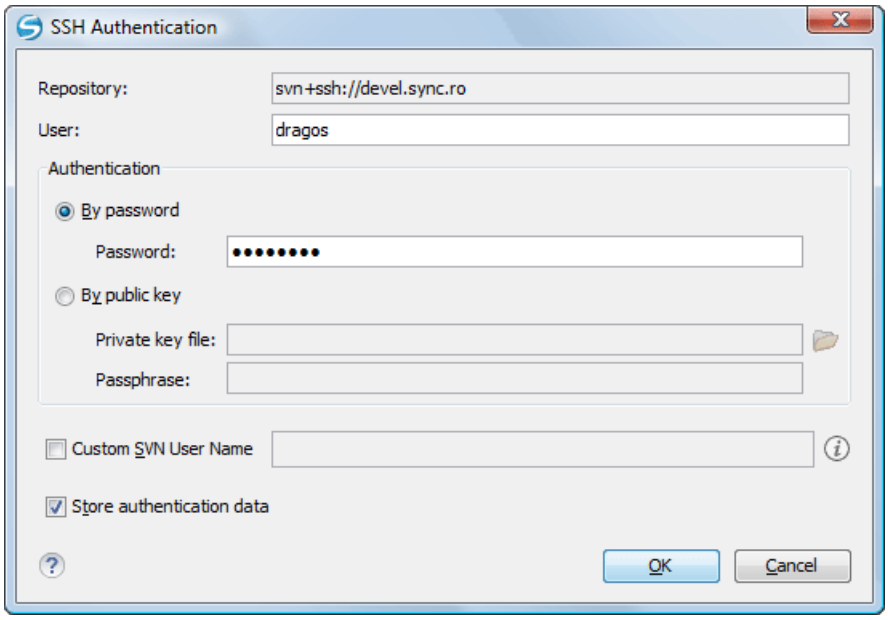


Figure 645: SSH Authentication Dialog Box

After the SSH authentication dialog box, another dialog box appears for entering the SVN user name that accesses the SVN repository. The SVN user name is recorded as the *committer* in SVN operations.

When connecting for the first time to a Subversion repository through SVN+SSH protocol, you will be asked to confirm if you trust the SSH host. The same dialog box is also displayed when the server changed the SSH key or when the key was deleted from the local Subversion cache folder.

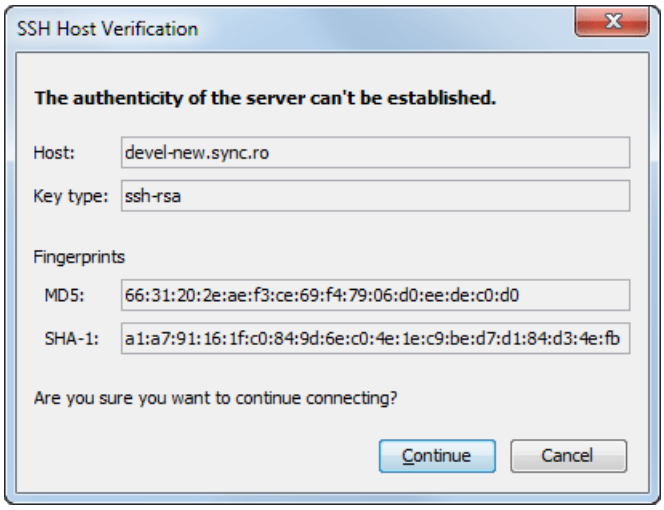


Figure 646: SSH server name and key fingerprint

Share a Project

Even if you start developing a new project, or you want to migrate an existing one to Subversion, the Syncro SVN Client allows you to easily share it with the rest of your team. The shared project directory is automatically converted to a working copy and added under Syncro SVN Client management. The **Share project** action is available in the **Tools** menu and the contextual menu of the **Repositories** view.

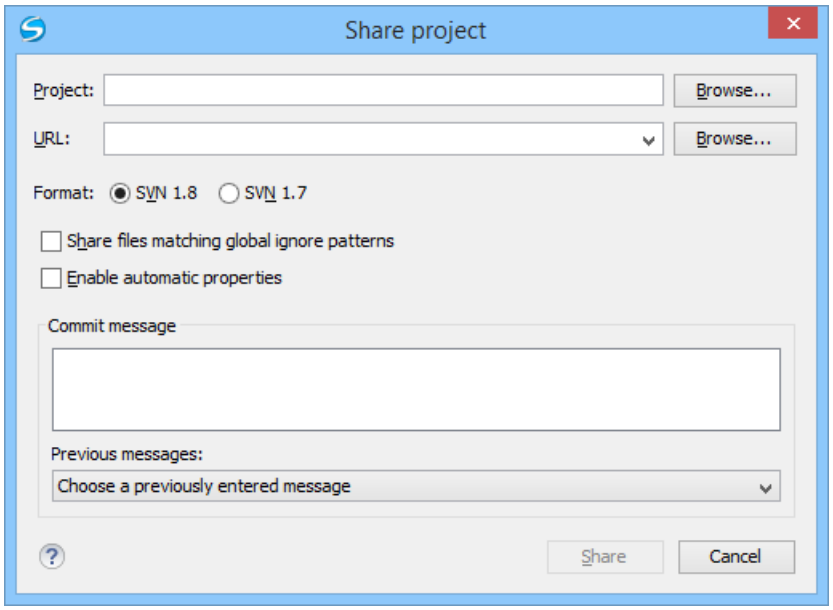


Figure 647: Share Project Dialog Box

The following options can be configured in the **Share project** dialog box:

Project

The location of the project folder on the local disk by using the text box or the **Browse** button. This folder should not be empty or already under version control.

Important: By default, the SVN system only imports the content of the specified folder, and not the root folder itself. Therefore, it is recommended to use the **Browse** button to select the project folder so that the client will automatically append the name of it to the specified URL.

URL

The new location of the project (inside the repository) that will be used to access it.

Note: *Peg revisions* have no effect for this operation since it is used to send information to the repository.

Attention: If the new location already exists, make sure that it is an empty directory to avoid mixing your project content with other files (if items exist with the same name, an error will occur and the operation will not proceed). Otherwise, if the address does not exist, it is created automatically.

Format

The SVN format of the working copy. You can choose between **SVN 1.8** or **SVN 1.7**.

Share files matching global ignore patterns

When selected, the file names that match the patterns defined in either of the following locations are also imported into the repository:

- The `global-ignores` property in *the SVN configuration file*.
- The ***File name ignore patterns*** *option* in the ***SVN*** *>* ***Working Copy*** *preferences page*.

Enable automatic properties/Disable automatic properties

Enables or disables automatic property assignment (per runtime configuration rules), overriding the `enable-auto-props` runtime configuration directive, defined in *the SVN configuration file*.

Note: This option is available only when there are defined properties to be applied automatically for newly added items under version control. You can define these properties in the SVN `config` file (in the `auto-props` section). Based on the value of the `enable-auto-props` runtime configuration directive, the presented option is either **Enable automatic properties**, or **Disable automatic properties**.

Defining a Working Copy

An Apache Subversion™ working copy is an ordinary directory tree on your local system, containing a collection of files. You can edit these files however you want, your working copy being your private work area. To make your

own changes available to others or incorporate changes made by others, you must explicitly tell Subversion to do so. You can even have multiple working copies of the same project.

svnkit | All Files | Modified | Incoming | Outgoing | Conflicts

Name	Date	Revision	Author	Size	Type
E:\svnkit	May 15, 2011	7636	alex		File Folder
gradle	May 4, 2011	7623	alex		File Folder
wrapper	May 4, 2011	7623	alex		File Folder
gradle-wrapper.jar	May 4, 2011	7618	alex	12 KB	Executable ...
gradle-wrapper.properties	May 4, 2011	7623	alex	1 KB	PROPERTIE...
svnkit	May 15, 2011	7636	alex		File Folder
svnkit-cli	May 10, 2011	7630	alex		File Folder
.settings	May 4, 2011	7618	alex		File Folder
src	May 10, 2011	7630	alex		File Folder
main	May 10, 2011	7630	alex		File Folder
conf	May 4, 2011	7618	alex		File Folder
java	May 4, 2011	7622	alex		File Folder
resources	May 4, 2011	7618	alex		File Folder
scripts	May 10, 2011	7630	alex		File Folder
jsvn	May 4, 2011	7618	alex	2 KB	File
jsvn.bat	May 10, 2011	7630	alex	2 KB	Windows B...
jsvnsetup.openvms	May 4, 2011	7618	alex	1 KB	OPENVMS File
build.gradle	May 4, 2011	7618	alex	2 KB	GRADLE File
svnkit-dav	May 4, 2011	7620	alex		File Folder
svnkit-distribution	May 4, 2011	7623	alex		File Folder
svnkit-javahl16	May 4, 2011	7618	alex		File Folder
svnkit-osgi	May 4, 2011	7623	alex		File Folder
svnkit-test	May 12, 2011	7635	alex		File Folder
.settings	May 4, 2011	7618	alex		File Folder
configurations	May 4, 2011	7618	alex		File Folder

Figure 648: Working Copy View

A Subversion working copy also contains some extra files, created and maintained by Subversion, to help it keep track of your files. In particular, each directory in your working copy contains a subdirectory named `.svn`, also known as the working copy *administrative directory*. This administrative directory contains an unaltered copy of the last updated files from the repository. This copy is usually referred to as the *pristine copy* or the *BASE revision* of the working copy. These files help Subversion recognize which files contain unpublished changes, and which files are out-of-date with respect to others' work.

A typical Subversion repository often holds the files (or source code) for several projects. Usually each project is a subdirectory in the repository's file system tree. In this arrangement, a user's working copy usually corresponds to a particular sub-tree of the repository.

Check Out a Working Copy

Check out means to make a copy of a project from a repository to your local file system. This copy is called a *working copy*. An Apache Subversion™ working copy is a specially formatted directory structure that contains additional `.svn` directories that store Subversion information, as well as a pristine copy of each item that is checked out.

To check out a working copy, locate and select the desired directory in the **Repositories** view and select the **Check out** action from the contextual menu, the toolbar, or the **Repository** menu.

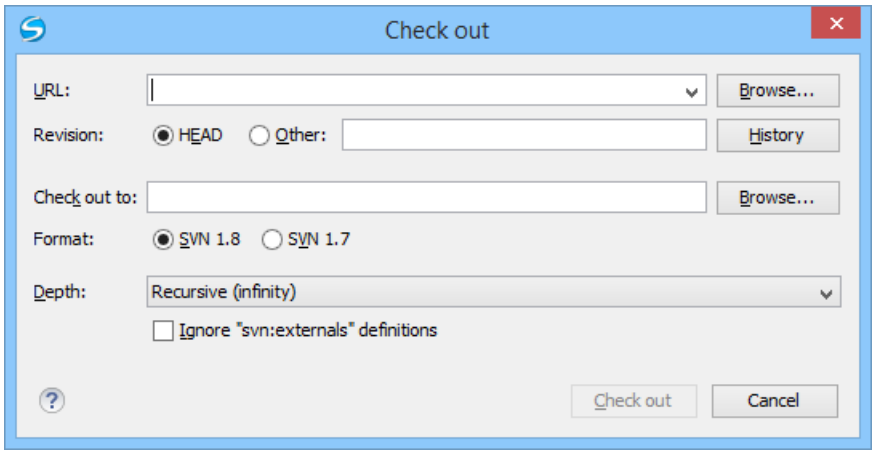


Figure 649: Check Out Dialog Box

The following options can be configured in the **Check out** dialog box:

URL

The location of the repository directory to be *checked out*.

Note: To check out an item that was deleted, moved, or replaced, you need to specify the original URL (before the item was removed) and use a *peg revision* at the end (for example, `URL@rev1234`).

Revision

You can choose between the **HEAD** or **Other** revision. If you need to *check out* a specific revision, specify it in the **Other** text box or use the **History** button and choose a revision from *the **History** dialog box*.

Check out to

Specify *the location where you want to check out* the new working copy by typing the local path in the text box or by using the **Browse** button. If the specified local path does not point to an existing directory, it will automatically be created.

Important: By default, the SVN system only checks out the content of the directory specified by the URL, and not the directory itself. Therefore, it is recommended to use the **Browse** button to select the *check out* location so that the client will automatically append the name of the remote directory to the path of the selected directory.

Warning: The destination directory should be empty. If files exist, they are skipped (left unchanged) by the *check out* operation and *displayed as modified* after the operation has finished. Also, the destination directory must not already be under version control.

Format

The SVN format of the working copy. You can choose between **SVN 1.8** or **SVN 1.7**.

Depth

The depth is useful if you want to *check out* only a part of the selected repository directory and bring the rest of the files and subdirectories in a future update. You can find out more about the checkout depth in the *sparse checkouts* section. You can choose between the following depths:

- **Recursive (infinity)** - Checks out all the files and folders contained in the selected folder.
- **Immediate children (immediates)** - Checks out only the child files and folders without recursing subfolders.
- **File children only (files)** - Checks out only the child files.
- **This folder only (empty)** - Checks out only the selected folder (no child file or folder is included).

Ignore "svn:externals" definitions

When selected, external items are ignored in the *check out* operation. This option is only available if you choose the **Recursive (infinity)** depth.

After a check out, the new working copy is added to the list in the ***Working Copy** view* and loaded automatically.

History Dialog Box

The **History** dialog box presents a list of revisions for a resource. It is opened from the dialog boxes that require setting an SVN revision number, such as *the **Check Out** dialog box* or *the **Branch / Tag** dialog box*. It presents information about revision, commit date, author, and commit comment.

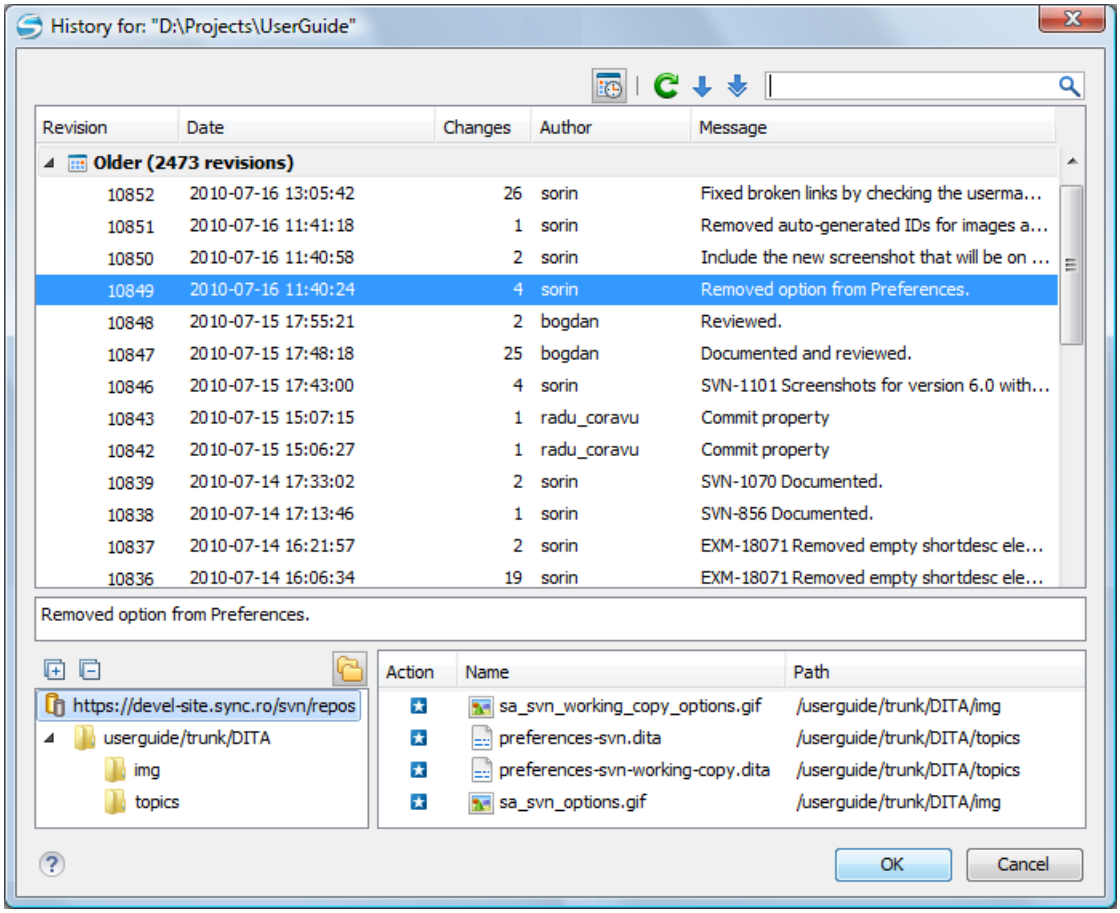


Figure 650: History Dialog Box

The initial number of entries in the list is 50. Additional revisions can be added to the list using the **Get next 50** and **Get all** buttons. The list of revisions can be refreshed at any time with the **Refresh** button. You can group revisions in predefined time frames (today, yesterday, this week, this month), by pressing the **Group by date** button from the toolbar.

The **Affected Paths** area displays all paths affected by the commit of the revision selected in history. You can see the changes between the selected revision and the file's previous state using the **Compare with previous version** action, available in the contextual menu.

Use an Existing Working Copy

Using an existing working copy is the process of taking a working copy that exists on your file system and connecting it to Apache Subversion™ repository. If you have a brand new project that you want to import into your repository, then see the section *Import resources into the repository*. The following procedure assumes that you have an existing valid working copy on your file system.

1. Click the **Working Copies Manager** toolbar button (on Mac OS X) in the ***Working Copy** view*.

   This action opens the **Working copies list** dialog box.
2. Press the **Add** button.
3. Select the working folder copy from the file system. The name is useful to differentiate between working copies located in folders with the same name. The default name is the name of the root folder of the working copy.

Note: For SVN 1.7 and newer working copies, all the internal information is kept only in the root directory. Thus, Syncro SVN Client needs to load the whole working copy.

4. Press the **OK** button.

The selected working copy is loaded and presented in the ***Working Copy*** *view*.

Notice: You can add working copies older than SVN 1.7. However, to load any of them, Syncro SVN Client will require to upgrade the working copy to SVN 1.8 format.

Manage Working Copy Resources

This section explains how to work with the resources that are displayed in the **Working Copy** view.

Edit Files

You can edit files from the ***Working Copy*** *view* by double clicking them or by right clicking them and choosing **Open** from the contextual menu.

Note that only one file can be edited at a time. If you try to open another file, it is opened in the same editor window. The editor has syntax highlighting for known file types, meaning that a different color is used for each type of recognized token in the file. If the selected file is an image, then it is previewed in the editor, with no access to modifying it.

After modifying and saving a file from a working copy, a modified marker - an asterisk (*) - will be added to the file's icon in the ***Working Copy*** *view*. The asterisk marks the files that have local modifications that were not committed to the repository.

Add Resources to Version Control

To share new files and folders (created in your working copy), add them to version control using the **Add to version control** option from the ***Working Copy*** *view*.

You can easily spot resources not under version control by the (*unversioned*) icon displayed in the **Local file status** column. Resources scheduled for addition (*added*) are displayed with this icon in the **Working Copy** view and are added in the repository after you commit them.

Note: Do not make a confusion between and icons. The former icon stands for resources that are actually copies of resources already committed in the repository, meaning they are *scheduled for addition with history*.

When you use the **Add to version control** option on a directory, its entire structure is scanned and all the resources that can be added under version control are presented.

Although it is not mandatory to add resources under version control explicitly, it is recommended. If you forgot to add a resource, when you *commit your changes*, the resource is presented in the commit dialog box, but not selected. When you commit and *unversioned* resource, it is automatically added under version control before starting the commit operation.

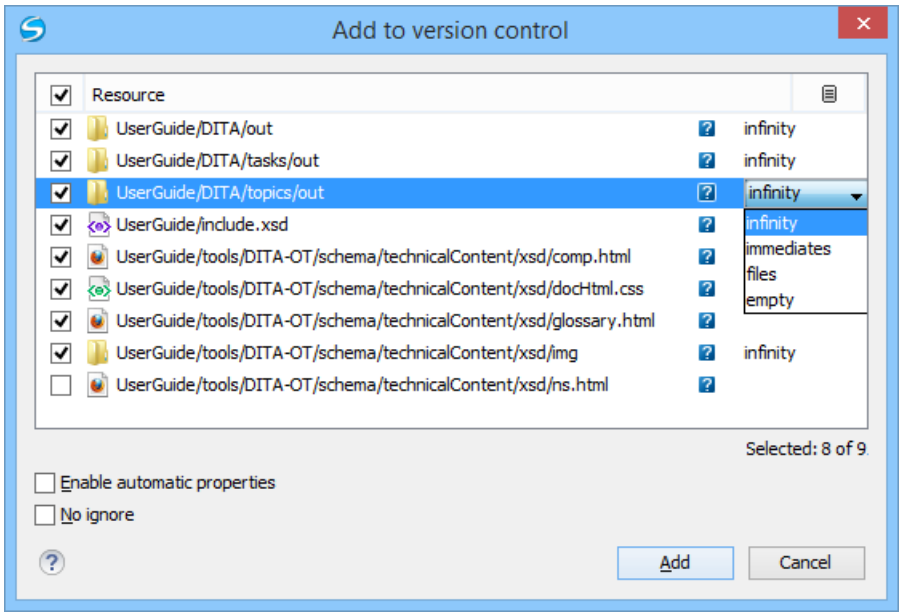


Figure 651: Add to Version Control Dialog Box

Note: *Ignored* ( ) items can also be added under version control.

The **Depth** column is displayed only when directories are also presented in the dialog box. For any directory, you can use one of the available values to instruct Subversion to limit the scope of the operation to a particular tree depth.

Note: The initial value of the **Depth** field can have the following values, depending on the *listing mode of the items in the working copy view*:

- *infinity* - When the working copy items are presented as a tree.
- *files* - When the working copy items are presented compressed.
- *empty* - When the working copy items are presented flat.

When you add unversioned or ignored directories, the initial value of the **Depth** field also depends on the state of the **Show unversioned directories content** and **Show ignored directories content** options. If these options are selected, the value is based on the listing mode of the items in the working copy view. When they are not selected, the value is *empty* .

The following options are available in this dialog box:

- **Enable automatic properties** or **Disable automatic properties** - enables or disables automatic property assignment (per runtime configuration rules), overriding the `enable-auto-props` runtime configuration directive, defined in the `config` file of the Subversion configuration directory.

  Note: This option is available only when there are defined properties to be applied automatically for resources newly added under version control. You can define these properties in the `config` file of the Subversion configuration directory, in the `auto-props` section. Based on the value of the `enable-auto-props` runtime configuration directive, the presented option is either **Enable automatic properties**, or **Disable automatic properties**.
- **No ignore** - when you select this option, file-name patterns defined to ignore *unversioned* resources do not apply. Resources that are located inside an *unversioned* directory selected for addition, and match these patterns, are also scheduled for addition in the repository.

  Note: This option is available only when directories are also presented in the dialog box.

  You can define file-name patterns to ignore *unversioned* resources in one of the following locations:

  - In the `config` file of the Subversion configuration directory (the `global-ignores` option from the `miscellany` section).
  - In the Oxygen XML Editor options (*open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **SVN** > **Working copy** > **Application global ignores**).

Each of the above two options is activated only when you select an item for which the option can be applied.

Ignore Resources Not Under Version Control

Some resources inside your working copy do not need to be subject to version control. These resources can be files created by the compiler, `*.obj`, `*.class`, `*.lst`, or output folders used to store temporary files. Whenever you *commit changes*, Apache Subversion™ shows your modified files in the commit dialog box, but the unversioned files are also listed. Since the unversioned files are committed unless otherwise specified, it is difficult to see exactly what you are committing.

The best way to avoid these problems is to add the derived files to the Subversion ignore list. That way they are never displayed in the commit dialog box and only genuine unversioned files that must be committed are displayed.

You can choose to ignore a resource by using the **Add to svn:ignore** action in the contextual menu of the ***Working Copy*** *view*.

In the **Add to svn:ignore** dialog box, you can specify the resource to be ignored by name or by a custom pattern. The custom pattern can contain the following wildcard characters:

- * - Matches any string of characters of any size, including the empty string.
- ? - Matches any single character.

For example, you can choose to ignore all text documents by using the pattern: `*.txt`.

The action **Add to svn:ignore** adds a predefined Subversion property called `svn:ignore` to the parent directory of the specified resource. In this property, there are specified all the child resources of that directory that must be ignored. The result is visible in the **Working Copy** view. The ignored resources are represented with gray icons.

Delete Resources

The ✕**Delete** action is available in the contextual menu of the ***Working Copy*** *view*. When you delete an item from the working copy, it is marked as *deleted* (scheduled for deletion from repository upon the next commit) and removed from the file system. Depending on the state of each item, you are prompted to confirm the operation.

If a resource is deleted from the file system without Subversion's knowledge, the resource is marked as *missing* ( ) in your working copy. You can decide what you want to do with a *missing* item:

- In the case of a commit, any *missing* item is first automatically deleted and then committed.

  Note: Not any *missing* item can be committed as *deleted*, and removed from the repository. For example, you cannot commit an item that no longer exists on the disk and that was scheduled for addition ( ) previously, since this item does not exist in the repository, but you can use the **Delete** action instead.
- If you want to recover *missing* items, either *update* the items themselves or one of their parent directories. This fetches their latest version from the repository.

You can also delete conflicting items (file content conflicts, property conflicts, tree-conflicts) and Syncro SVN Client automatically marks them as resolved.

Note: It is recommended that you resolve conflicts manually to avoid loosing any important remote modifications.

Finally, you can change your mind and *revert* the deleted items to their initial, pristine, state.

Copy Resources

You can copy resources from various locations of the working copy. You select them in the ***Working Copy*** *view* and then use **Copy to** from the contextual menu. This is not a simple file system copy, but an Apache Subversion™ command. It will copy the resource and the copy will also have the original history. This is one of the important features of Subversion, as you can keep track of where the copied resources originated.

Based on the selected items, the **Copy to** action is available only if it can be performed. Even if the operation would not normally be possible in SVN (due to some invalid local file states against copy), Oxygen XML Editor performs the copy operation as a simple file system operation. This means no SVN versioning meta-data is affected.

Note:

- If you copy an item to a directory that is *not under version control* (*unversioned* or *ignored*), the history of the item is not preserved. For example, when copying directories, all items inside them will also be copied without history.
- If you copy a directory that contains *external* items, these are not copied. This is specific for SVN 1.7 working copies only. To fetch the *external* items, use the **Update** operation on the copied directory.

In the **Copy to** dialog box, you can navigate through the working copy directories to choose a target directory, to copy inside it. If you try to copy a single resource you are also able to change that resource's name. For *versioned* items, you can select **Ignore resource history** to copy them without their history (similar to a simple file system copy).

Note: The **Copy to** dialog box only presents all the local directories that are a valid destination against the copy operation, based on their local file status. Also, the *working copy settings* are taken into account.

In the **Commit** dialog box, only the directory in question will appear without its children.

Move Resources

As in the case of the copy command, you can move several resources at once. Select the resources in the ***Working Copy*** *view* and choose the **Move to** action from the contextual menu. The move command actually behaves as if a copy followed by a delete command were issued. You will find the moved resources at the desired destination and also at their original location, but marked as *deleted*.

Note: *External* items cannot be moved using the **Move to** action, because they cannot be deleted. Instead, you should edit the `svn:externals` property defining the *external* item and use the **Update** operation on the item's parent folder for the change to take effect.

Attention: For SVN 1.8 working copies: when committing items that were moved and/or renamed, make sure you select both the source and the destination. Otherwise, the commit operation will fail.

Rename Resources

The **Rename** action is available in the contextual menu of the ***Working Copy*** *view* and can be performed on a single resource. This action acts as a move command with the destination directory being the same as the original location of the resource. A copy of the original item is created with the new name, also keeping its history. The original item is marked as *deleted*.

Note: *External* items cannot be renamed using the **Rename** action because they cannot be deleted. Instead, you should edit the *svn:externals* property defining the *external* item, then use the **Update** operation on the item's parent folder for the change to take effect.

Attention: For SVN 1.8 working copies: when committing items that were moved and/or renamed, make sure you select both the source and the destination. Otherwise, the commit operation will fail.

Lock / Unlock Resources

The idea of version control is based on the *copy-modify-merge* model of file sharing. This model states that each user contacts the repository and creates a local working copy (check out). Users can then work independently and modify their working copies according to their needs. When their goal has been accomplished, it is time for the users to share their work with the others, to send them to the repository (commit). When a user has modified a file that has been also modified on the repository, the two files will have to be merged. The version control system assists the user with the merging as much as it can, but in the end the user is the one that must make sure it is done correctly.

The copy-modify-merge model only works when files are contextually mergeable: this is usually the case of line-based text files (such as source code). However this is not always possible with binary formats, such as images or sounds. In these situations, the users must each have exclusive access to the file, ending up with a *lock-modify-unlock* model. Without this, one or more users could end up wasting time on changes that cannot be merged.

An SVN lock is a piece of metadata that grants exclusive access to a user. This user is called the lock owner. A lock is uniquely identified by a lock token (a string of characters). If someone else attempts to commit the file (or delete a parent of the file), the repository demands two pieces of information:

- User authentication - the user performing the commit must be the lock owner

- Software authorization - the user's working copy must have the same lock token as the one from the repository, proving that it is the same working copy where the lock originated from.

Scanning for Locks

When starting to work on a file that is not contextually mergeable (usually a binary file), it is better to verify if someone else is not already working on that file. You can do this in the ***Working Copy** view* by selecting one or more resources, then right-clicking them and choosing the **Scan for Locks** action from the contextual menu.

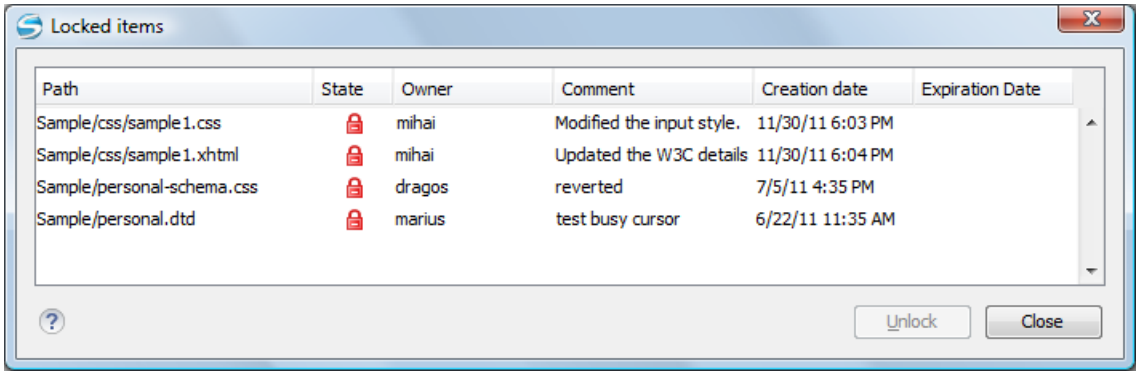


Figure 652: Locked Items Dialog Box

The **Locked items** dialog box contains a table with all the resources that were found locked on the repository. For each resource there are specified: resource path, state of the lock, owner of the lock, lock comment, creation and expiration date for the lock (if any).

The state of the lock can be one of the following:

- - Appears when one of the following conditions apply:
  - Another user has locked the file in the repository.
  - The file was locked by the same user from another working copy.
  - The file was locked from the **Repositories** view.
- - Displayed after you have locked a file from the current working copy.
- - A file already locked from your working copy is no longer locked in the repository (it was unlocked by another user).
- - A file already locked from your working copy is being locked by another user. Now the owner of the file lock is the user who stole the lock from you.

You can unlock a resource by selecting it and pressing the **Unlock** button.

Related Information:

Working Copy Locks on page 1537
Repository Locks on page 1530

Locking a File

By locking a file, you have exclusive write access to it in the repository.

You can lock a file from your working copy or directly from the **Repositories** view.

Note: You can only lock files (not directories). This is a restriction imposed by Apache Subversion™.

The **Lock** dialog box allows you to write a comment when you set a lock or when you *steal* an existing one. Note that you should *steal* a lock only after you made sure that the previous owner no longer needs it. Otherwise, you may cause an unsolvable conflict, which could be the reason the lock was put there in the first place. The Subversion server can have a policy concerning lock stealing, as it may not allow you to do this if certain conditions are not met.

The lock stays in place until you unlock the file or until someone breaks it. There is also the possibility that the lock expires after a period of time specified in the Subversion server policy.

Unlocking a File

A file can be unlocked from the contextual menu of the ***Working Copy*** *view*. A dialog box will prompt you to confirm the unlocking and it will also allow you to break the lock (unlock it by force).

Synchronize with Repository

In the work cycle you will need to incorporate other people's changes (update) and to make your own work available to others (commit). This is what the **Incoming** and **Outgoing** modes of *the* ***Working Copy*** *view* was designed for, to help you send and receive modifications from the repository.

The **Incoming** and **Outgoing** modes of this view focus on incoming and outgoing changes. The incoming changes are the changes that other users have committed in the repository since you last updated your working copy. The outgoing changes are the modifications you made to your working copy as a result of editing, removing or adding resources.

The view presents the status of the working copy resources against the BASE revision after a **Refresh** operation. You can view the state of the resources versus a repository HEAD revision by using the **Synchronize** action from *the* ***Working Copy*** *view*.

View Differences

One of the most common requirements in project development is to see what changes have been made to the files from your Working Copy or to the files from the repository. You can examine these changes after a synchronize operation with the repository, by using the **Open in compare editor** action from the contextual menu.

The text files are compared using a built-in ***Compare*** *view* that uses a line differencing algorithm or a specified external diff application (if such an application is set in the ***SVN Diff*** *preferences page*). When a file with outgoing status is involved, the compare is performed between the file from the working copy and the BASE revision of the file. When a file with incoming or conflict status is involved, the differences are computed using a three-way algorithm that means that the local file and the repository file are each compared with the BASE revision of the file. The results are displayed in the same view. The differences obtained from the local file comparison are considered outgoing changes and the ones obtained from the repository file comparison are considered incoming changes. If any of the incoming changes overlap outgoing changes then they are in conflict.

A special case of difference is a *diff pseudo-conflict*. This is the case when the left and the right sections are identical but the BASE revision does not contain the changes in that section. By default, this type of changes are ignored. If you want to change this, you can go to the **SVN** preferences page and select the ***Allow unversioned obstructions*** *option*.

The right editor of the internal compare view presents either the BASE revision or a revision from the repository of the file so its content cannot be modified. By default, when opening a synchronized file in the **Compare** view, a compare is automatically performed. After modifying and saving the content of the local file presented in the left editor, another compare is performed. You will also see the new refreshed status in the ***Working Copy*** *view*.

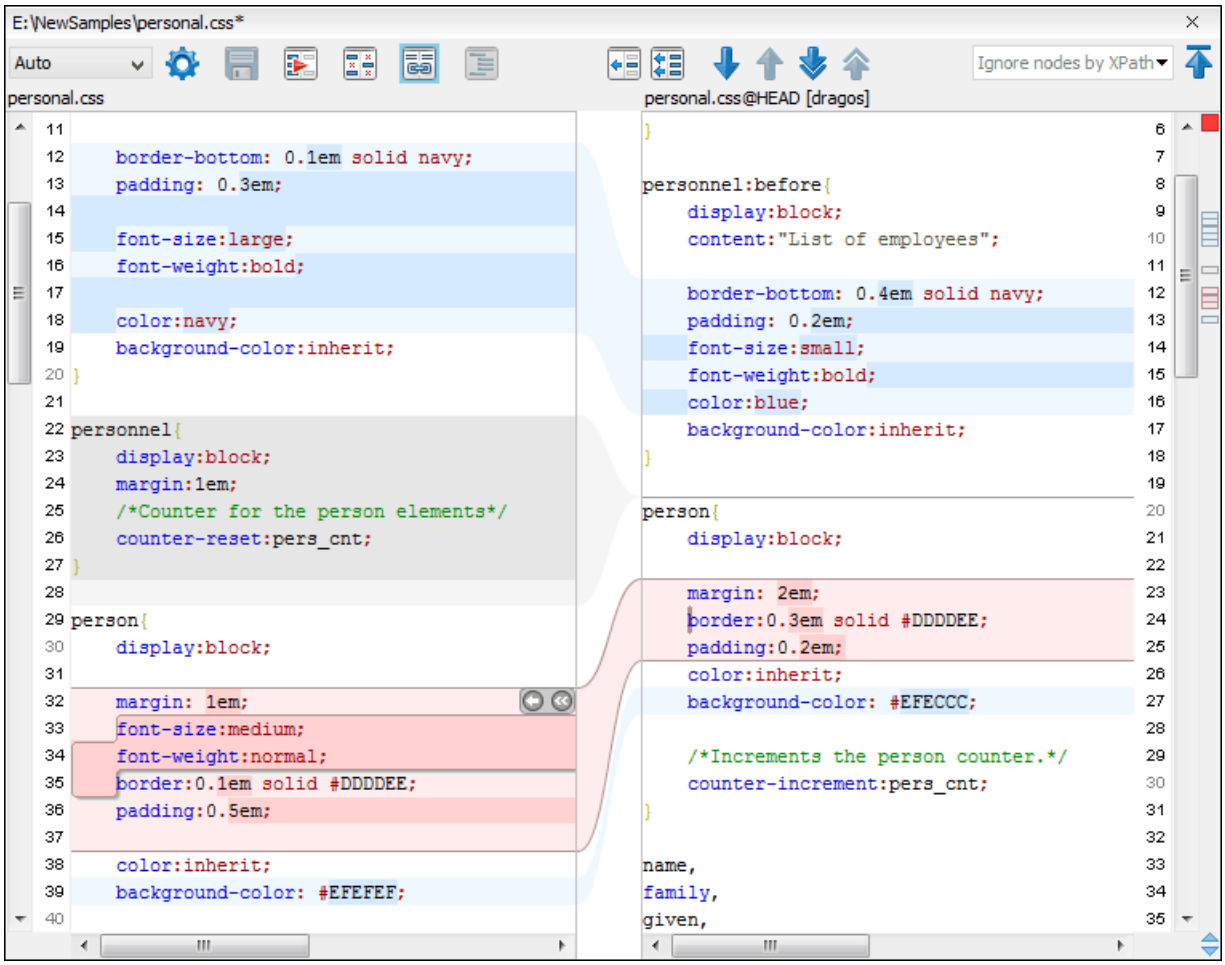


Figure 653: Compare View

At the top of each of the two editors, there are presented the name of the opened file, the corresponding SVN revision number (for remote resources) and the author who committed the associated revision.

There are three types of differences:

- Incoming changes - Changes committed by other users and not present yet in your working copy file. They are marked with a blue highlight and on the middle divider the arrows point from right to left.
- Outgoing changes - Changes you have done in the content of the working copy file. They are marked with a gray highlight and the arrows on the divider are pointing from left to right.
- Conflicting changes - This is the case when the same section of text that you already modified in the local file has been modified and committed by some other person. They are marked with a red highlight and red diamonds on the divider.

There are numerous actions and options available in the ***Compare View*** *toolbar* or in the **Compare** menu from the main menu. You can decide that some changes need adjusting or that new ones must be made. After you perform the adjustments, you may want to perform a new compare between the files. For this case there is an action called **Perform files differencing**. After each files differencing operation the first found change will be selected. You can navigate from one change to another by using the actions **Go to first**, **Go to previous**, **Go to next** and **Go to last modification**. If you decide that some incoming change needs to be present in your working file you can use the action **Copy change from right to left**. This is useful also when you want to override the outgoing modifications contained in a conflicting section. The **Copy all non-conflicting changes from right to left** action copies all incoming changes that are not contained inside a conflicting section in your local file.

Suppose that only a few words or letters are changed. Considering that the differences are performed taking whole lines of text into account, the change will contain all the lines involved. To find exactly what words or letters have changed, the **Word Details** and **Character Details** dialog boxes are available. They present a more detailed comparison result when you double-click the middle divider of a difference.

When you want to examine only the changes in the real text content of the files, while disregarding the changes in the number of white spaces between words or lines, there is an option available in the *SVN Preferences* that allows you to enable or disable the white space ignoring feature of the compare algorithm.

Conflicts

A file conflict occurs when two or more developers have changed the same few lines of a file or the properties of the same file. As Subversion knows nothing of your project, it leaves resolving the conflicts to the developers. Whenever a conflict is reported, you should open the file in question, and try to analyze and resolve the conflicting situation.

Real Conflicts vs Mergeable Conflicts

There are two types of conflicts:

- *real conflict* (icon in *Name* column) - Syncro SVN Client considers the following resource states to be real conflicts:
  - *conflicted* state - A file reported by SVN as being in this state is obtained after it was updated/merged while having incoming and outgoing content or property changes at the same time, changes that could not be merged. A content conflict (icon in *Local file status* column) is reported when the modified file has binary content or it is a text file and both local and remote changes were found on the same line. A properties conflict (icon in *Local properties status* column) is reported when a property's value was modified both locally and remotely.
  - *tree conflicted* state (icon in *Local file status* column) - Obtained after an update or merge operation, while having changes at the directory structure level (for example, file is locally modified and remotely deleted or locally scheduled for deletion and remotely modified).
  - *obstructed* state (icon in *Local file status* column) - Obtained after a resource was versioned as one kind of object (file, directory, symbolic link), but has been replaced outside Syncro SVN Client by a different kind of object.
- *pseudo-conflict* (icon in *Name* column) - A file is considered to be in *pseudo-conflict* when it contains both incoming and outgoing changes. When incoming and outgoing changes do not intersect, an update operation may automatically merge the incoming file content into the existing locally one. In this case, the *pseudo-conflict* marker is removed. This marker is used only as a warning that should prevent you to run into a real conflict.

Note:

- A conflicting resource cannot be committed to repository. You have to resolve it first, by using **Mark Resolved** action (after manually editing/merging file contents) or by using **Mark as Merged** action (for pseudo-conflicts).
- and icons are presented only when one of the following view modes is selected: **Modified**, **Incoming**, **Outgoing**, **Conflicts**.
- The icon is used also for folders to signal that they contain a file in real conflict or pseudo-conflict state.

Content Conflicts vs Property Conflicts

A *Content conflict* appears in the content of a file. A merge occurs for every inbound change to a file that is also modified in the working copy. In some cases, if the local change and the incoming change intersect each other, Apache Subversion™ cannot merge these changes without intervention. So if the conflict is real when updating the file in question the conflicting area is marked like this:

```
<<<<<<< filename
your changes
=======
code merged from repository
>>>>>>> revision
```

Also, for every conflicted file Subversion places three additional temporary files in your directory:

- `filename.ext.mine` - This is your file as it existed in your working copy before you updated your working copy, that is without conflict markers. This file has your latest changes in it and nothing else.
- `filename.ext.rOLDREV` - This is the file that was the BASE revision before you updated your working copy, that is the file revision that you updated before you made your latest edits.
- `filename.ext.rNEWREV` - This is the file that Subversion client just received from the server when you updated your working copy. This file corresponds to the HEAD revision of the repository.

OLDREV and NEWREV are revision numbers. If you have conflicts with binary files, Subversion does not attempt to merge the files by itself. The local file remains unchanged (exactly as you last changed it) and you will get `filename.ext.r*` files also.

A *Property conflict* is obtained when two people modify the same property of the same file or folder. When updating such a resource a file named `filename.ext.prej` is created in your working copy containing the nature of the conflict. Your local file property that is in conflict will not be changed. After resolving the conflict, you should use the **Mark resolved** action to commit the file. Note that the **Mark resolved** action does not really resolve the conflict. It just removes the conflicted flag of the file and deletes the temporary files.

Edit Real Content Conflicts

The conflicts of a file in the conflicted state (a file with the red double arrow icon) can be edited visually with the **Compare** view (the built-in file comparison tool) or with an *external diff application*. Resolving the conflict means deciding for each conflict if the local version of the change will remain or the remote one instead of the special conflict markers inserted in the file by the SVN server.

The **Compare** view (or the external diff application *set in Preferences*) is opened with the **Edit Conflict** action, which is available on the contextual menus of *the **Working Copy** view* for files in the conflicted state (an update operation was executed but the differences could not be merged without conflicts). The external diff application is called with 3 parameters because it is a 3-way diff operation between the local version of the file from the working copy and the HEAD version from the SVN repository with the BASE version from the working copy as common ancestor.

If the***Show warning dialog when edit conflicts** option* is selected, you will be warned at the beginning of the operation that the operation will overwrite the conflict version of the file received from the SVN server (the version that contains the conflict markers <<<<<<<, =======, >>>>>>>) with the original local version of the file that preceded the update operation. If you press the OK button the visual conflict editing will proceed and a backup file of the conflict version received from the SVN server is created in the same working copy folder as the file with the edited conflicts. The name of the backup file is obtained by appending the extension `.sync.bak` to the file as stored on the SVN server. If you press the **Cancel** button the visual editing will be aborted.

The usual actions on the differences between two versions of a file are available on the toolbar of this view:

Save

Saves the modifications of the local version of the file displayed in the left side of the view.

Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.

Ignore Whitespaces

Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces.

Synchronized scrolling

Toggles synchronized scrolling. When toggled on, a selected difference can be seen in both panels.

Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on OS X))

Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.

Note: When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.

Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.

Copy All Changes from Right to Left

Copies all changes from the file in the right panel to the file in the left panel.

Next Block of Changes (Ctrl + Period (Command + Period on OS X))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

Note: A change block groups one or more consecutive lines that contain at least one change.

Previous Block of Changes (Ctrl + Comma (Command + Comma on OS X))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

Next Change (Ctrl + Shift + Period (Command + Shift + Period on OS X))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

Previous Change (Ctrl + Shift + Comma (Command + Shift + M on OS X))

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

First Change (Ctrl + B (Command + B on OS X))

Jumps to the first change.

The operation begins by overwriting the conflict version of the file received from the SVN server (the version that contains the conflict markers <<<<<<<, =======, >>>>>>>) with the original local version of the file before running the update action that created the conflict. After that the differences between this original local version and the repository version are displayed in the **Compare** view.

If you want to edit the conflict version of the file directly in a text editor instead of the visual editing offered by the **Compare** view you should work on the local working copy file after the update operation without running the action **Edit Conflict**. If you decide that you want to edit the conflict version directly after running the action **Edit Conflict** you have to work on the `.sync.bak` file.

If you did not finish editing the conflicts in a file at the first run of the action **Edit Conflict** you can run the action again and you will be prompted to choose between resuming the editing where the previous run left it and starting again from the conflict file received from the SVN server.

After the conflicts are edited and saved in the local version of the file you should run one of the following:

- The **Mark Resolved** action on the file so that the result of the conflict editing process can be committed to the SVN repository.
- The **Revert** action so that the repository version overwrites all the local modifications.

Both actions remove the backup file and other temporary files created with the conflict version of the local file.

Revert Your Changes

If you want to undo changes made in your working copy, since the last update, select the items you are interested in, right-click to display the contextual menu and select **Revert**. A dialog box will open that shows you the files and folders that you have changed and can be reverted. Select those you want to revert and click the **OK** button. Revert will undo only your local changes. It does not undo any changes that have already been committed. If you choose to revert a conflicting item to its pristine copy, then the eventual conflict is solved by losing your outgoing modifications. If you try to revert a resource not under version control, the resource will be deleted from the file system.

Note: By default, a directory will be recursively reverted (including any other modified item it contains). However, if the directory has only property changes, you need to explicitly choose if the operation will include any modified items found inside it.

If you want some of your outgoing changes to be overridden you must first open the file in ***Compare*** *view* and choose the sections to be replaced with ones from the repository file. This can be achieved either by editing directly the file or by using the action **Copy change from right to left** from the ***Compare*** *view toolbar*. After editing the conflicting file you have to run the action **Mark as merged** before committing it.

If you want to drop all local changes and bring all incoming changes into your working copy resource, you can use the **Override and update** action. It discards the changes in the local file and updates it from the repository. A dialog box will display the files that will be affected.

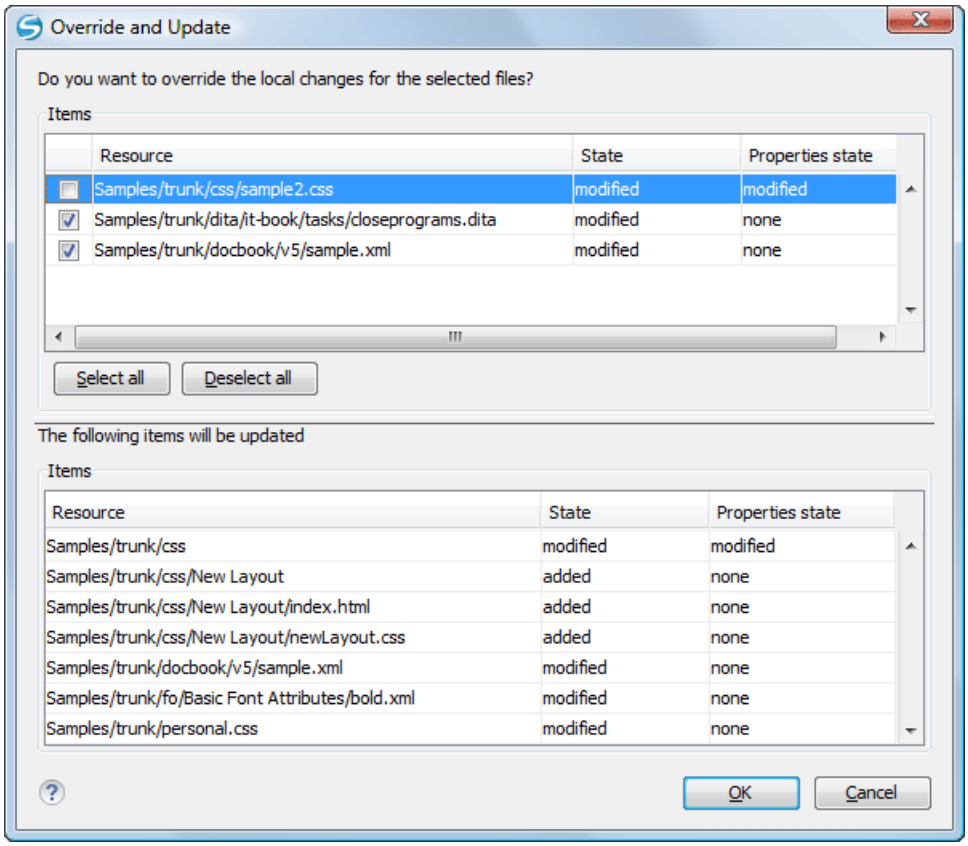


Figure 654: Override and Update Dialog Box

In the first table of the dialog box you will be able to see the resources that will be overridden. In the second table you will find the list of resources that will be updated. Only resources that have an incoming status are updated.

Tip: If you want to roll-back out of your working copy changes that have already been committed to the repository, see *Merge Revisions*.

Merge Conflicted Resources

Before you can safely commit your changes to the repository you must first resolve all conflicts. In the case of pseudo-conflicts they can be resolved in most cases with an update operation that will merge the incoming modifications into your working copy resource. In the case of real conflicts, conflicts that persist after an update operation, it is necessary to resolve the conflict using the built-in compare view and editor or, in the case of properties conflict, the ***Properties*** *view*. Before you can commit you must *mark as resolved* the affected files.

Both pseudo and real conflicts can be resolved without an update. You should open the file in the compare editor and decide which incoming changes need to be copied locally and which outgoing changes must be overridden or modified. After saving your local file you have to use the *Mark as merged* action from the contextual menu before committing.

Drop Incoming Modifications

In the situation when your file is in conflict but you decide that your working copy file and its content is the correct one, you can decide to drop some or all of the incoming changes and commit afterwards. The action **Mark as merged** proves to be useful in this case too. After opening the conflicting files with ***Compare*** *view*, *Editor* or editing their properties in the **Properties** view and deciding that your file can be committed in the repository replacing the existing one, you should use the **Mark as merged** action. When you want to override completely the remote file with the local file you should run the **Override and commit** action, which drops any remote changes and commits your file.

In general it is much safer to analyze all incoming and outgoing changes using the **Compare** view and only after to update and commit.

Tree Conflicts

A *tree conflict* is a conflict at the directory tree structure level and occurs when the user runs an update action on a resource that has the following conditions:

- It is locally modified and the same resource was deleted from the repository (or deleted as a result of being renamed or moved).
- It was locally deleted (or deleted as a result of being renamed or moved) and the same resource is incoming as modified from the repository.

The same conflict situation can occur after a merge or a switch action. The action ends with an error and the folder containing the file that is now in the tree conflict state is also marked with a conflict icon.

Such a conflict can be resolved in one of the following ways that are available when the user double clicks on the conflicting resource or when running the **Edit conflict** action:

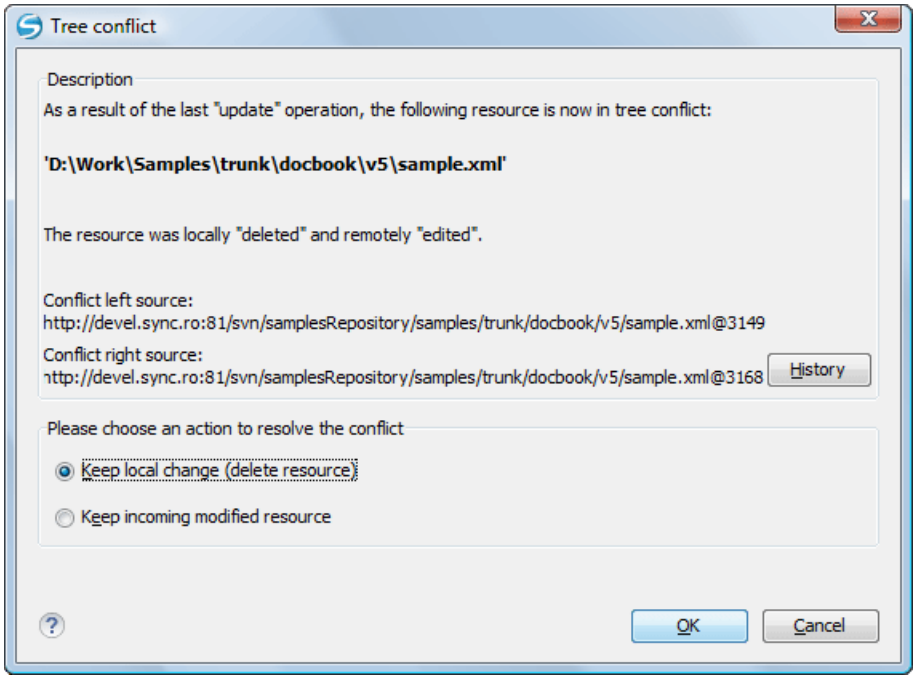


Figure 655: Resolve a tree conflict

- **Keep local change (delete resource)** - Keeps the incoming change that comes from the repository.
- **Keep incoming modified resource** - If there is a renamed version of the file committed by other user that will be added to the working copy too.

Update the Working Copy

While you are working on a project, other members of your team may be committing changes to the project repository. To get these changes, you have to *update* your working copy. Updating may be done on single files, a set of selected files, or recursively on entire directory hierarchies. The update operation can be performed from ***Working Copy*** *view*. It updates the selected resources to the last synchronized revision (if remote information is available) or to the *HEAD* revision of the repository.

There are three different kinds of incoming changes:

- *Non-conflicting* - A non-conflicting change occurs when a file has been changed remotely but has not been modified locally.
- *Conflicting, but auto-mergeable* - An auto-mergeable conflicting change occurs when a text file has been changed both remotely and locally (for example, has non-committed local changes) but the changes are on different lines of text. Not applicable to binary resources (for example, multimedia files, PDFs, executable program files)
- *Conflicting* - A conflicting change occurs when one or more of the same lines of a text file have been changed both remotely and locally.

If the resource contains only incoming changes or the outgoing changes do not intersect with incoming ones then the update will end normally and the Subversion system will merge incoming changes into the local file. In the case of a conflicting situation the update will have as result a file with conflict status.

The Oxygen XML Editor allows you to update your working copy files to a specific revision, not only the most recent one. This can be done by using the **Update to revision/depth** action from the **Working Copy** view (**All Files** view mode) or the **Update to revision** action from the ***History*** *view* contextual menu.

If you select multiple files and folders and then you perform an **Update** operation, all of those files and folders are updated one by one. The Subversion client makes sure that all files and folders belonging to the same repository are updated to the exact same revision, even if between those updates another commit occurred.

When the update fails with a message saying that there is already a local file with the same name Subversion tried to check out a newly versioned file, and found that an unversioned file with the same name already exists in your working folder. Subversion will never overwrite an unversioned file unless you specifically do this with an **Override and update** action. If you get this error message, the solution is simply to rename the local unversioned file. After completing the update, you can check to see if the renamed file is still needed.

Send Your Changes to the Repository

Sending the changes you made to your working copy is known as *committing* the changes. If your working copy is up-to-date and there are no conflicts, you are ready to commit your changes.

The **Commit** action sends the changes from your local working copy to the repository. The **Commit** dialog box presents all the items that you can commit.

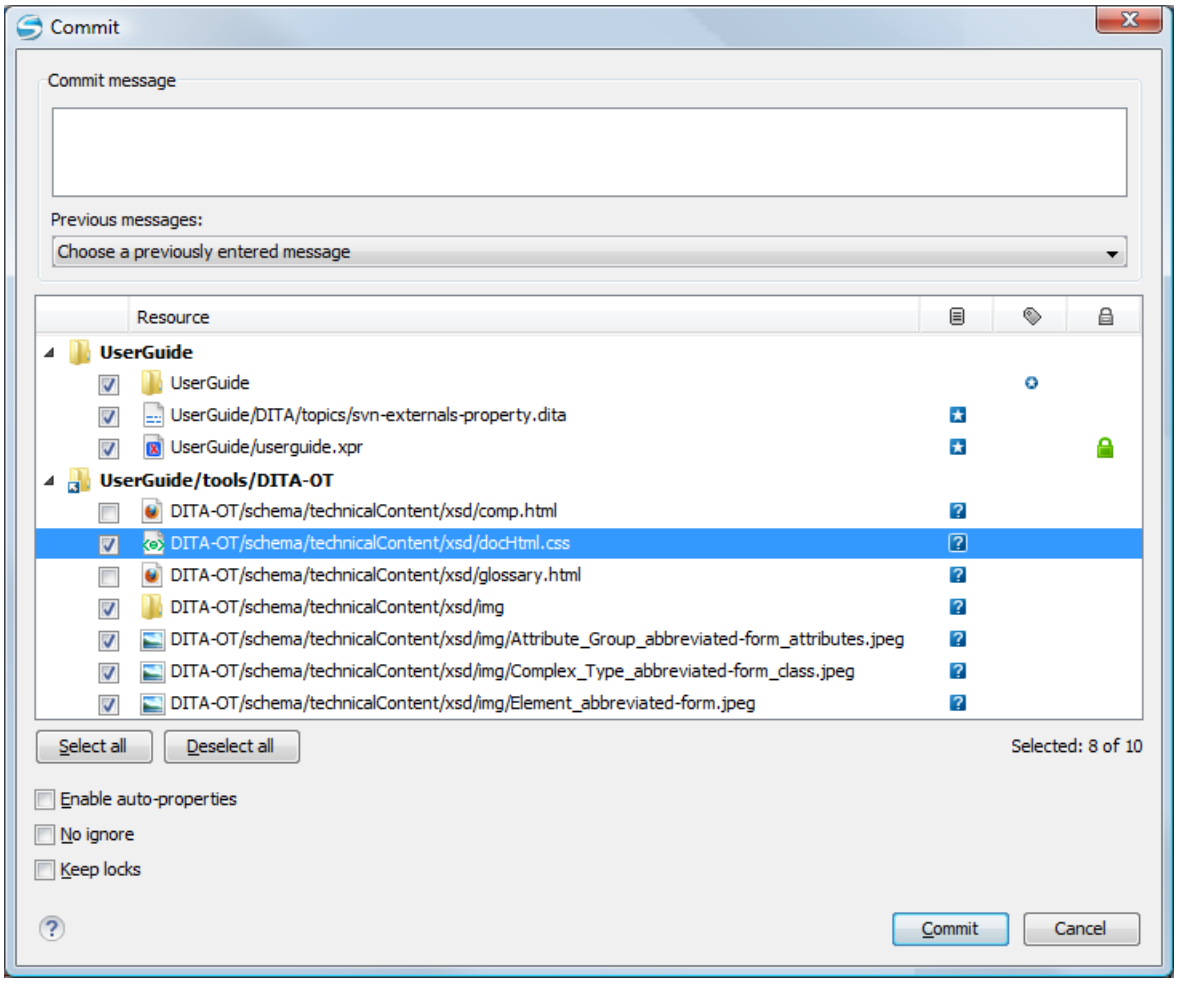


Figure 656: Commit dialog box

Enter a message to associate with the commit, or choose a previous message from the **Previous messages** list (the last 10 commit messages will be remembered even after restarting the SVN client application).

An item that can be committed has one of the following states: *added*, *modified* (content or properties), *replaced*, and *deleted*. All items that have one of these states are selected in the dialog box by default. If you do not want to commit one of the items, deselect it.

Attention: For SVN 1.8 working copies: when committing items that were moved and/or renamed, make sure you select both the source and the destination. Otherwise, the commit operation will fail.

Besides the items that have one of the mentioned states, Syncro SVN Client also includes the files being *unversioned* or *missing* and these items are handled automatically:

- *Unversioned* items are added under version control.
- *Missing* items are deleted.

Note: If the **Show unversioned directories content** option is not selected, the **Commit** dialog box does not display the items inside an *unversioned*directory.

Unversioned or *missing* items are not selected by default in the **Commit** dialog box, unless you have selected them explicitly when issuing the commit command.

Note: In some cases, items that have one of the above states are not presented in the **Commit** dialog box.

For example:

- Items that have been *added* or *replaced* previously, but now are presented as *missing* after being removed from the file system, outside of an SVN client. Such items do not exist in the repository and you should use the **Delete** action to remove them from your working copy.

- Items that have incoming changes from the repository, after a synchronization. You need to have your working copy up-to-date before committing your changes.
- Files that, after a synchronization, appear as locked by other users or from other locations than the current working copy.

Note: Due to dependencies between items, when you select or clear an *unversioned* ( ) or *added* ( ) item in the **Commit** dialog box, other items with one of these states can be selected or cleared automatically.

The modifications that will be committed for each file can be reviewed in the compare editor window by double clicking a file in the **Commit** dialog box, or by right clicking and selecting the **Show Modifications** action from the contextual menu. This option is available to review only file content changes, not property changes.

The **Local file status** column indicates the actual state of the items and the **Local properties status** column indicates whether or not the properties of an item are modified.

The **Lock information** column is displayed if at least one of the files in the **Commit** dialog box has lock information associated with it, valid against the commit operation.

The following options are available in this dialog box:

- **Enable automatic properties** or **Disable automatic properties** - enables or disables automatic property assignment (per runtime configuration rules), overriding the `enable-auto-props` runtime configuration directive, defined in the `config` file of the Subversion configuration directory.

  Note: This option is available only when there are defined properties to be applied automatically for resources newly added under version control. You can define these properties in the `config` file of the Subversion configuration directory, in the `auto-props` section. Based on the value of the `enable-auto-props` runtime configuration directive, the presented option is either **Enable automatic properties**, or **Disable automatic properties**.
- **Keep locks** - selecting the **Keep locks** option preserves any locks you set on various files.

  Note: This option is available only when files that you locked are presented in the dialog box.

Each of the above options is activated only when you select an item for which the option can be applied.

Your working copy must be up-to-date with respect to the resources you commit. This is ensured by using the **Update** action prior to committing, resolving conflicts and re-testing as needed. If your working copy resources you are trying to commit are out of date you will get an appropriate error message.

Committing to Multiple Locations

Although Subversion does not support committing to multiple locations at once, Syncro SVN Client offers this functionality regarding *external* items.

If items to be committed belong to different *external* definitions than those found in the working copy, they are grouped under the corresponding item that indicates their repository origin. Each parent item is rendered bold and its corresponding repository location is presented when hovering it. Parent items are decorated with a small arrow ( ) if they are *external* definitions. The working copy root directory is never decorated and is not presented if there are no *external* items listed (all items belong to the main working copy). Each child item is presented relative to the parent item.

Note: When an *external* directory has modifications of its own, it is presented both as a parent item and as an item that you can select and commit. This is always the case for *external* files.

The sets of items belonging to *external* definitions from the same repository are committed together, resulting a single revision. So, the number of revisions can be smaller than the number of *externals*. External definitions are considered from the same repository if they have the same protocol, server address, port, and repository address within the server.

Note: *External* files are always from the same repository as the parent directory that defines them, so they are always committed together with the changes from their parent directory.

Integration with Bug Tracking Tools

Users of bug tracking systems can associate the changes they make in the repository resources with a specific ID in their bug tracking system. The only requirement is that the user includes the bug ID in the commit message

that they enter in the **Commit** dialog box. The format and the location of the ID in the commit message are configured with SVN properties.

To make the integration possible Syncro SVN Client needs some data about the bug tracking tool used in the project. You can configure this using the following *SVN properties* that must be set on the folder that contains resources associated with the bug tracking system (usually they are set recursively on the root folder of the working copy):

- **bugtraq:message** - A string property. If it is set the ***Commit** dialog box* will display a text field for entering the bug ID. It must contain the string *%BUGID%*, which is replaced with the bug number on commit.
- **bugtraq:label** - A string property that sets the label for the text field configured with the **bugtraq:message** property.
- **bugtraq:url** - A string property that is the URL pointing to the bug tracking tool. The URL string should contain the substring *%BUGID%* which Syncro SVN Client replaces with the issue number. That way the resulting URL will point directly to the correct issue.
- **bugtraq:warnifnoissue** - A boolean property with the values *true*/*yes* or *false*/*no*. If set to *true*, the Syncro SVN Client will warn you if the bug ID text field is left empty. The warning will not block the commit, only give you a chance to enter an issue number.
- **bugtraq:number** - A boolean property with the value *true* or *false*. If this property is set to *false*, then any character can be entered in the bug ID text field. If the property is set to *true* or is missing then only numbers are allowed as the bug ID.
- **bugtraq:append** - A boolean property. If set to *false*, then the bug ID is inserted at the beginning of the commit message. If *yes* or not set, then it's appended to the commit message.
- **bugtraq:logregex** - This property contains one or two regular expressions, separated by a newline. If only one expression is set, then the bug ID's must be matched in the groups of the regular expression string (for example, `[Ii]ssue #?(\d+)`). If two expressions are set, then the first expression is used to find a string which relates to a bug ID but may contain more than just the bug ID (for example, `Issue #123` or `resolves issue 123`). The second expression is then used to extract the bug ID from the string extracted with the first expression. An example: if you want to catch every pattern `issue #XXX` and `issue #890, #789` inside a log message you could use the following strings:
  - `[Ii]ssue #?(\d+)(,? ?#?(\d+))+`
  - `(\d+)`

The data configured with these SVN properties is stored on the repository when a revision is committed. A bug tracking system or a statistics tools can retrieve the revisions that affected a bug from the SVN server and present the commits related to that bug to the user of the bug tracking system.

If the **bugtraq:url** property was filled in with the URL of the bug tracking system and this URL includes the *%BUGID%* substring as specified above in the description of the **bugtraq:url** property then the ***History** view* presents the bug ID as a hyperlink in the commit message. Clicking such a hyperlink in the commit message of a revision opens a Web browser at the page corresponding to the bug affected by that commit.

Obtain Information for a Resource

This section explains how to obtain information for a SVN resource:

Request Status Information for a Resource

While you are working with the SVN Client you often need to know which files you have changed, added, removed, or renamed, or even which files got changed and committed by others. This is where the **Synchronize** action from the ***Working Copy** view* comes in handy. The **Working Copy** view shows you every file that has changed your working copy, as well as any unversioned files you may have.

If you want more detailed information about a given resource, you can use the **Show SVN Information** action. This action is available from the **File** menu or the contextual menu of the **Working Copy**, **Repositories**, **History**, or **Directory Change Set** views, or from the **Revision Graph** dialog box. The **SVN Information** dialog box will be displayed, showing information about the selected resource. The information displayed depends on the location of the item (local or remote) and may include the following:

- Local path and repository location
- Revision number

- Last change author, revision and date
- Information about locks
- Local file status
- Local properties status
- Local directory depth
- Repository location and revision number for copied files or directories
- Path information about locally moved items
- Path information about conflict generated files
- Remote file status
- Remote properties status
- File size and other information

The value of a property of the resource displayed in the dialog box can be copied by right-clicking the property and selecting the **Copy** action.

Request History for a Resource

In Apache Subversion™, both files and directories are versioned and have a history. If you want to examine the history for a selected resource and find out what happened at a certain revision you can use the **History view** that can be accessed from *Repositories view*, *Working Copy view*, *Revision Graph*, or *Directory Change Set view*. From the **Working copy view** you can display the history of local versioned resources. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Related Information:

History View on page 1543

Management of SVN Properties

In the ***Properties** view* you can read and set the Apache Subversion™ properties of a file or folder. There is a set of predefined properties with special meaning to Subversion. For more information about properties in Subversion see the SVN Subversion specification. Subversion properties are revision dependent. After you change, add or delete a property for a resource, you have to commit your changes to the repository.

If you want to change the properties of a given resource you need to select that resource from the ***Working Copy** view* and run the **Show properties** action from the contextual menu. The ***Properties** view* will show the local properties for the resource in the working copy. Once the Properties view is visible, it will always present the properties of the currently selected resource. There are actions available in the **Properties** view *toolbar* that allow you to add, change, and delete the properties.

If you choose the **Add a new property** action, a new dialog box will appear that contains the following:

- **Name** - Combo box that allows you to enter the name of the property. The drop-down menu of the combo box presents the predefined Subversion properties (such as **svn:ignore**, **svn:externals**, **svn:needs-lock**, etc.)
- **Current value** - Text area that allows you to enter the value of the new property.

If the selected item is a directory, you can also set the property recursively on its children by selecting the **Set property recursively** checkbox.

If you want to change the value for a previously set property, you can use the **Edit property** action, which will display a dialog box with the following information:

- **Name** - Property name (cannot be changed).
- **Current value** - The current value (can be changed).
- **Base value** - The value of the property, if any, from the resource in the pristine copy (cannot be changed).

If you want to completely remove a property previously set you can choose the **Remove property** action. It will display a confirmation dialog box in which you can also choose if the property will be removed recursively.

There is a **Refresh** action in the ***Properties** view* that can be used when the properties have been changed from outside the view. This can happen, for example, when the view was already presenting the properties of a resource and they have been changed after an **Update** operation.

Branches and Tags

One of the fundamental features of version control systems is the ability to create a new line of development from the main one. This new line of development will always share a common history with the main line if you look far enough back in time. This line is known as a *branch*. Branches are mostly used to try out features or fixes. When the feature or fix is finished, the branch can be merged back into the main branch (*trunk*).

Another feature of version control systems is the ability to take a snapshot of a particular revision, so you can at any time recreate a certain build or environment. This is known as *tagging*. Tagging is especially useful when making release versions.

In Apache Subversion™, there is no difference between a *tag* and a *branch*. On the repository, both are ordinary directories that are created by copying. The trick is that they are cheap copies instead of physical copies. Cheap copies are similar to hard links in Unix, which means that they merely link to a specific tree and revision without making a physical copy. As a result, branches and tags occupy little space on the repository and are created very quickly.

Provided that nobody ever commits to the directory in question, it remains a tag. If people start committing to it, it becomes a branch.

Create a Branch / Tag

To create a branch or tag by copying a directory, use the **Branch/Tag** action that is available in the **Tools** menu when an item is selected in the ***Working Copy*** *view* or ***Repositories*** *view*, or from the contextual menu of the **Repositories** view.

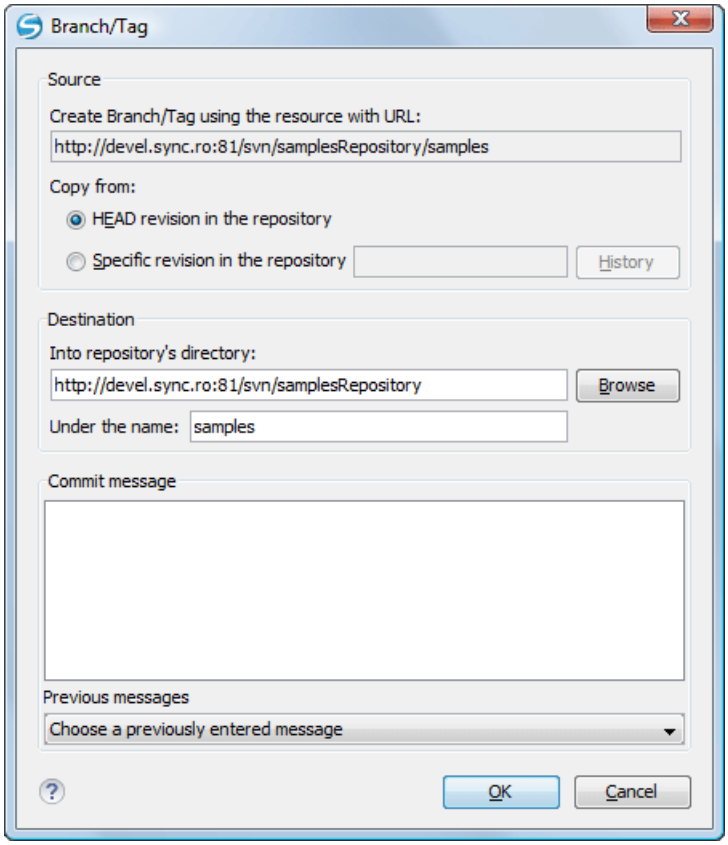


Figure 657: Branch/Tag Dialog Box

You can configure the following options in this dialog box:

You can specify the source revision of the copy in the **Copy from** section. You can choose between the following options:

- **HEAD revision in the repository** - The new branch or tag will be copied in the repository from the HEAD revision. The branch will be created very quickly, as the repository will make a *cheap* copy.

- **Specific revision in the repository** - The new branch will be copied into the repository, but you can specify the exact desired revision. For example, this is useful if you forgot to make a branch or tag when you released your application. If you click the **History** button you can select the revision number from *the **History** dialog box*. This type of branch will also be created very quickly.
- **Working copy** - (Available only if the item is selected from the **Working copy** view). The new branch will be a copy of your local working copy. If you have updated some files to an older revision in your working copy, or if you have made local changes, that is exactly what goes into the copy. This involves transferring some data from your working copy back to the repository, or more specifically, the locally modified files.

You can specify the location of the new branch or tag in the **Destination** section:

- **Into repository's directory** - *The URL of the parent directory* of the new branch or tag.

  Note: *Peg revisions* have no effect for this operation since it is used to send information to the repository.
- **Under the name** - You can specify another branch or tag name other than the name of the resource selected in the **Repositories** or **Working copy** view.

The new branch or tag will be created as a child of the specified URL of the repository directory and will have the new name.

Merging

At some stage during the development process, you will want to merge the changes made on a *branch* back into the *trunk*, or vice-versa. The *merge* is accomplished by comparing two points (branches or revisions) in the repository and applying the obtained differences to your working copy. This process is closely related to the *diff* concept.

Note: A *branch* is a line of development that exists independently of another line, yet still shares a common history if you look far enough back in time. A *branch* always begins life as *a copy of something* (such as a trunk, another branch, or tag), and moves on from there, generating its own history.

The **Merge** action is available in the **Tools** menu. The working copy item selected when you issued the command will be the one receiving the generated changes. If there is no item selected, the *merge* operation will be performed on the entire working copy.

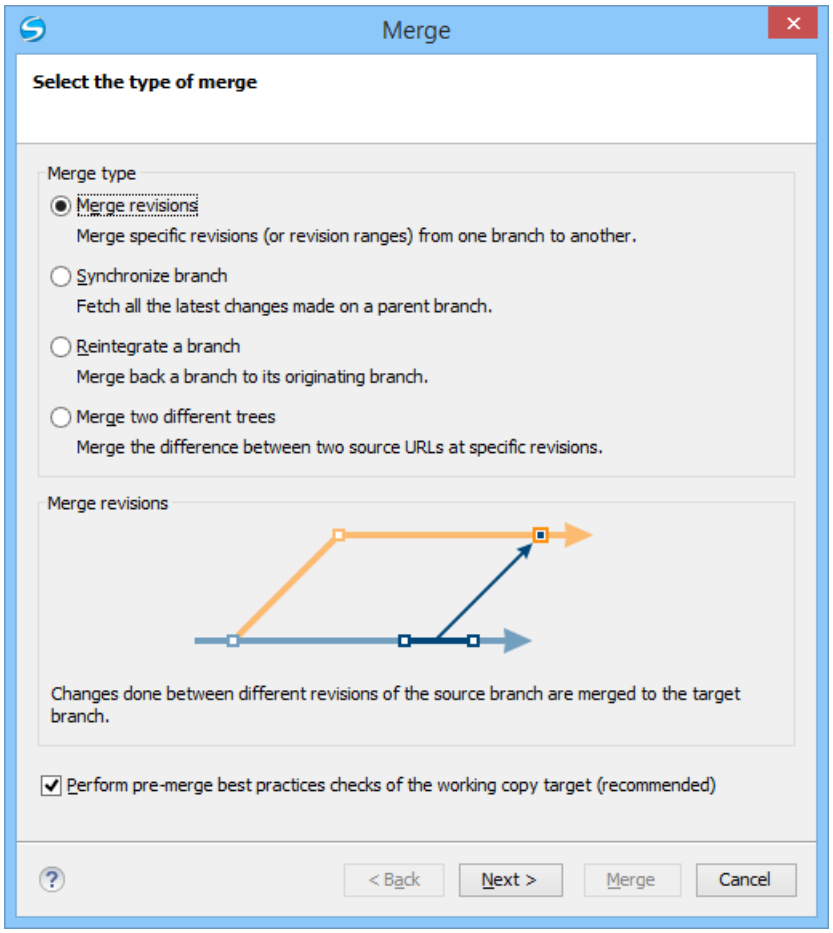


Figure 658: Merge Wizard

The four types of merging are as follows:

- ***Merge revisions*** - Port changes from one branch to another. Note that the *trunk* can also be considered a branch, in this context.
- ***Synchronize branch*** - Fetch all the changes made on a parent branch (or the *trunk*) to a child branch.
- ***Reintegrate a branch*** - Merge a branch back to its parent branch (can also be the *trunk*).
- ***Merge two different trees*** - Integrate the changes done on a branch to a different branch.

It is recommended that you enable the following pre-merge check:

- ***Perform pre-merge best practices checks of the working copy target*** - When selected, the SVN Client checks if the working copy target item is ready for the merge operation and displays the **pre-merge checks** wizard page.

Remember: It is a good idea to perform a merge into an unmodified working copy. If you have made changes to your working copy, commit them first. If the *merge* does not go as you expect, you may want to revert the changes and revert cannot recover your uncommitted modifications.

Important: The above recommendation becomes mandatory when *reintegrating a branch*.

Pre-Merge Checks

Before performing a merge, it is recommended to make sure that the working copy target item is ready for the merge operation. The SVN Client includes a best practices step that checks various conditions of the working copy target item to ensure that the merge operation will succeed. By selecting the **Perform pre-merge best practices checks of the working copy target** option in the first page of the **Merge** wizard, the **Pre-merge checks** wizard page is displayed to give you a summary of the verified conditions.

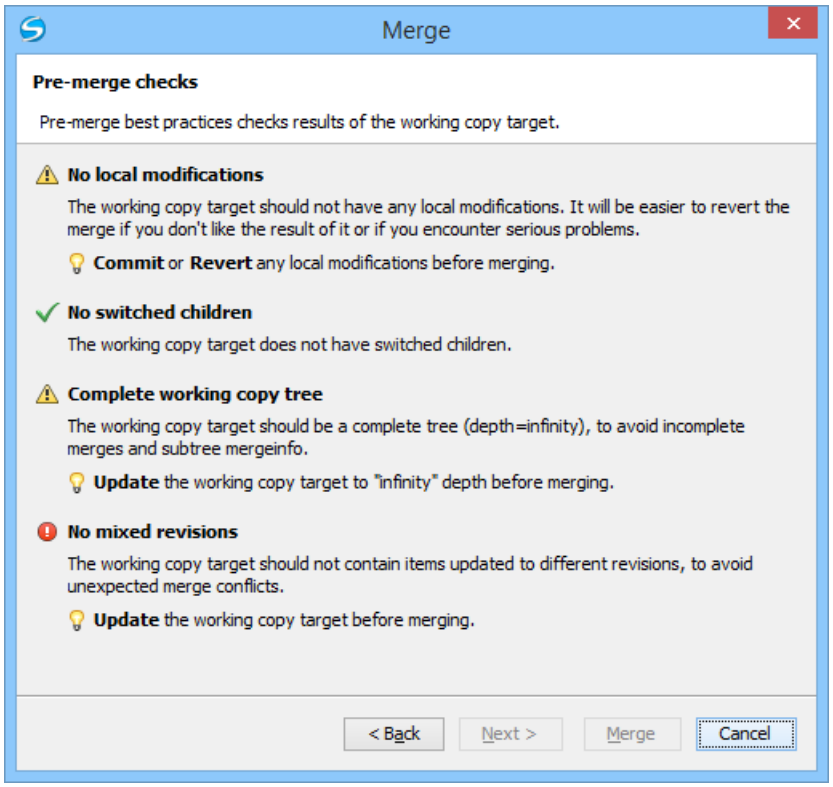


Figure 659: Pre-Merge Checks Wizard Page

The following conditions are checked in this operation:

No local modifications

The working copy item (or any of its children) receiving the merge should not contain uncommitted changes, to make it easier to revert merge-generated changes if you encounter unexpected results.

Tip: If this condition fails, you should *commit* or *revert* the local modifications before merging.

No switched children

None of the children of the working copy item receiving the merge should be switched, to avoid incomplete merges and *subtree mergeinfo*.

Tip: If this condition fails, you should switch back all the children before merging.

Complete working copy tree

The working copy item receiving the merge should be a complete directory tree structure with an infinite depth, to avoid incomplete merges and *subtree mergeinfo*.

Tip: If this condition fails, you should change the *sticky* depth of the working copy item receiving the merge to *infinity* value.

No mixed revisions

To avoid unexpected merge conflicts, the working copy item that is receiving the merge should not contain items that were updated to other revisions.

Tip: If this condition fails, you should *update* the working copy before merging.

Each condition is marked with an icon that represents the state of the condition. The possible states are as follows:

- ✓ (**Successful**) - The condition is fulfilled successfully.
- ⚠ (**Warning**) - The condition is not fulfilled, but it is not mandatory.
- ! (**Error**) - The condition is not fulfilled and is mandatory (therefore, the operation cannot proceed until you solve the error).

Tip: For each condition state, a message is displayed that gives you additional information about the results and, for warning or errors, a hint that explains how you can solve them.

Important: After solving any of the warnings or errors, it is recommended that you perform the *pre-merge checks* again to make sure your new changes are valid.

Merge Revisions

This case is when you have made one or more changes to a branch and you want to duplicate them in another branch. For example, we know that a problem has been fixed by committing revisions 17, 20, and 25 on branch B1. These changes are also needed in branch B2. Thus, to merge them, we need a working copy of the B2 branch.

To merge revisions from a different branch, follow these steps:

1. Go to menu **Tools** > **Merge**.
   The **Merge** wizard is opened.
2. Select the **Merge revisions** option.
3. It is recommended that you select the **Perform pre-merge best practices checks of the working copy target** option to make sure that the working copy target item is ready for the merge operation.
   a) Press the **Next** button.
      If the **Perform pre-merge best practices checks of the working copy target** option is selected, *the **Pre-Merge Checks** wizard page* is displayed.

      Note: If errors are found you need to solve them before proceeding.
4. Press the **Next** button.
   The **Merge revisions** wizard page is displayed.
5. In the **Merge from (URL)** text box, enter *the URL of the branch or tag* that contain the changes that you want to duplicate in your working copy. In our example, it is the URL of the B1 branch.

   You may also click the **Browse** button to browse the repository and find the desired branch. If you have previously merged from this branch, then you can simply use the drop down menu, which displays a history of previously used URLs.

   Note: If the URL belongs to a different repository than the working copy, the **Ignore ancestry / Disable merge tracking** option (in the ***Merge Options** wizard page*) will be selected automatically (and you cannot change this). This is because the *Subversion client cannot track changes between different repositories*.

   Tip: You can also specify a *peg revision* at the end of the URL (for example, `URL@rev1234`). The peg revision does not affect the merge range you select. By default, the HEAD revision is assumed.
6. In the **Revisions to merge** section, choose between the **all revisions** and **specific revision(s)** options.

   - **all revisions** - The operation will include *all eligible revisions* that were not yet merged.
   - **specific revision(s)** - You can specify one or more individual revisions and/or revision ranges. Also, you can mix *forward* ranges (for example, `1-5`), *backward* ranges (for example, `20-15`), and subtract specific revisions from a range (for example, `1-5, -3`).

   Note: If using the Subversion command-line client, a revision range of the form `1-5` means all changes starting from revision 2 up to revision 5 (the changes necessary to reach revision 5, committed after revision 1). Unlike the Subversion command-line client, in Syncro SVN Client the revision ranges are inclusive, meaning that it will process all revisions, starting with revision 1, up to and including revision 5.

   Attention: The HEAD revision is the only non-numerical revision allowed, and it can only be used when specifying revision ranges as one of the ends of the range (for example, `10-HEAD`). Be careful when using it, as it might not refer to the desired revision, if it has recently been committed by another user.

   Tip: If you want to perform a *reverse merge* and roll-back your working copy changes that have already been committed to the repository, use the *negative revisions* notation (for example, `-7`) or *backward revision ranges* (for example, `20-10`).

   a) If you press the **History** button, *the **History** dialog box* is displayed, which allows you to select one or more revisions to be merged.
7. Optionally, if you want to *configure the options* for your merge, press the **Next** button.
   The ***Merge Options** wizard page* is displayed that allows you to configure options for the operation.

Warning: If the **Ignore ancestry / Disable merge tracking** option is selected and you chose **all revisions** in the **Revisions to merge** section, revisions that were previously merged will also be included, which may result in conflicts.

8. Press the **Merge** button.
   The merge operation is performed.

If the merge is completed successfully, all the changes corresponding to the selected revisions should be merged in your working copy.

It is recommended to look at the results of the merge, in the working copy, to review the changes and see if it meets your expectations. Since merging can sometimes be complicated, *you may need to resolve conflicts* after making major changes.

Note: The merge result is only in your local working copy and needs to be committed to the repository for it to be available to others.

Synchronize a Branch

While working on your own branch, other people on your team might continue to make important changes in the parent branch (which can be the *trunk* itself or any other branch). It is recommended to periodically duplicate those changes in your branch to make sure your changes are compatible with them. This is done by performing a *synchronize merge*, which will bring your branch up-to-date with any changes made to its ancestral parent branch since your branch was last created or synchronized. Subversion is aware of the history of your branch and can detect when it split away from the parent branch.

Frequently keeping your branch in sync with the parent branch helps you to prevent unexpected conflicts when the time comes for you to duplicate your changes back into the parent branch. The synchronization uses *merge tracking* to skip all those revisions that have already been merged, thus a sync merge can be repeated periodically to fetch all the latest changes of the parent branch to keep up-to-date with it.

Important: It is recommended to synchronize the whole working copy that was created from the child branch (the root of the working copy), rather than just a part of it.

After running the *synchronize merge*, your working copy from the child branch now contains new local modifications, and these edits are duplications of all of the changes that have happened on the *trunk* since you first created your branch. At this point, your private branch is now synchronized with the trunk.

To synchronize your branch with its parent branch, follow these steps:

1. Go to **Tools** > **Merge**.
   The **Merge** wizard is opened.
2. Select the **Synchronize branch** option.
3. It is recommended that you select the **Perform pre-merge best practices checks of the working copy target** option to make sure that the working copy target item is ready for the merge operation.
   a) Press the **Next** button.
      If the **Perform pre-merge best practices checks of the working copy target** option is selected, *the **Pre-Merge Checks** wizard page* is displayed.

      Note: If errors are found you need to solve them before proceeding.
4. Press the **Next** button.
   The **Synchronize branch** wizard page is displayed.
5. In the **Parent branch (URL)** text box, enter *the URL of the branch from which you created your branch*. This means that the URL must belong to the same repository as your working copy that was created from the child branch.

   You may also click the **Browse** button to browse the repository and find the desired branch. If you have previously merged from this branch, then you can simply use the drop down menu, which displays a history of previously used URLs.

   Tip: You can also specify a *peg revision* at the end of the URL (for example, `URL@rev1234`). The peg revision specifies both the peg revision of the URL and the latest revision that will be considered for merging. By default, the `HEAD` revision is assumed.
6. Optionally, if you want to *configure the options* for your merge, press the **Next** button.

The ***Merge Options*** *wizard page* is displayed that allows you to configure options for the operation.

Note: The **Ignore ancestry / Disable merge tracking** option is not available for this merge type, since a synchronization merge should always be recorded in the destination branch.

7. Press the **Merge** button.
   The merge operation is performed.

If the merge is completed successfully, all the changes corresponding to the selected revisions should be merged in your working copy.

It is recommended to look at the results of the merge, in the working copy, to review the changes and see if it meets your expectations. Since merging can sometimes be complicated, *you may need to resolve conflicts* after making major changes.

Note: The merge result is only in your local working copy and needs to be committed to the repository for it to be available to others.

Reintegrate a Branch

There are some conditions that apply to reintegrate a branch:

- The server must support merge tracking.
- The source branch (to be reintegrated) must be coherently synchronized with its parent branch. This means that all revisions between the branching point and the last revision merged from the parent branch to the child branch must be merged to the latter one (there must be no missing revisions in-between).
- The working copy **must not** contain the following:
  - Local modifications.
  - A mixture of revisions (all items must point to the same revision).
  - Sparse directories (all directories must be of **infinity** depth).
  - Switched items.
- The revision of the working copy must be greater than or equal to the last revision of the parent branch with which the child branch was synchronized.

Tip: You can use *the pre-merge checks option* to make sure these conditions are fulfilled.

This method is useful when you have a feature branch on which the development has concluded and it should be merged back into its parent branch. Since you have kept the feature branch synchronized with its parent, the latest versions of them will be absolutely identical except for your feature branch changes. These changes can be reintegrated into the parent branch by using a working copy of it and the **Reintegrate a branch** option.

This method uses the *merge-tracking* features of Apache Subversion™ to automatically calculate the correct revision ranges and to perform additional checks that will ensure that the branch to be reintegrated has been fully updated with its parent changes. This ensures that you do not accidentally undo work that others have committed to the parent branch since the last time you synchronized the child branch with it. After the merge, all branch development will be completely merged back into the parent branch, and the child branch will be redundant and can be deleted from the repository.

Tip: Before reintegrating the child branch it is recommended to synchronize it with its parent branch one more time, to help avoid any possible conflicts.

To reintegrate a child branch into its parent branch, follow these steps:

1. Go to menu **Tools** > **Merge**.
   The **Merge** wizard is opened.
2. Select the **Reintegrate a branch** option.

   Note: This option is not available if the selected working copy item (or if it is a directory, any of the items inside of it) has any type of modification. This is because it is mandatory for the target item to have no modifications.
3. It is recommended that you select the **Perform pre-merge best practices checks of the working copy target** option to make sure that the working copy target item is ready for the merge operation.
   a) Press the **Next** button.

If the **Perform pre-merge best practices checks of the working copy target** option is selected, *the **Pre-Merge Checks** wizard page* is displayed.

Note: If errors are found you need to solve them before proceeding.

4. Press the **Next** button.
   The **Reintegrate a branch** wizard page is displayed.
5. In the **Child branch (URL)** text box, enter *the URL of the child branch to be reintegrated*. This means that the URL must belong to the same repository as your working copy that was created from the parent branch.

   You may also click the **Browse** button to browse the repository and find the desired branch. If you have previously merged from this branch, then you can simply use the drop down menu, which displays a history of previously used URLs.

   Tip: You can also specify a *peg revision* at the end of the URL (for example, `URL@rev1234`). The peg revision specifies both the peg revision of the URL and the latest revision that will be considered for merging. By default, the `HEAD` revision is assumed.

   The ***Merge Options** wizard page* is displayed that allows you to configure options for the operation.

   Note: Since a *reintegrate merge* is so specialized, most of the merge options are not available, except for those in the **File Comparison** category.
6. Press the **Merge** button.
   The merge operation is performed.

If the merge is completed successfully, all the changes corresponding to the selected revisions should be merged in your working copy.

It is recommended to look at the results of the merge, in the working copy, to review the changes and see if it meets your expectations. Since merging can sometimes be complicated, *you may need to resolve conflicts* after making major changes.

Note: The merge result is only in your local working copy and needs to be committed to the repository for it to be available to others.

Merge Two Different Trees

This merge type is useful when you need to duplicate changes from one child branch (for example, `CB1`) to another child branch (`CB2`) from the same parent branch. The SVN client will calculate the changes necessary to get from the `HEAD` revision of the parent branch (or the *trunk*) to the `HEAD` revision of one of its child branches (`CB1`), and apply those changes to your working copy of the other branch (`CB2`). The result is that the latter child branch (`CB2`) will also include the changes made on the original child branch (`CB1`), although that branch was not reintegrated into the parent branch.

This merge type could also be used to reintegrate a child branch back into its parent when the repository does not support *merge tracking*.

Note: If the server does not support *merge-tracking*, then this is the only way to merge a branch back to its parent.

1. Go to menu **Tools** > **Merge**.
   The **Merge** wizard is opened.
2. Select the option **Merge two different trees**.
3. It is recommended that you select the **Perform pre-merge best practices checks of the working copy target** option to make sure that the working copy target item is ready for the merge operation.
   a) Press the **Next** button.
      If the **Perform pre-merge best practices checks of the working copy target** option is selected, *the **Pre-Merge Checks** wizard page* is displayed.

      Note: If errors are found you need to solve them before proceeding.
4. Press the **Next** button.
   The **Merge two different trees** wizard page is displayed.
5. In the **From (starting URL and revision)** section enter *the URL of the first branch*.

You may also click the **Browse** button to browse the repository and find the desired branch. If you have previously merged from this branch, then you can simply use the drop down menu, which displays a history of previously used URLs.

Tip: If you are using this method to merge a feature branch back to its parent branch, you need to start the merge wizard from within a working copy of the parent. In this field enter the full URL of the parent branch. This may sound wrong, but remember that the parent is the starting point to which you want to add the branch changes.

Note: If the URL belongs to a different repository than the working copy, the **Ignore ancestry / Disable merge tracking** option (in the ***Merge Options*** *wizard page*) will be selected automatically (and you cannot change this). This is because the *Subversion client cannot track changes between different repositories*.

Tip: You can also specify a *peg revision* at the end of the URL (for example, `URL@rev1234`). By default, the HEAD revision is assumed.

6. Enter the last revision number at which the two trees were synchronized by choosing between **HEAD revision** and **other revision**.
   - **HEAD revision** - Use this option if you are sure that no one else has committed changes since the last synchronization.
   - **other revision** - Use this option to input a specific revision number and avoid losing recent commits. You can use the **History** button to see a list of all revisions.
7. In the **To (ending URL and revision)** section enter *the URL of the second branch*.

   You may also click the **Browse** button to browse the repository and find the desired branch. If you have previously merged from this branch, then you can simply use the drop down menu, which displays a history of previously used URLs.

   Tip: If you are using this method to merge a feature branch back to its parent branch, enter the URL of the feature branch. This way, only the changes unique to this branch will be merged, since the branch should have been periodically synchronized with its parent.

   Attention: The URL must point to the same repository as the one in the **From (starting URL and revision)** field. Otherwise, the operation will not be allowed, since Subversion cannot compute changes between items from different repositories.

   Tip: You can also specify a *peg revision* at the end of the URL (for example, `URL@rev1234`). By default, the HEAD revision is assumed.
8. Select a revision to compute all changes committed up to that point by choosing between **HEAD revision** and **other revision**.
   - **HEAD revision** - This is the default selected revision.
   - **other revision** - Use this option if you want to enter a previous revision. You can use the **History** button to see a list of all revisions.
9. Optionally, if you want to *configure the options* for your merge, press the **Next** button.
   The ***Merge Options*** *wizard page* is displayed that allows you to configure options for the operation.

   Warning: If the **Ignore ancestry / Disable merge tracking** option is selected and you chose **all revisions** in the **Revisions to merge** section, revisions that were previously merged will also be included, which may result in conflicts.
10. Press the **Merge** button.
    The merge operation is performed.

If the merge is completed successfully, all the changes corresponding to the selected revisions should be merged in your working copy.

It is recommended to look at the results of the merge, in the working copy, to review the changes and see if it meets your expectations. Since merging can sometimes be complicated, *you may need to resolve conflicts* after making major changes.

Note: The merge result is only in your local working copy and needs to be committed to the repository for it to be available to others.

Merge Options

Here is the list of options that can be used when merging:

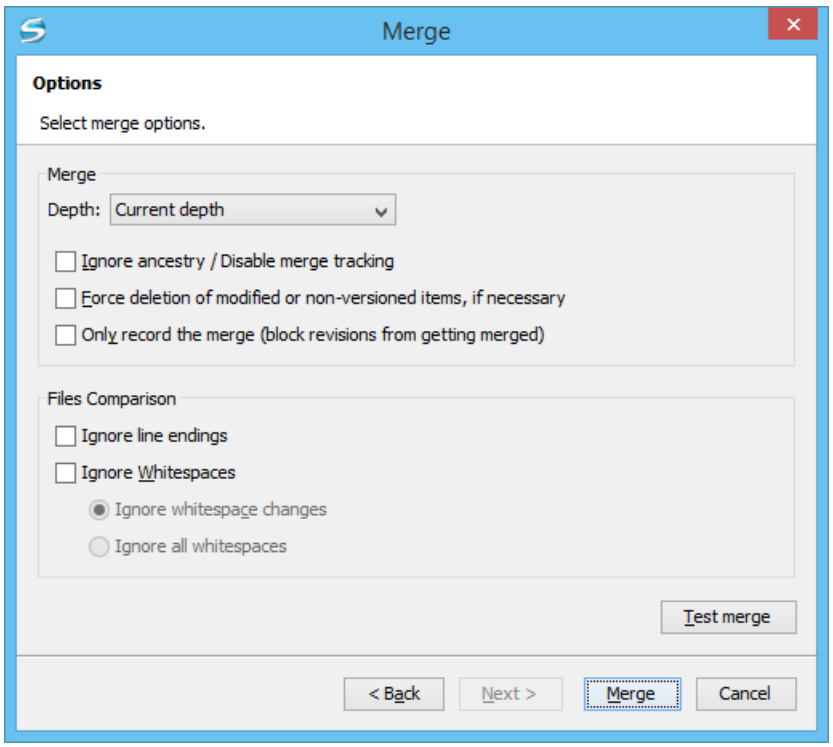


Figure 660: Merge Wizard - Advanced Options

- **Depth** (This option is applicable only for directories) - sets the depth of the merge operation. You can specify how far down into your working copy the merge should go by selecting one of the following values:
  - **Current depth** - Obeys the depths registered for the directories in the working copy that are to be switched.
  - **Recursive (infinity)** - Merges all the files and folders contained in the selected folder. This is the recommended depth for most users, to avoid incomplete merges and *subtree mergeinfo*.
  - **Immediate children (immediates)** - Merges only the child files and folders without recursing subfolders.
  - **File children only (files)** - Merges only the child files.
  - **This folder only (empty)** - Merges only the selected folder (no child files or folders are included).

  Note: The *depth* term is described in the *Sparse checkouts* section. The default depth is the current depth of the working copy item receiving the merge.
- **Ignore ancestry / Disable merge tracking** - Changes the way two items are merged if they do not share a common ancestry. Most merges involve comparing items that are ancestrally related to one another. However, occasionally you may want to merge unrelated items. If this option is not selected, the first item will be replaced with the second item. In these situations, you would want the merge to do a path-based comparison only, ignoring any relations between the items. For example, if two different files have the same name and are in the same relative location, deselecting the option replaces one of the files with the other one, and selecting it merges their contents.

  Note: If the URL of the merge source belongs to a different repository than the URL of the target working copy item (the one receiving the changes), this option is selected automatically (and you cannot change this). This is because the *Subversion client cannot track changes between different repositories*.
- **Force deletion of modified or non-versioned items, if necessary** - If not selected, when the merge operation involves deleting locally modified or non-versioned items, it will fail. This is done to prevent data loss. This option is only available if there are uncommitted changes in the working copy.
- **Only record the merge (block revisions from getting merged)** - Available when the ***Ignore ancestry / Disable merge tracking*** *option* is not selected. It enables a special mode of the merge operation that just records it in the local merge tracking information, without actually performing it (does not modify any file contents or the structure of your working copy). You might want to select this option for two possible reasons:

    - You made (or will make) the merge manually, and therefore need to mark the revisions as being merged to make the merge tracking system aware of them. This will exclude them from future merges.
    - You want to prevent one or more particular changes from being fetched in subsequent merges.
- **Ignore line endings** - Allows you to specify how the line ending changes should be handled. By default, all such changes are treated as real content changes, but you can ignore them if you select this option.
- **Ignore whitespaces** - Allows you to specify how the whitespace changes should be handled. By default, all such changes are treated as real content changes, but you can ignore them if you select this option.
    - **Ignore whitespace changes** - Ignores changes in the amount of whitespaces or to their type (for example, when changing the indentation or changing tabs to spaces).

      Note: Whitespaces that were added where there were none before, or that were removed, are still considered to be changes.
    - **Ignore all whitespaces** - Ignores all types of whitespace changes.
- **Test merge** - Performs a dry run of the merge operation, allowing you to *preview* it without actually performing the merge. In the **Console** view you will see a list of the working copy items that will be affected and how they will be affected. This is helpful in detecting whether or not a merge will be successful, and where conflicts may occur.

Resolving Merge Conflicts

After the merge operation is finished, it is possible to have some items in conflict. This means that some incoming modifications for an item could not be merged with the current working copy version. If there are such conflicts, the **Merge conflicts** dialog box will appear, presenting the items that are in conflict. This dialog box offers you choices for resolving the conflicts.

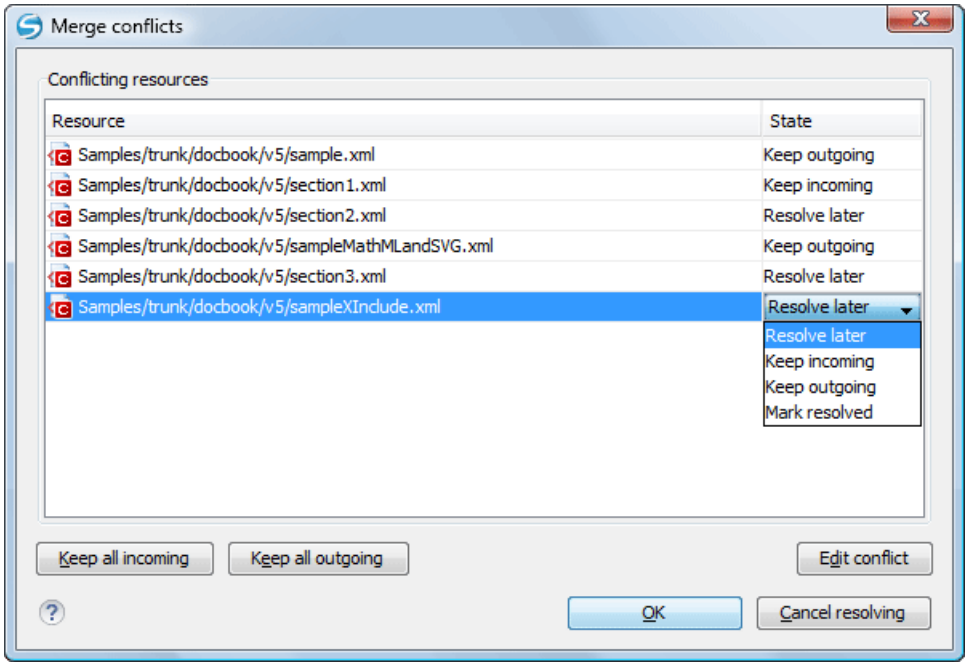


Figure 661: Merge Conflicts Dialog Box

The options to resolve a conflict are as follows:

- **Resolve later** - Used for leaving the conflict as it is, to manually resolve it later.
- **Keep incoming** - This option keeps all the incoming modifications and discards all current ones from your working copy.
- **Keep outgoing** - This option keeps all current modifications from your working copy and discards all incoming ones.
- **Mark resolved** - You should choose this option after you have manually solved the conflict, to instruct the Subversion that it was resolved. To do this, use the **Edit conflict** button, which displays a dialog box that presents the contents of the conflicting items (the content of the working copy version versus the incoming version).

Sub-tree Merges

It is recommended to perform a merge on the whole working copy (select its root directory when triggering the operation) to avoid *sub-tree mergeinfo*. *Sub-tree mergeinfo* is the *mergeinfo* recorded to describe a *sub-tree merge*. That is, a merge done directly to a child of a branch root that might be needed in certain situations. There is nothing special about *sub-tree merges* or *sub-tree mergeinfo* except that the complete record of merges to a branch may not be contained solely in the *mergeinfo* on the branch root and you may have to look to any *sub-tree mergeinfo* to get a full accounting. Fortunately, Subversion does this for you and rarely will you need to look for it.

Merging from Foreign Repositories

Subversion supports merging from foreign repositories. While all merge source URLs must point to the same repository, the merge target (from the working copy) may come from a different repository than the source. However, copies made in the merge source will be transformed into plain additions in the merge target. Also, *merge-tracking* is not supported for merges from foreign repositories.

Note: When performing merges from repositories other than the one corresponding to the target item (from the working copy), the ***Ignore ancestry / Disable merge tracking*** *option* in the ***Merge Options*** *wizard page* will be selected automatically (and you cannot change this).

General Merge Recommendations

As a recommendation, you should only merge into clean working copies that **do not** contain any of the following:

- Modifications.
- Sparse directories (all directories must be of depth *infinity*).
- Switched items.

Important: This recommendation becomes mandatory when performing a *reintegrate merge* operation. Also, trying to merge to mixed-revision working copies will fail in all types of merge operations.

Remember: The merge result is only in your local working copy and needs to be committed to the repository for it to be available to others.

Switch the Repository Location

The **Switch** action is useful when the repository location of a working copy, or an already committed item in the working copy, must be changed within the same repository. The action is available on the **Tools** menu when a versioned resource is selected in the current working copy that is displayed in the ***Working Copy*** *view*.

Note: *External* items cannot be switched using this action. Instead, change the value of the `svn:externals` property set on the parent directory of the external item and update the parent directory.

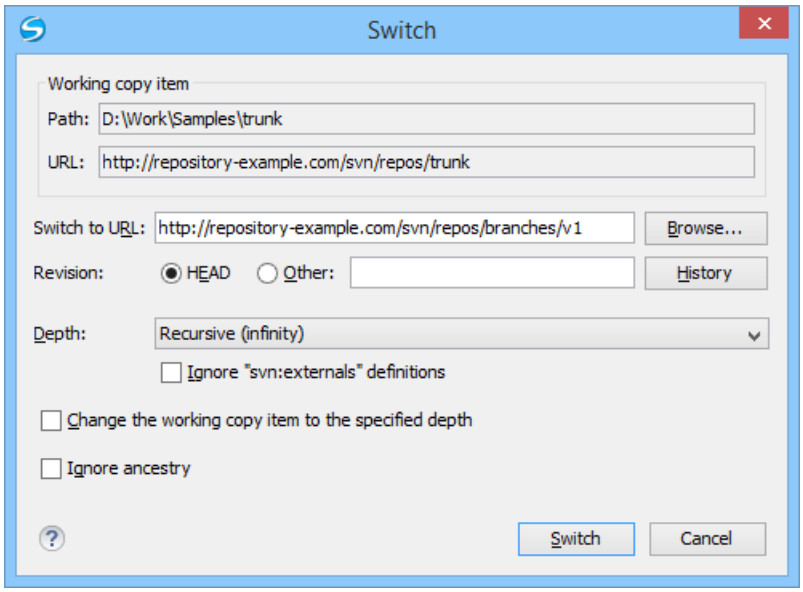


Figure 662: Switch Dialog Box

The following options can be configured in the **Switch** dialog box:

Switch to URL

The new location in the same repository you are switching to.

Tip: You can switch to items that were deleted, moved, or replaced, by specifying the original URL (before the item was removed) and use a *peg revision* at the end (for example, URL`@rev1234`).

Note: For items that are already *switched* that you want to switch back, the proposed URL is the original location of the item.

Revision

The specific version of the location that you are switching to.

Depth - (This option is applicable only for directories)

Current depth

Obeys the depths registered for the directories in the working copy that are to be switched.

Recursive (infinity)

Switches the location of the selected folder and all of its files and folders.

Immediate children (immediates)

Switches the location of the selected folder and its child files and folders without recursing subfolders.

File children only (files)

Switches the location of the selected folder and its child files.

This folder only (empty)

Switches the location of the selected folder (no child files or folders are included).

Ignore "svn:externals" definitions

When selected, external items are ignored in the switch operation. This option is only available if you choose the **Current depth** or **Recursive (infinity)** depth.

Change the working copy item to the specified depth

Changes the *sticky* depth on the directory in the working copy.

Ignore ancestry

When not selected, the SVN system does not allow you to switch to a location that does not share a common ancestry with the current location. If selected, the SVN does not check for a common ancestry.

Relocate a Working Copy

Sometimes the base URL of the repository is changed after a working copy is checked out from that URL. For example, if the repository itself is moved to a different server. In such cases, you do not have to check out a working copy from the new repository location. It is easier to change the base URL of the root folder of the working copy to *the new URL of the repository*.

Note: *Peg revisions* have no effect for this operation.

This **Relocate** action is available in the **Tools** menu when selecting the root item of the working copy.

Note: *External* items that are defined using absolute URLs and that point to the same repository as the working copy are not affected by the **Relocate** action (the URL is not updated). To relocate these items, change the value of the `svn:externals` property for each external item (set on their parent directories). For example, if an external item is defined as `externalDir http://host/path/to/repo/to/dir` and the repository was moved to another server (`host2`) and its protocol changed to `https`, then the value of the property needs to be updated to `externalDir https://host2/path/to/repo/to/dir`.

Tip: If you edit external items using the method described in the previous note, on the next update the system will try to fetch the external items again. To avoid this, you can invoke the **Relocate** action on each of these external items.

Patches

This section explains how to work with patches in Syncro SVN Client.

What is a Patch

Suppose you are working with a set of XML files that you want to tag the project and distribute releases to other team members. While working on the project and correcting problems, you may need to distribute the corrections to other team members. In this case, you can distribute a patch (a collection of differences) that would correct the problems when applied over the last distribution. By default, the Syncro SVN Client generates patches in *the Unified Diff format*, but it can also use *the Git format*.

Creating a patch in Apache Subversion™ implies the access to either two revisions of a versioned item, or two different versioned items from the repository:

- *Two revisions of a version item* - the item can be local or remote and you can select two versions of it. This also applies when you need to generate a patch that only contains the changes in the working copy that were not yet committed.
- *Two different versioned items from the repository* - the items are remote and you need to specify a revision for both.

Warning: The resulting patch file may contain content that was written using a mix of encodings, based upon the encodings of the files that were compared. If you open the generated patch file in a text editor, it may result in unrecognizable content.

Generating a Patch - Local Items

Based on a versioned item (already committed or scheduled for addition) in the working copy, you can generate patches that contain the local changes, the differences between a specific revision of that item and the item itself, or differences between the pristine item and another item from the repository. There are four options for generating a patch based upon local items.

To open the **Create patch** wizard, use the **Create patch** action from the **Tools** menu or from the contextual menu in the **Modified**, **Incoming**, **Outgoing**, or **Conflicts** modes.

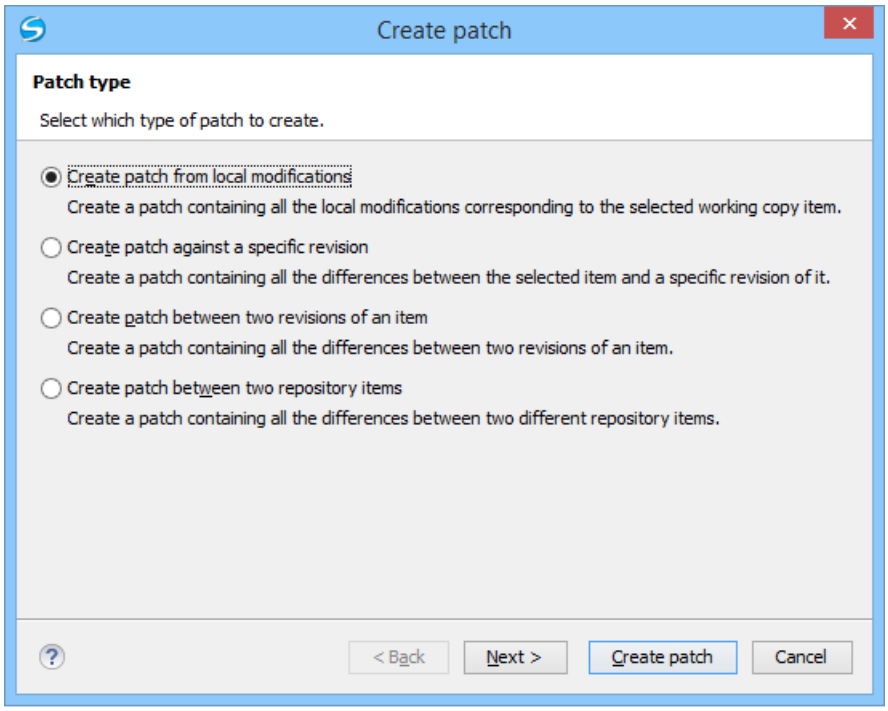


Figure 663: Create Patch Wizard - Local Items

Create Patch from Local Modifications

This is the most commonly used type of patch and contains only the local changes for the selected item.

This option is useful if you want to share changes with other team members and you are not yet ready to commit them. This option is only available for local items that contain modifications. It is not available for items in which the local file status is *unversioned or ignored, and in some cases missing or obstructed*.

The steps are as follows:

1. Go to menu **Tools** > **Create patch**.
   This opens the **Create patch** wizard.
2. Select the **Create patch from local modifications** option in the dialog box.
3. Optionally, if you want *to configure the options* for your patch, press the **Next** button.

   This options page does not remember your selections when creating future patches. It will revert to the default values.

   The **Options** wizard page is displayed.
4. Press the **Create patch** button.

   If the patch is applied on a folder of the working copy and that folder contains *unversioned files*, this step of the wizard offers the option of selecting the ones that will be included in the patch.

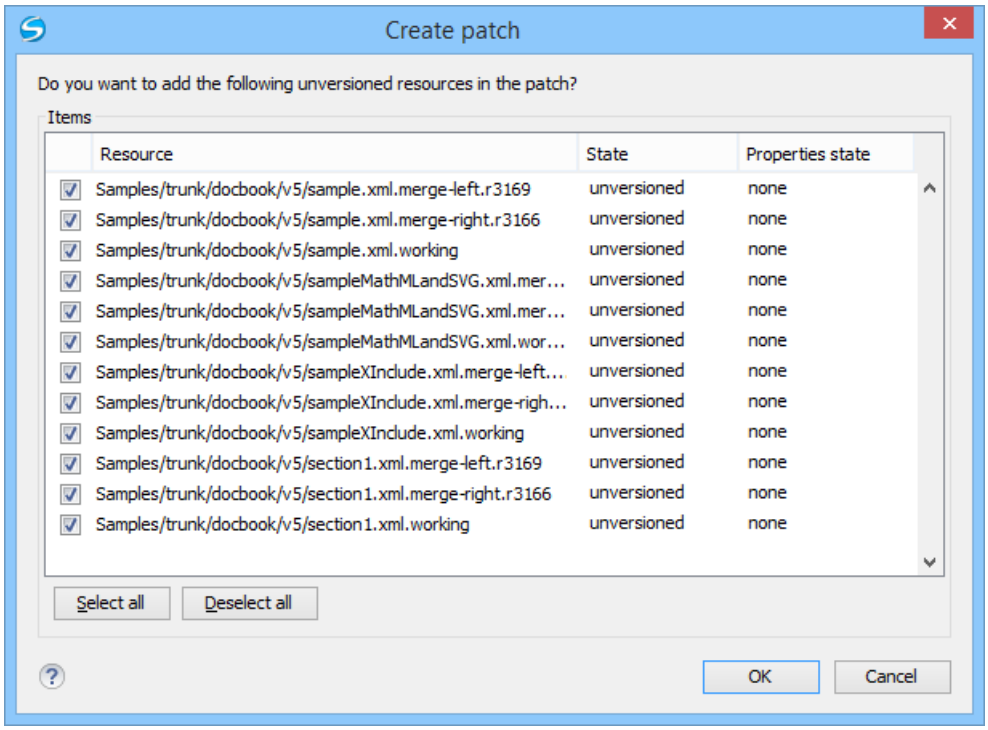


Figure 664: Create Patch Dialog Box - Add Unversioned Resources

The patch is created and stored in the path specified in *the **Output** section of the **Options** page* or in the default location.

Create Patch Against a Specific Revision

This type of patch contains changes between an old revision and the current content from the selected item within the working copy.

This option is useful if you want to obtain differences between an older revision and the current state of the working copy (for instance, to test how current changes apply to an older version).

The steps are as follows:

1. Go to menu **Tools** > **Create patch**.
   This opens the **Create patch** wizard.
2. Select the **Create patch against a specific revision** option in the dialog box.
3. Press the **Next** button.
   The second step of the wizard is opened:

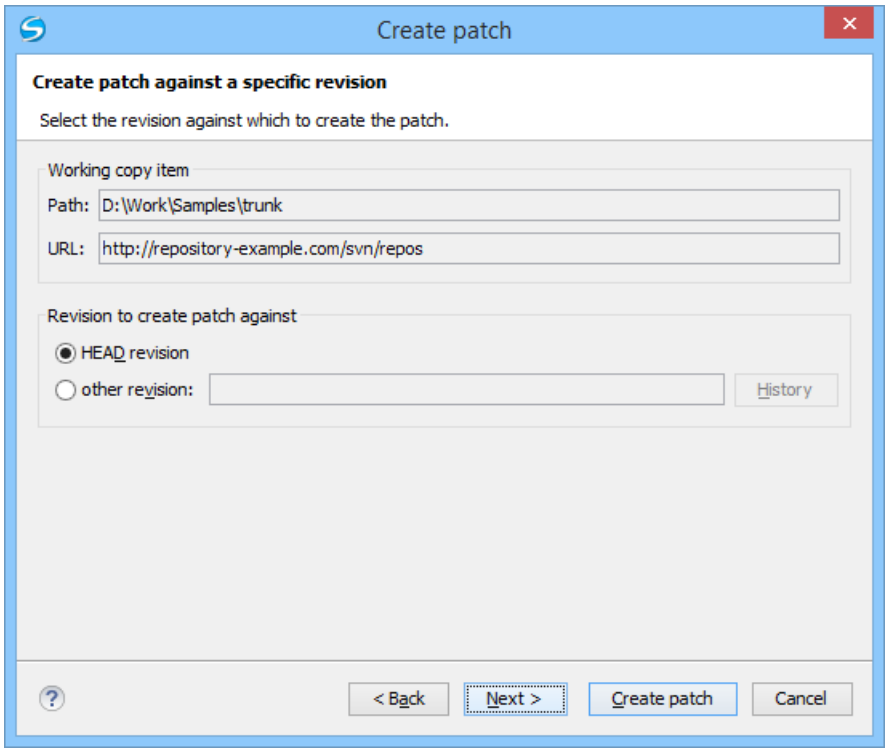


Figure 665: Create Patch Wizard - Step 2

4. Select the **revision to create patch against**.

   You can select between the **HEAD revision** and a specific revision number. For the **other revision** option, you can press *the **History** button* to display a list of the item revisions.

   Note: If the **revision to create patch against** is older than the revision for which the working copy item was updated, the patch will include changes that were made **after** the selected revision.

5. Optionally, if you want *to configure the options* for your patch, press the **Next** button.

   This options page does not remember your selections when creating future patches. It will revert to the default values.

   The **Options** wizard page is displayed.

6. Press the **Create patch** button.
   The patch is created and stored in the path specified in *the **Output** section of the **Options** page* or in the default location.

Create Patch Between Two Revisions of an Item

This type of patch contains historical changes between two revisions of a selected item.

This option is useful if you want to share changes between two revisions with other team members.

Tip: If you need to generate a patch between two revisions of a previously *deleted*, *moved*, or *replaced* item, you should use *the **Create patch between two repository items** option* instead.

The steps are as follows:

1. Go to menu **Tools** > **Create patch**.
   This opens the **Create patch** wizard.
2. Select the **Create patch between two revisions of an item** option in the dialog box.
3. Press the **Next** button.
   The second step of the wizard is opened:

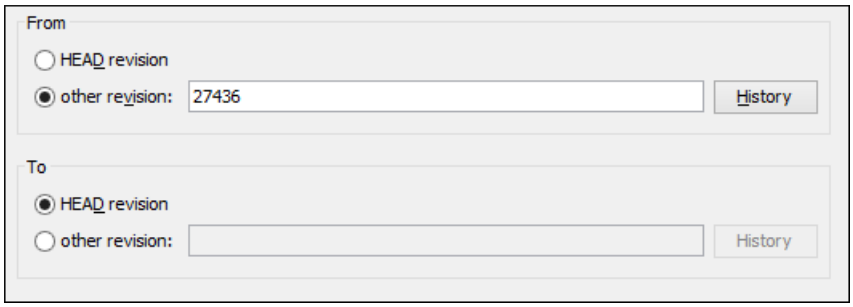


Figure 666: Create Patch Wizard - Step 2

4. Select the starting and ending revisions in the **From** and **To** sections.

   You can select between the **HEAD revision** and a specific revision number. For the **other revision** option, you can press *the **History** button* to display a list of the item revisions.

   Note: The patch will only include changes between the two specified revisions, starting with the changes that were made **after** the older revision.

   Tip: If you want to reverse changes done between two revisions by using a patch file, you can specify the newer revision in the **From** section and the older version in the **To** section.

5. Optionally, if you want *to configure the options* for your patch, press the **Next** button.

   This options page does not remember your selections when creating future patches. It will revert to the default values.

   The **Options** wizard page is displayed.

6. Press the **Create patch** button.
   The patch is created and stored in the path specified in *the **Output** section of the **Options** page* or in the default location.

Create Patch Between Two Repository Items

This type of patch contains changes between one version of an item and a specific version of another item.

This option is useful for generating a patch that contains changes between existing, or even previously deleted, moved, or replaced items from different branches. This is the default option when you do not have a working copy loaded, when no repository items are selected, or when exactly two repository items of the same kind are selected.

Tip: To access an item that was deleted, moved, or replaced, you need to specify the original URL (before the item was removed) and use a *peg revision* at the end (for example, `URL@rev1234`).

The steps are as follows:

1. Go to menu **Tools** > **Create patch**.
   This opens the **Create patch** wizard.
2. Select the **Create patch between two repository items** option in the dialog box.
3. Press the **Next** button.
   The second step of the wizard is opened:

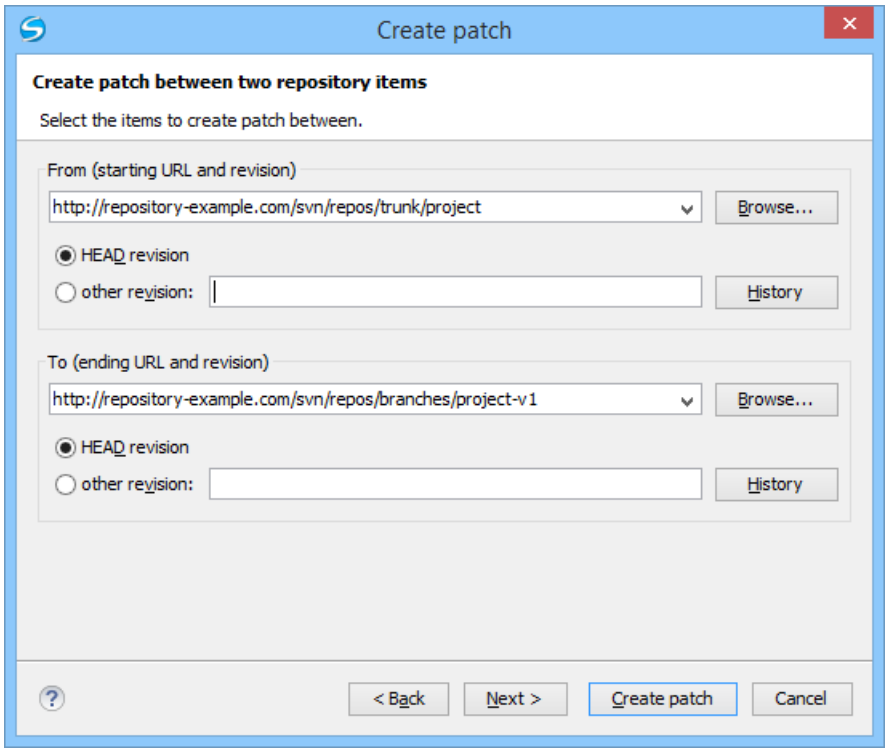


Figure 667: Create Patch Wizard - Step 2

4. Select *the starting and ending URLs* and revisions in the **From** and **To** sections.

   You can select between the **HEAD revision** and a specific revision number. For the **other revision** option, you can press *the **History** button* to display a list of the item revisions.

   Important: Both URLs must point to items from the same repository.

   Note: If you use a *peg* revision in the URL field, anything specified in the **other revision** field is ignored.

5. Optionally, if you want *to configure the options* for your patch, press the **Next** button.

   This options page does not remember your selections when creating future patches. It will revert to the default values.

   The **Options** wizard page is displayed.

6. Press the **Create patch** button.
   The patch is created and stored in the path specified in *the **Output** section of the **Options** page* or in the default location.

Generating a Patch - Remote Items

Based on a repository item, you can generate patches that contain the differences between two specific revisions of that item, or between a revision of that same item and another revision of another item from the repository. There are two options for generating a patch based upon remote items.

To open the **Create patch** wizard, use the **Create patch** action from the **Tools** menu.

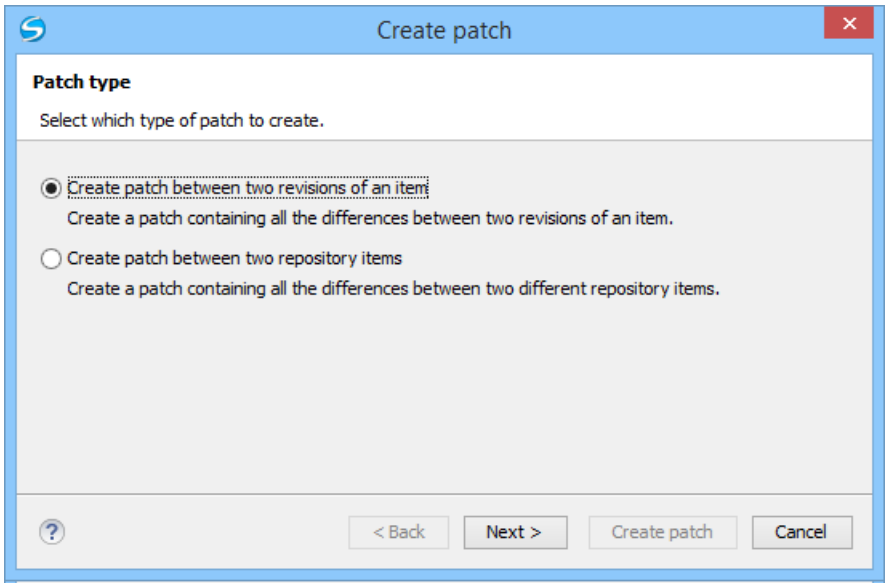


Figure 668: Create Patch Wizard - Remote Items

Create Patch Between Two Revisions of an Item

This type of patch contains historical changes between two revisions of a selected item.

This option is useful if you want to share changes between two revisions with other team members.

Tip: If you need to generate a patch between two revisions of a previously *deleted*, *moved*, or *replaced* item, you should use *the **Create patch between two repository items** option* instead.

The steps are as follows:

1. Go to menu **Tools** > **Create patch**.
   This opens the **Create patch** wizard.
2. Select the **Create patch between two revisions of an item** option in the dialog box.
3. Press the **Next** button.
   The second step of the wizard is opened:

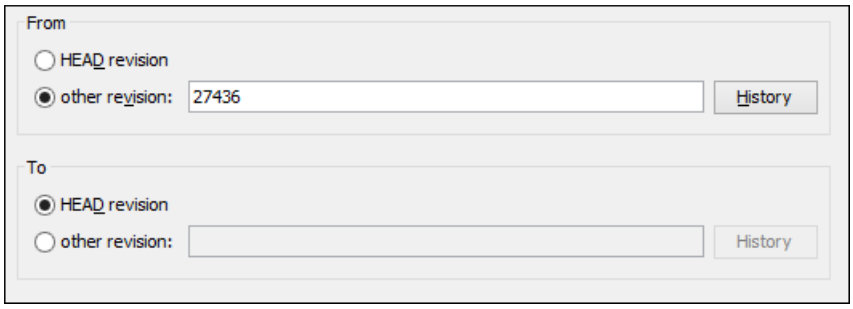


Figure 669: Create Patch Wizard - Step 2

4. Select the starting and ending revisions in the **From** and **To** sections.

   You can select between the **HEAD revision** and a specific revision number. For the **other revision** option, you can press *the **History** button* to display a list of the item revisions.

   Note: The patch will only include changes between the two specified revisions, starting with the changes that were made **after** the older revision.

   Tip: If you want to reverse changes done between two revisions by using a patch file, you can specify the newer revision in the **From** section and the older version in the **To** section.
5. Optionally, if you want *to configure the options* for your patch, press the **Next** button.

   This options page does not remember your selections when creating future patches. It will revert to the default values.

The **Options** wizard page is displayed.

6. Press the **Create patch** button.
   The patch is created and stored in the path specified in *the **Output** section of the **Options** page* or in the default location.

Create Patch Between Two Repository Items

This type of patch contains changes between one version of an item and a specific version of another item.

This option is useful for generating a patch that contains changes between existing, or even previously deleted, moved, or replaced items from different branches. This is the default option when you do not have a working copy loaded, when no repository items are selected, or when exactly two repository items of the same kind are selected.

Tip: To access an item that was deleted, moved, or replaced, you need to specify the original URL (before the item was removed) and use a *peg revision* at the end (for example, `URL@rev1234`).

The steps are as follows:

1. Go to menu **Tools** > **Create patch**.
   This opens the **Create patch** wizard.
2. Select the **Create patch between two repository items** option in the dialog box.
3. Press the **Next** button.
   The second step of the wizard is opened:

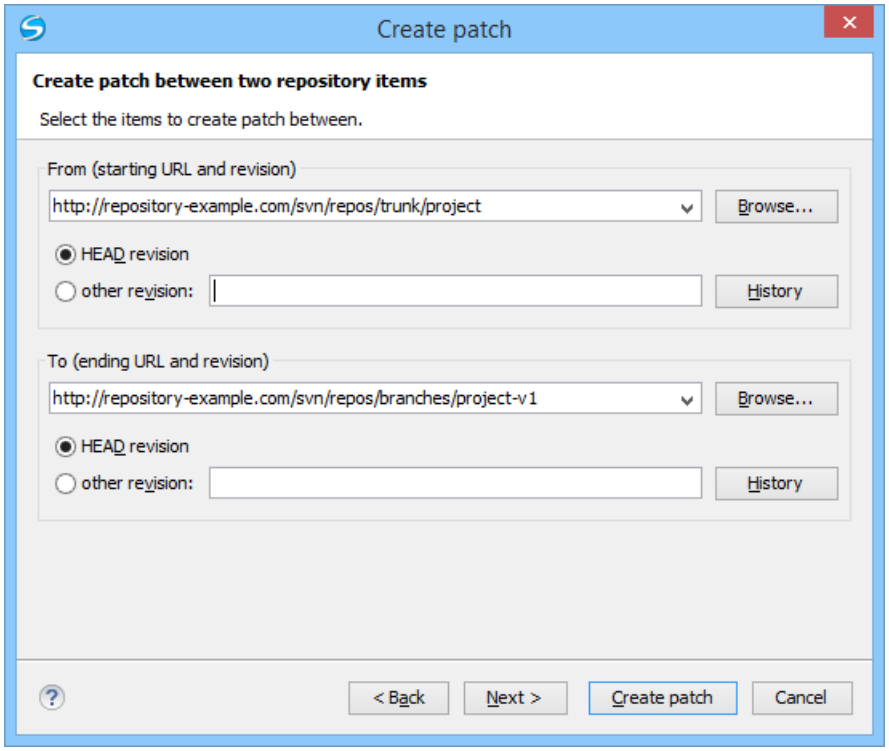


Figure 670: Create Patch Wizard - Step 2

4. Select *the starting and ending URLs* and revisions in the **From** and **To** sections.
   You can select between the **HEAD revision** and a specific revision number. For the **other revision** option, you can press *the **History** button* to display a list of the item revisions.

   Important: Both URLs must point to items from the same repository.

   Note: If you use a *peg* revision in the URL field, anything specified in the **other revision** field is ignored.
5. Optionally, if you want *to configure the options* for your patch, press the **Next** button.
   This options page does not remember your selections when creating future patches. It will revert to the default values.
   The **Options** wizard page is displayed.

6. Press the **Create patch** button.
   The patch is created and stored in the path specified in *the **Output** section of the **Options** page* or in the default location.

Patch Options

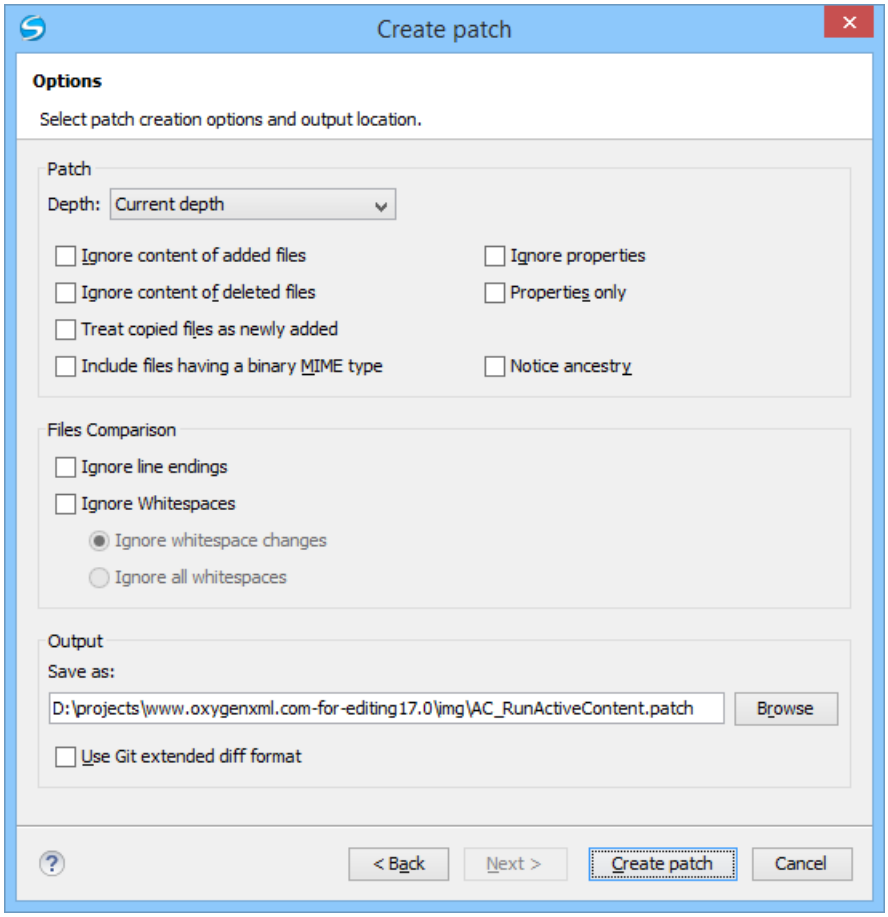


Figure 671: Create Patch Wizard - Options

Patch Section

Depth - (This option is applicable only for directories)

Current depth

The depth of recursing the folder for creating the patch is the same as the depth of that same folder in the working copy (available only when generating patches that contain changes from the working copy).

Recursive (infinity)

The patch is created on all the files and folders contained in the selected folder.

Immediate children (immediates)

The patch is created only on the child files and folders without recursing subfolders.

File children only (files)

The patch is created only on the child files.

This folder only (empty)

The patch is created only on the selected folder (no child file or folder is included in the patch).

Ignore content of added files

When selected, the patch file does not include the content of the *added* items. This option corresponds to the `--no-diff-added` option of the `svn diff` command.

Ignore content of delete files

When selected, the patch file does not include the content of the *deleted* items. This option corresponds to the `--no-diff-deleted` option of the `svn diff` command.

Treat copied files as newly added

When selected, copied items are treated as new, rather than comparing the items with their sources. This option corresponds to the `--show-copies-as-adds` option of the `svn diff` command.

Include files having a binary MIME type

When selected, the application is forced to compare items that are considered binary file types. This option corresponds to the `--force` option of the `svn diff` command.

Ignore properties

When selected, differences in the properties of items are ignored. This option corresponds to the `--ignore-properties` option of the `svn diff` command.

Properties only

When selected, only differences in the properties of the items are included in the patch file (file content is ignored). This option corresponds to the `--properties-only` option of the `svn diff` command.

Note: The **Ignore properties** and **Properties only** options are mutually exclusive.

Notice ancestry

If selected, the SVN common ancestry is taken into consideration when calculating the differences. This option corresponds to the `--notice-ancestry` option of the `svn diff` command.

Files Comparison Section

Ignore line endings

If selected, the differences in line endings are ignored when the patch is generated. This option corresponds to the `--ignore-eol-style` option of the `svn diff` command.

Ignore whitespaces

If selected, it allows you to specify how the whitespace changes should be handled. When selected, you can then choose between two options:

- **Ignore whitespace changes** (`--ignore-space-change`) - Ignores changes in the amount of whitespaces or to their type (for example, when changing the indentation or changing tabs to spaces).

  Note: Whitespaces that were added where there were none before, or that were removed, are still considered to be changes.
- **Ignore all whitespaces** (`--ignore-all-space`) - Ignores all types of whitespace changes.

Output Section

Save as

The patch will be created and saved in the specified file.

Use Git extended diff format

When selected, the patch is generated using the *Git* format. This option corresponds to the `--git` option of the `svn diff` command.

Working with Repositories

This section explains how to locate and browse SVN repositories in Syncro SVN Client.

Importing Resources Into a Repository

Importing resources into a repository is the process of copying local files and directories into a repository so that they can be managed by an Apache Subversion™ server. If you have already been using Subversion and you have an existing working copy you want to use, then you will likely want to follow the procedure for *using an existing working copy*.

The **Import folder** and **Import Files** actions are available from the **Import** submenu of the **Repository** menu or of the contextual menu in the **Repositories** view. These actions open a dialog box that allow you to select the directories or files that will be imported into the selected repository location.

The **Import folder** action opens the **Import folder** dialog box.

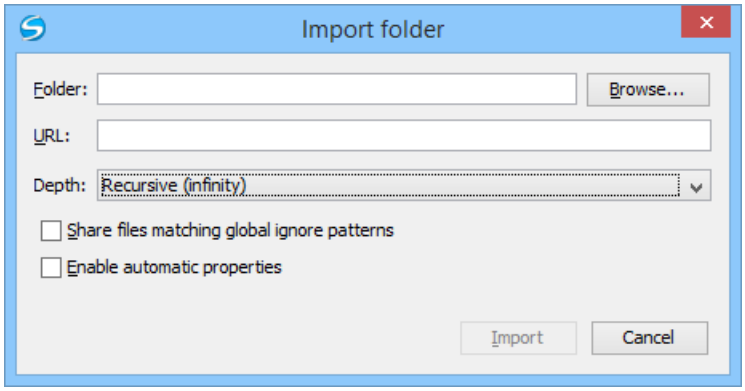


Figure 672: Import Folder Dialog Box

You can configure the following options:

Folder

Specify *the local folder* by using the text box or the **Browse** button. This folder should not be empty or already under version control.

Important: By default, the SVN system only imports the content of the specified folder, and not the folder itself. Therefore, it is recommended to use the **Browse** button to select the local folder so that the client will automatically append the name of it to the specified URL.

URL

Specify *the repository location* that will be used to access the folder to be imported.

Note: *Peg revisions* have no effect for this operation since it is used to send information to the repository.

Attention: If the new location already exists, make sure that it is an empty directory to avoid mixing your project content with other files (if items exist with the same name, an error will occur and the operation will not proceed). Otherwise, if the address does not exist, it is created automatically.

Depth

Recursive (infinity)

Imports all the files and folders contained in the selected folder.

Immediate children (immediates)

Imports only the child files and folders without recursing subfolders.

File children only (files)

Imports only the child files.

This folder only (empty)

Imports only the selected folder (no child file or folder is included).

Share files matching global ignore patterns

When selected, the file names that match the patterns defined in either of the following locations are also imported into the repository:

- The `global-ignores` property in *the SVN configuration file*.
- The ***File name ignore patterns*** *option* in the ***SVN*** *>* ***Working Copy*** *preferences page*.

Enable automatic properties/Disable automatic properties

Enables or disables automatic property assignment (per runtime configuration rules), overriding the `enable-auto-props` runtime configuration directive, defined in *the SVN configuration file*.

Note: This option is available only when there are defined properties to be applied automatically for newly added items under version control. You can define these properties in the SVN `config` file (in the `auto-props` section). Based on the value of the `enable-auto-props` runtime configuration directive, the presented option is either **Enable automatic properties**, or **Disable automatic properties**.

Exporting Resources From a Repository

This is the process of taking a resource from the repository and saving it locally in a clean form, with no version control information. This is very useful when you need a clean build for an installation kit.

The **Export** dialog box is similar to the **Check out** dialog box:

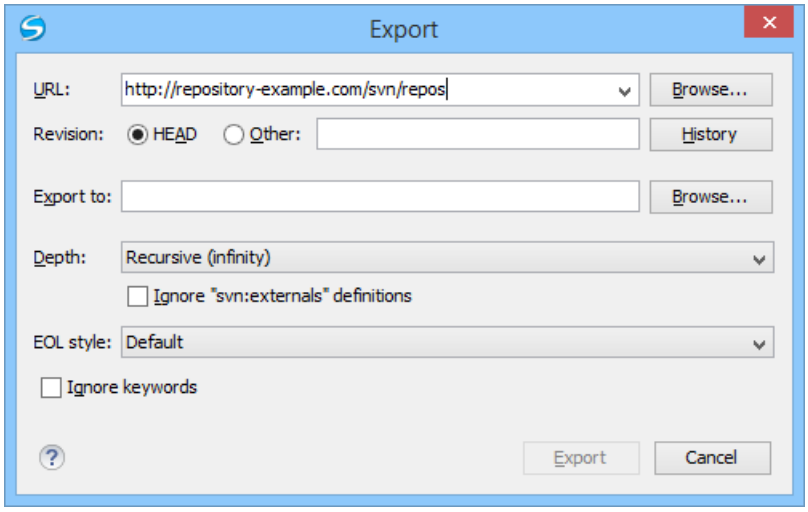


Figure 673: Export from Repository Dialog Box

You can configure the following options:

URL

Specify *the source directory from the repository* by using the text box or the **Browse** button.

Tip: To export an item that was deleted, moved, or replaced, you need to specify the original URL (before the item was removed) and use a *peg revision* at the end (for example, URL`@rev1234`).

Note: The content of the selected directory from the repository and not the directory itself will be exported to the file system.

Revision

You can choose between the **HEAD** or **Other** revision. If you need to export a specific revision, specify it in the **Other** text box or use the **History** button and choose a revision from the **History** dialog box.

Export to

Specify *the location where you want to export* the repository directory by typing the local path in the text box or by using the **Browse** button. If the specified local path does not point to an existing directory, it will automatically be created.

Important: By default, the SVN system only exports the content of the directory specified by the URL, and not the directory itself. Therefore, it is recommended to use the **Browse** button to select the *export* location so that the client will automatically append the name of the remote directory to the path of the selected directory.

Warning: The destination directory should be empty. If files exist, they will be overwritten by exported files with matching names.

Depth

Recursive (infinity)

Exports all the files and folders contained in the selected folder.

Immediate children (immediates)

Exports only the child files and folders without recursing subfolders.

File children only (files)

Exports only the child files.

This folder only (empty)

Exports only the selected folder (no child file or folder is included).

Ignore "svn:externals" definitions

When selected, external items are ignored in the export operation. This option is only available if you choose the **Recursive (infinity)** depth.

EOL style

Defines the *end-of-line (EOL)* marker that should be used when exporting files that have the value or the `svn:eol-style` property set to `native`. You can choose between the following styles:

- **Default** - It uses the system-specific *end-of-line* marker.
- **CRLF** - The Windows-specific *end-of-line* marker (*carriage return - line feed*).
- **LF** - The Unix / OS X-specific *end-of-line* marker (*line feed*).
- **CR** - The Mac OS 9 (or older)-specific *end-of-line* marker (*carriage return*).

Ignore keywords

When selected, the export operation does not expand the *SVN keywords* found inside the files.

Copy / Move / Delete Resources From a Repository

Once you have a location defined in the ***Repositories*** *view*, you can run commands (such as copy, move, and delete) directly on the repository. The commands correspond to the following actions in the contextual menu:

The **Copy to** and **Move to** action allows you to copy and move individual or multiple resources to a specific directory from the *HEAD* revision of the repository.

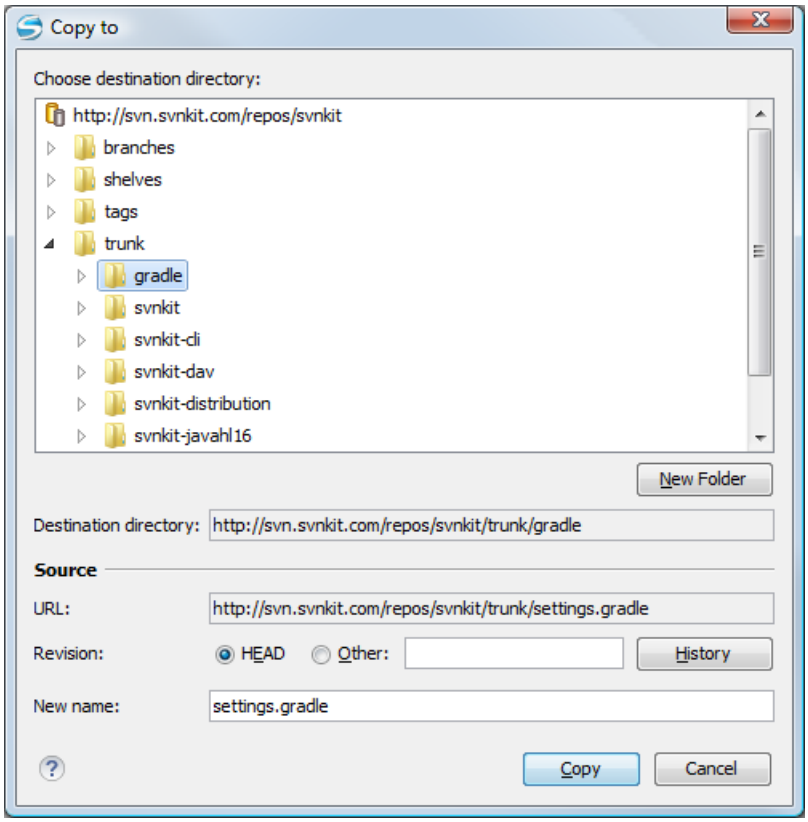


Figure 674: Copy/Move Items in Repository

The dialog box used to copy or move items allows you to browse the *HEAD* revision of the repository and select the destination of the items, presenting its repository URL below the tree view.

The **Source** section presents relevant options regarding the item(s) that you move or copy:

- **URL** - This field is displayed only if you copy/move a single item.
- **Revision** - Presents the revision from which you copy one or more items, allowing you to also choose another revision.

  Note: Since only items from the HEAD revision can be moved, the **Revision** options are not presented for the **Move to** action.

  Note: When you copy a single item while browsing a revision other than *HEAD*, the **Revision** options present this revision but does not allow you to change it. The same applies if copying multiple items.
- **New name** - This option is presented when you copy or move a single item, allowing you to also rename it.

Another useful action is **Delete**, allowing you to erase resources directly from the repository.

All three actions are commit operations and you will be prompted with the **Commit message** dialog box.

Sparse Checkout

Sometimes you only need to check out certain parts of a directory tree. In this case, you can check out the top directory (using the***Check out*** *action from the* ***Repositories*** *view*) and then recursively update only the needed directories (using the ***Update*** *action from the* ***Working Copy*** *view*). Each directory then has a depth set to it, with four possible values:

- **Recursive (infinity)** - Updates all descendant directories and files recursively.
- **Immediate children (immediates)** - Updates the directory, including direct child directories and files, but does not populate the child directories.
- **File children only (files)** - Updates the directory, including only child files without the child directories.
- **This folder only (empty)** - Updates only the selected directory, without updating any children.

For some operations, you can use as depth the current depth registered on the directories from the working copy (the value **Current depth**). This is the depth value defined in a previous check out or update operation.

The sparse checked out directories are presented in the ***Working Copy*** *view* with a marker corresponding to each depth value, in the top left corner, as follows:

- **Recursive (infinity)** - This is the default value and it is has no mark. The directory has no limiting depth.
- **Immediate children (immediates)** - The directory is limited to direct child directories (without contents) and files.
- **File children only (files)** - The directory is limited to direct child files only.
- **This folder only (empty)** - The directory has *empty* depth set.

A depth set on a directory means that some operations process only items within the specified depth range. For example, **Synchronize** on a working copy directory reports the repository modified items within the depth set on the directory and those existing in the working copy outside of this depth.

The depth information is also presented in the **SVN Information** dialog box and in the tool tip displayed when hovering a directory in the **Working Copy** view.

Syncro SVN Client Views

The main working area occupies the center of the application window, which contains the most important views:

- *Repositories View*
- *Working Copy View*
- *History View*
- *Console View*

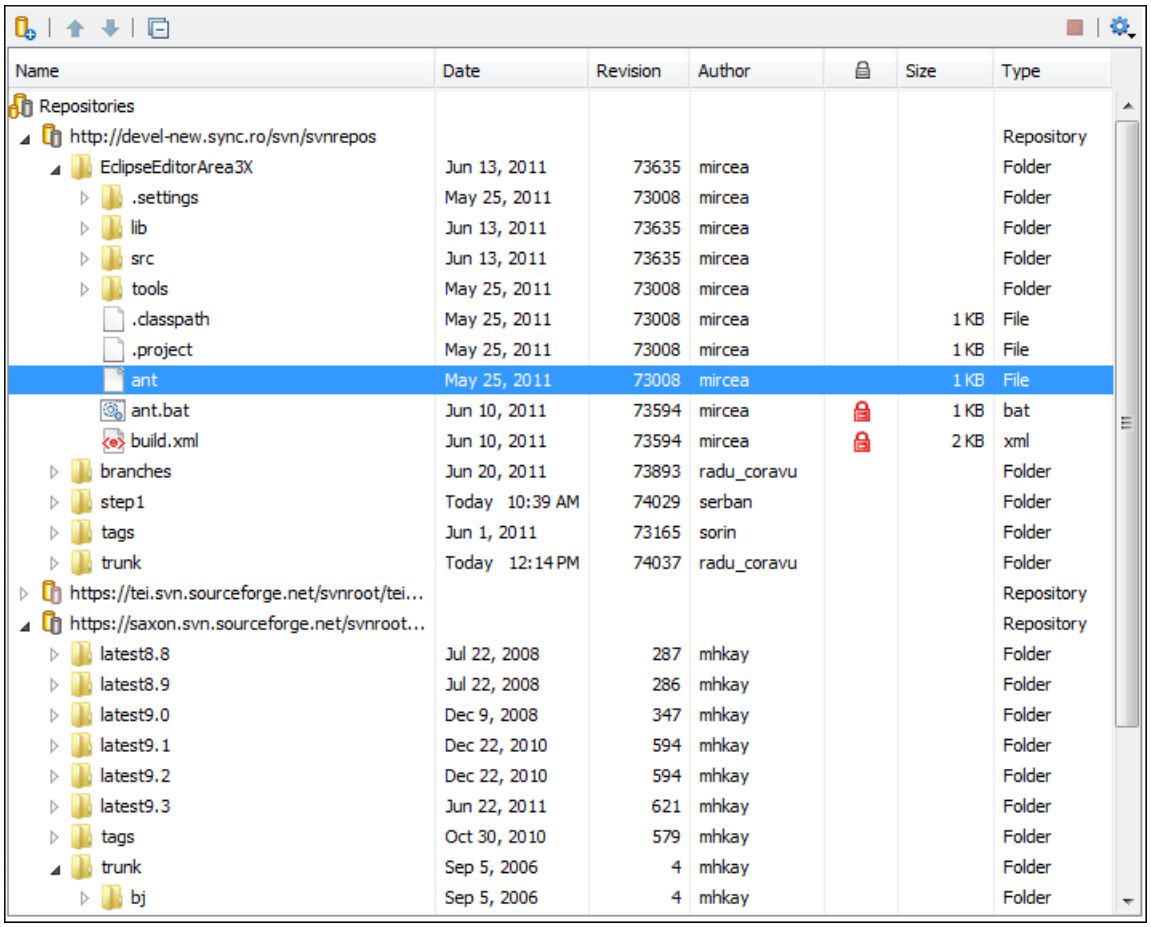
The other views that support the main working area are also presented in this section.

Repositories View

The **Repositories** view allows you to define and manage Apache Subversion™ repository locations and browse repositories. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

If no connections to your repository are available, you can *add a new repository location*. Repository files and folders are presented in a tree view with the repository locations at the first level, where each location represents a connection to a specific repository. More information about each resource is displayed in a tabular form:

- **Date** - Date when the resource was last modified.
- **Revision** - The revision number at the time the resource was last modified.
- **Author** - Name of the person who made the last modification on the resource.
- **Size** - Resource size on disk.
- **Lock information** - Information about the lock status of a file. When a repository file is locked by a user the icon is displayed in this column. If no icon is displayed the file is not locked. The tooltip of this column displays the details about the lock:
  - **Owner** - The name of the user who created the lock.
  - **Date** - The date when the user locked the file.
  - **Expires on** - Date when the lock expires. Lock expiry policy is set in the repository options, on the server side.
  - **Comment** - The message attached when the file was locked.
- **Type** - Contains the resource type or file extension.



Name	Date	Revision	Author	Size	Type
Repositories					
http://devel-new.sync.ro/svn/svnrepos					Repository
EclipseEditorArea3X	Jun 13, 2011	73635	mircea		Folder
.settings	May 25, 2011	73008	mircea		Folder
lib	Jun 13, 2011	73635	mircea		Folder
src	Jun 13, 2011	73635	mircea		Folder
tools	May 25, 2011	73008	mircea		Folder
.classpath	May 25, 2011	73008	mircea	1 KB	File
.project	May 25, 2011	73008	mircea	1 KB	File
ant	May 25, 2011	73008	mircea	1 KB	File
ant.bat	Jun 10, 2011	73594	mircea	1 KB	bat
build.xml	Jun 10, 2011	73594	mircea	2 KB	xml
branches	Jun 20, 2011	73893	radu_coravu		Folder
step1	Today 10:39 AM	74029	serban		Folder
tags	Jun 1, 2011	73165	sorin		Folder
trunk	Today 12:14 PM	74037	radu_coravu		Folder
https://tei.svn.sourceforge.net/svnroot/tei...					Repository
https://saxon.svn.sourceforge.net/svnroot...					Repository
latest8.8	Jul 22, 2008	287	mhkay		Folder
latest8.9	Jul 22, 2008	286	mhkay		Folder
latest9.0	Dec 9, 2008	347	mhkay		Folder
latest9.1	Dec 22, 2010	594	mhkay		Folder
latest9.2	Dec 22, 2010	594	mhkay		Folder
latest9.3	Jun 22, 2011	621	mhkay		Folder
tags	Oct 30, 2010	579	mhkay		Folder
trunk	Sep 5, 2006	4	mhkay		Folder
bj	Sep 5, 2006	4	mhkay		Folder

Figure 675: Repositories View

Toolbar

The **Repositories** view's toolbar contains the following buttons:

- **New Repository Location** - Allows you to enter a new repository location by means of the **Add SVN Repository** dialog box.
- **Move Up** - Move the selected repository up one position in the list of repositories in the **Repositories** view.
- **Move Down** - Move the selected repository down one position in the list of repositories in the **Repositories** view.
- **Collapse all** - Collapses all repository trees.
- **Stop** - Stops the current repository browsing operation executed when a repository node is expanded. This is useful when the operation takes too long or the server is not responding.
- **Settings** - Allows you to configure the resource table appearance.

Repositories View Contextual Menu Actions

The **Repositories** view contextual menu contains various actions, depending on the selected item. If a repository location is selected, the following management actions are available:

New Repository Location (Ctrl + Alt + N (Command + Alt + N on OS X))

Displays the **Add SVN Repository** dialog box. This dialog box allows you to define a new repository location.

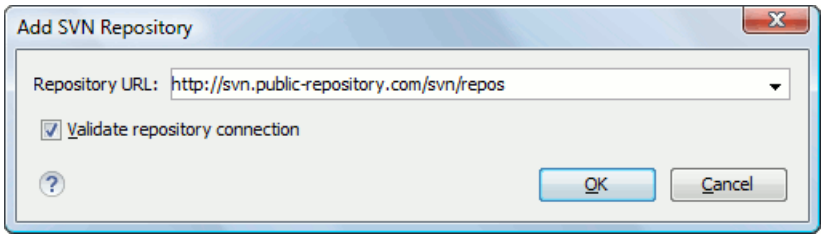


Figure 676: Add SVN Repository Dialog Box

If the **Validate repository connection** option is selected, the URL connection is validated before being added to the **Repositories** view.

Edit Repository Location (Ctrl + Alt + E (Command + Alt + E on OS X))

Context-dependent action that allows you to edit the selected repository location using the **Edit SVN Repository** dialog box. It is active only when a repository location root is selected.

Change the Revision to Browse (Ctrl + Alt + B (Command + Alt + B on OS X))

Context-dependent action that allows you to change the selected repository revision using the **Change the Revision to Browse** dialog box. It is active only when a repository location root is selected.

Remove Repository Location (Ctrl + Alt + R (Command + Alt + R on OS X))

Allows you to remove the selected repository location from the view. It shows you a confirmation dialog box before removal. It is active only when a repository location root is selected.

The following actions are common to all repository resources:

Open

Opens the selected file in the Editor view in read-only mode.

Open with

Displays the **Open with** dialog box to specify the editor in which the selected file is opened. If multiple files are selected, only external applications can be used to open the files.

Save as

Saves the selected files locally, as they are in the browsed revision.

Refresh (F5)

Refreshes the resource selected in the **Repositories** view.

Check out (Ctrl + Alt + O (Command + Alt + O on OS X))

Allows you to create a working copy from a repository directory, on your local file system. To read more about this operation, see the section *Check out a working copy*.

Branch/Tag

Allows you to create a branch or a tag from the selected folder in the repository. To read more about how to create a branch/tag, see the *Creation and management of Branches/Tags* section.

Share project

Allows you to *share a new project* using an SVN repository. The local project is automatically converted into an SVN working copy.

Import:

Import folder (Ctrl + Shift + L (Command + Shift + L on OS X))

Allows you to import the contents of a specified folder from the file system into the selected folder in a repository. To read more about this operation, see the section *Importing resources into a repository*.

Note: The difference between the **Import folder** and **Share project** actions is that the latter also converts the selected directory into a working copy.

Import Files (Ctrl + Shift + I (Command + Shift + I on OS X))

Imports the files selected from the files system into the selected folder in the repository.

Export

Opens *the **Export** dialog box* that allows you to configure options for exporting a folder from the repository to the local file system.

Show History (Ctrl + H (Command + T on OS X))

Displays the history of the selected resource. At the start of the operation, you can set filtering options.

Show Annotation (Ctrl + Shift + A (Command + Shift + A on OS X))

Opens the **Show Annotation** dialog box that computes *the annotations for a file and displays them in the **Annotations** view*, along with the history of the file in the **History** view.

Revision Graph (Ctrl + G (Command + G on OS X))

This action allows you to see the graphical representation of a resource history. For more details about a resource revision graph see *Revision Graph*. This operation is available for any resource selected in the **Repositories** view or **Working Copy** view.

Copy URL Location (Ctrl + Alt + U (Command + Alt + U on OS X))

Copies to clipboard the URL location of the selected resource.

Copy to

Copies to a specified location the currently selected resource(s). This action is also available when you browse other revisions than the latest one (*HEAD*), to allow restoring previous versions of an item.

Move to (Ctrl + M (Command + M on OS X))

Moves to a specified location the currently selected resource(s).

Rename ((F2))

Renames the selected resource.

Delete ((Delete))

Deletes selected items from the repository via an immediate commit.

New Folder (Ctrl + Shift + F (Command + Shift + F on OS X))

Allows you to create a folder in the selected repository path (available only for folders).

Locking

The following options are available only for files:

Lock (Ctrl + K (Command + K on OS X))

Allows you to lock certain files for which you need exclusive access. For more details on the use of this action, see *Locking a file*.

Unlock (Ctrl + Shift + K (Command + Shift + K on OS X))

Releases the exclusive access to a file from the repository. You can also choose to unlock it by force (*break the lock*).

Show SVN Properties (Ctrl + Shift + P (Command + Shift + P on OS X))

Brings up the *Properties view* displaying the SVN properties for the selected resource. This view does not allow adding, editing, or removing SVN properties of a repository resource. These operations are allowed only for working copy resources.

Show SVN Information (Ctrl + I (Command + I on OS X))

Provides additional information for the selected resource. For more details, go to *Obtain information for a resource*.

Assistant Actions

When there is no repository configured, the **Repositories** view mode lists the following two actions:

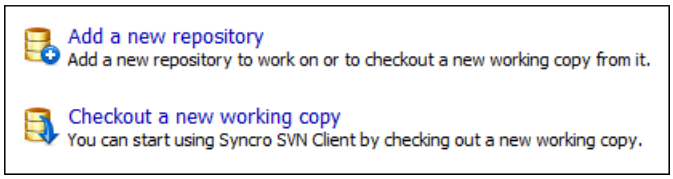


Figure 677: Repositories View Actions

Drag and Drop Operations

The structure of the files tree can be changed with drag and drop operations inside the **Repositories** view. These operations behave in the same way with the **Copy to**/**Move to** operations.

Working Copy View

The **Working Copy** view allows you to manage the content of an SVN working copy. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

The toolbar contains the following:

- The list of defined working copies.
- A set of view modes that allow you to filter the content of the working copy based on the resource status (such as incoming or outgoing changes).
- **Settings** menu.

If you click any of the view modes (**All Files**, **Modified**, **Incoming**, **Outgoing**, **Conflicts**), the information displayed changes as follows:

- **All Files** - Resources (files and folders) are presented in a hierarchical structure with the root of the tree representing the location of the working copy on the file system. Each resource has an icon representation that describes the type of resource and also depicts the state of that resource with a small overlay icon.

svnkit | All Files | Modified | Incoming | Outgoing | Conflicts

Name	Date	Revision	Author	Size	Type
E:\svnkit	May 15, 2011	7636	alex		File Folder
gradle	May 4, 2011	7623	alex		File Folder
wrapper	May 4, 2011	7623	alex		File Folder
gradle-wrapper.jar	May 4, 2011	7618	alex	12 KB	Executable ...
gradle-wrapper.properties	May 4, 2011	7623	alex	1 KB	PROPERTIE...
svnkit	May 15, 2011	7636	alex		File Folder
svnkit-cli	May 10, 2011	7630	alex		File Folder
.settings	May 4, 2011	7618	alex		File Folder
src	May 10, 2011	7630	alex		File Folder
main	May 10, 2011	7630	alex		File Folder
conf	May 4, 2011	7618	alex		File Folder
java	May 4, 2011	7622	alex		File Folder
resources	May 4, 2011	7618	alex		File Folder
scripts	May 10, 2011	7630	alex		File Folder
jsvn	May 4, 2011	7618	alex	2 KB	File
jsvn.bat	May 10, 2011	7630	alex	2 KB	Windows B...
jsvnsetup.openvms	May 4, 2011	7618	alex	1 KB	OPENVMS File
build.gradle	May 4, 2011	7618	alex	2 KB	GRADLE File
svnkit-dav	May 4, 2011	7620	alex		File Folder
svnkit-distribution	May 4, 2011	7623	alex		File Folder
svnkit-javahl16	May 4, 2011	7618	alex		File Folder
svnkit-osgi	May 4, 2011	7623	alex		File Folder
svnkit-test	May 12, 2011	7635	alex		File Folder
.settings	May 4, 2011	7618	alex		File Folder
configurations	May 4, 2011	7618	alex		File Folder

Figure 678: Working Copy View - All Files View Mode

- **Modified** - The resource tree presents resources modified locally (including those with conflicting content) and remotely. Decorator icons are used to differentiate between various resource states:
  - Incoming modification from repository:
    - - File content or properties modified remotely.
    - - New file added remotely.
    - - File deleted remotely.
  - Outgoing modification to repository:
    - - File content or properties modified locally.
    - - New file added locally.
    - - File deleted locally.
  - - **Pseudo-conflict state** - A resource being locally and remotely modified at the same time, or a parent directory of such a resource.
  - - **Real conflict state** - A resource that had both incoming and outgoing changes and not all the differences could be merged automatically through the update operation (manually editing the local file is necessary for resolving the conflict).

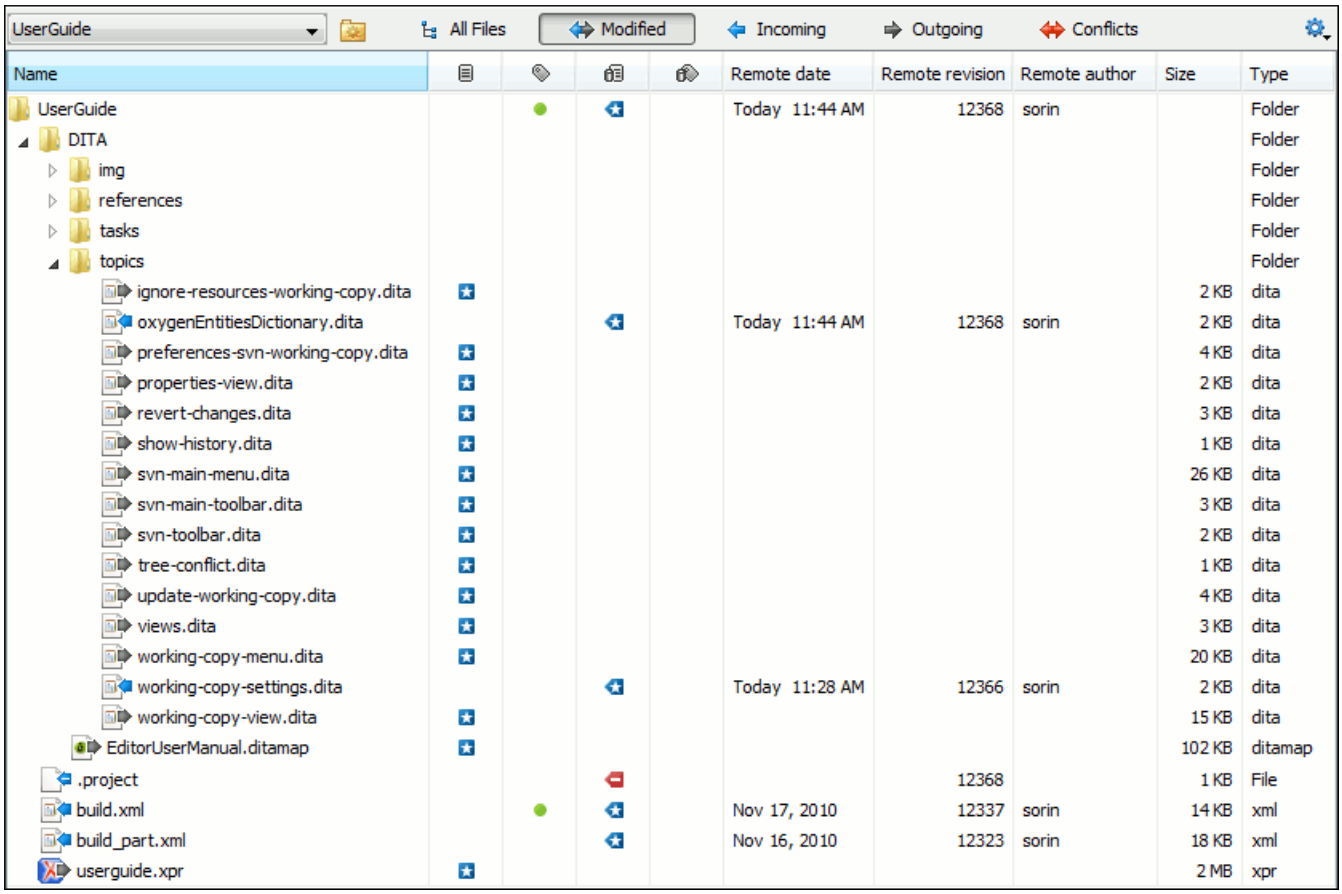


Figure 679: Working Copy View - Modified View Mode

- **Incoming** - The resource tree presents only incoming changes.
- **Outgoing** - The resource tree presents only outgoing changes.
- **Conflicts** - The resource tree presents only conflicting changes (real conflicts and pseudo-conflicts).

The following columns provide information about the resources:

- **Name** - Resource name. Resource icons can have the following decorator icons:
  - Additional status information:
    - **Propagated modification marker** - A folder marked with this icon indicates that the folder itself presents some changes (such as modified properties) or a child resource has been modified.
    - **External** - This indicates a mapping of a local directory to the URL of a versioned resource. It is declared with a `svn:externals` property in the parent folder and it indicates a working copy not directly related with the parent working copy that defines it.
    - **Switched** - This indicates a resource that has been switched from the initial repository location to a new location within the same repository. The resource goes to this state as a result of *the Switch action* executed from the contextual menu of the Working Copy view.
    - **Grayed** - A resource with a grayed-out icon, but no overlaid icon, is an ignored resource. It is obtained with the **Add to svn:ignore** action.
  - Current SVN depth of a folder:
    - **Immediate children (immediates)** (a variant of *sparse checkout*) - The directory contains only direct file and folder children. Child folders ignore their content.
    - **File children only (files)** (a variant of *sparse checkout*) - The directory contains only direct file children, disregarding any child folders.

- **This folder only (empty)** (a variant of *sparse checkout*) - The directory discards any child resource.

Note:

- Any folder not marked with one of the depth icons, has recursive depth (*infinity*) set by default (presents all levels of child resources).
- Although folders not under version control can have no depth set, Oxygen XML Editor presents *unversioned* and *ignored* folders with *empty* depth when **Show unversioned directories content** or **Show ignored directories content** options are not selected.

- **Local file status** - Shows the changes of working copy resources that were not committed to the repository yet. The following icons are used to mark resource status:
  - - Resource is *not under version control* (*unversioned*).
  - - Resource is being *ignored* because it is not under version control and its name matches a file name pattern defined in one of the following places:
    - *global-ignores* section in the SVN client-side *`config` file*.

      Attention: If you do not explicitly set the `global-ignores` runtime configuration option (either to your preferred set of patterns or to an empty string), Subversion uses the default value.
    - ***Application global ignores** option* of Oxygen XML Editor.
    - The value of a *`svn:ignore` property* set on the parent folder of the resource being ignored.
  - - Marks a newly created resource, *scheduled for addition* to the version control system.
  - - Marks a resource *scheduled for addition*, created by copying a resource already under version control and inheriting all its SVN history.
  - - The content of the resource has been *modified*.
  - - Resource has been *replaced* in your working copy (the file was scheduled for deletion, and then a new file with the same name was scheduled for addition in its place).
  - - Resource is *deleted* (scheduled for deletion from **Repository** upon the next commit).
  - - The resource is *incomplete* (as a result of an interrupted *check out* or *update* operation).
  - - The resource is *missing* because it was moved or deleted without using an SVN-aware application.
  - - The contents of the resource is in *real conflict state*.
  - - Resource is in a *name conflict* state.
  - - Resource is in *tree conflict* state after an update operation because:
    - Resource was locally modified and incoming deleted from repository.
    - Resource was locally scheduled for deletion and incoming modified.
  - - Resource is *obstructed* (versioned as one kind of object: file, directory, or symbolic link, but has been replaced outside Syncro SVN Client by a different kind of object).
- **Local properties status** - Marks the resources that have SVN properties, with the following possible states:
  - - The resource has SVN properties set.
  - - The resource properties have been modified.
  - - Properties for this resource are in *real conflict* with property updates received from the repository.
- **Revision** - The current revision number of the resource.
- **Date** - Date when the resource was last time modified on the disk.
- **BASE Revision** - The revision number of the pristine version of the resource.
- **BASE Date** - Date when the pristine version of the resource was last time committed in the repository.
- **Author** - Name of the person who made the last modification on the pristine version of the resource.
- **Remote file status** - Shows changes of resources recently modified in the repository. The following icons are used to mark incoming resource status:
  - - Resource is newly added in repository.
  - - The content of the resource has been modified in repository.
  - - Resource was replaced in repository.

- - Resource was deleted from repository.
- **Remote properties status** - Resources marked with the icon have incoming modified properties from the repository.
- **Remote revision** - Revision number of the resource latest committed modification.
- **Remote date** - Date of the resource latest modification committed on the repository.
- **Remote author** - Name of the author who committed the latest modification on the repository.
- **Lock information** - Shows the lock state of a resource. The lock mechanism is a convention intended to help you signal other users that you are working with a particular set of files. It minimizes the time and effort wasted in solving possible conflicts generated by clashing commits. A lock gives you exclusive rights over a file, only if other users follow this convention and they do not try to bypass the lock state of a file.

  A folder can be locked only by the SVN client application, completely transparent to the user, if an operation in progress was interrupted unexpectedly. As a result, folders affected by the operation are marked with the symbol. To clear the locked state of a folder, use the **Clean up** action.

  Note: Users can lock only files.

  The following lock states are displayed:

  - *no lock* - the file is not locked. This is the default state of a file in the SVN repository.
  - *remotely locked* ( ) - shown when:
    - Another user has locked the file in the repository.
    - The file was locked by the same user from another working copy.
    - The file was locked from the **Repositories** view.

    If you try to commit a new revision of the file to the repository, the server does not allow you to bypass the file lock.

    Note: To commit a new revision, you need to wait for the file to be unlocked. Ultimately, you might try to *break* or *steal* the lock, but this is not what other users expect. Use these actions carefully, especially when you are not the file lock owner.
  - *locked* ( ) - displayed after you have locked a file from the current working copy. Now you have exclusive rights over the corresponding file, being the only one who can commit changes to the file in the repository.

    Note: Working copies keep track of their locked files, so the locks are presented between different sessions of the application. Synchronize your working copy with the repository to make sure that the locks are still valid (not *stolen* or *broken*).
  - *stolen* ( ) - a file already locked from your working copy is being locked by another user. Now the owner of the file lock is the user who stole the lock from you.
  - *broken* ( ) - a file already locked from your working copy is no longer locked in the repository (it was unlocked by another user).

    Note: To remove the *stolen* or *broken* states from your working copy files, you have to **Update** them.

  If one of your working copy files is locked, hover the mouse pointer over the lock icon to see more information:

  - Lock type - current file lock state
  - Owner - the name of the user who created the lock
  - Date - the date when the user locked the file
  - Expires on - date when the lock expires. Lock expiry policy is set in the repository options, on the server side
  - Comment - the message attached when the file was locked
- **Size** - Resource size on disk
- **Type** - Contains the resource type or file extension

Note: The working copy table allows you to show or hide any of its columns and also to sort its contents by any of the displayed columns. The table header provides a contextual menu that allows you to customize the displayed information.

The toolbar contains the following options for switching to a different working copy:

- **List of Defined Working Copies** - A drop-down menu that contains all the working copies Oxygen XML Editor is aware of. When you select a different working copy from the list, the newly selected working copy content is scanned and displayed in the **Working Copy** view.
- (on Mac OS X) **Working Copies Manager** - Opens a dialog box that displays the working copies Oxygen XML Editor is aware of. In this dialog box, you can add existing working copies or remove those you no longer need. If you try to add a folder that is not a valid Subversion working copy, Oxygen XML Editor warns you that the selected directory is not under version control.

  Note: Removing a working copy from this dialog box does NOT remove it from your file system; you will have to do that manually.

Working Copy Settings

The **Settings** button from the toolbar of the **Working Copy** view provides the following options:

- **Show unversioned directories content** - Displays the content of unversioned directories.

  Note: If this option is not selected, it will be ignored for items that, after a synchronize, are reported as incoming from the repository. This applies for all working copy modes, except **All Files**.
- **Show ignored items** - Displays the ignored resource when **All Files** mode is selected.
- **Show ignored directories content** - Displays the content of ignored directories when **All Files** mode is selected.

  Note: Although *ignored* items are not presented in the **Modified**, **Incoming**, and **Conflicts** modes, they will be if, after a synchronize, they are reported as incoming from the repository.
- **Show deleted items** - Displays the deleted resource when **All Files** mode is selected. All other modes always display deleted resources, disregarding this option.
- **Tree** / **Compressed** / **Flat** - Affect the way information is displayed inside the **Modified**, **Incoming**, **Outgoing**, and **Conflicts** view modes.
- **Configure columns** - Allows you to customize the structure of the **Working Copy** view data. This action opens the following dialog box:

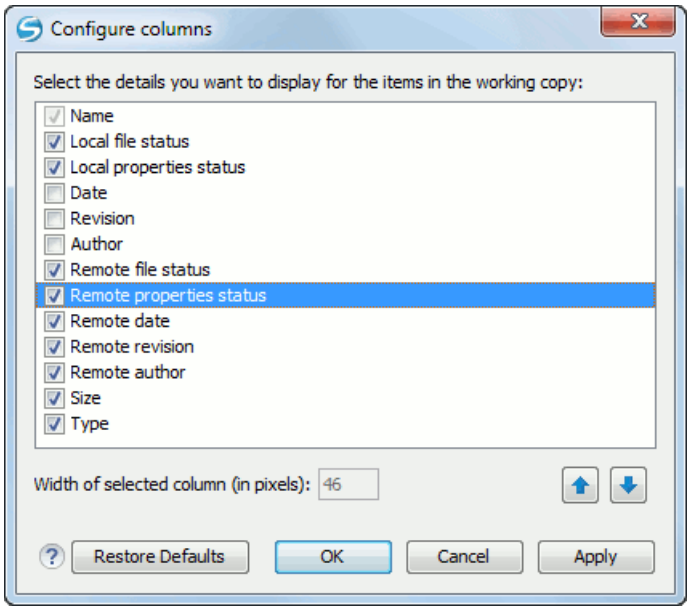


Figure 680: Configure Columns of Working Copy View

The order of the columns can be changed with the two arrow buttons. The column size can be edited in the **Width of selected column** field. The **Restore Defaults** button reverts all columns to the default order, width, and enabled/disabled state from the installation of the application.

Working Copy Format

When an SVN working copy is loaded, Syncro SVN Client first checks the format of the working copy:

- If the format is older than SVN 1.7, you are prompted to upgrade it to SVN 1.8 to load it.
- If the format is 1.7, Syncro SVN Client takes into account the state of the ***When loading an old format working copy****option*.

To change how working copy formats are handled, *open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)***, go to **SVN** > **Working copy**, and configure the options in the ***Administrative area*** section.

Note:

- The format of the working copy can be downgraded or upgraded at any time with the **Upgrade** and **Downgrade** actions available in the **Tools** menu. These actions allow switching between SVN 1.7 and SVN 1.8 working copy formats.
- SVN 1.7 working copies cannot be downgraded to older formats.

Refresh a Working Copy

A refresh is a frequent operation triggered automatically when you switch between two working copies using the toolbar selector of the **Working Copy** view and when you switch between Oxygen XML Editor and other applications.

The **Working Copy** view features a fast refresh mechanism: the content is cached locally when loading the working copy for the first time. Later on, when the same working copy is displayed again, the application uses this cache to detect the changes between the cached content and the current content found on disk. The refresh operation is run on these changes only, thus improving the response time. improvement is noticeable especially when working with large working copies.

Working Copy View Contextual Menu Actions

The contextual menu in the **Working Copy** view contains the following actions:

Edit conflict (Ctrl (Command on OS X) + E)

Opens the **Compare** editor, allowing you to modify the content of the currently conflicting resources. For more information about editing conflicts, see *Edit conflicts*.

Open in Compare Editor (Ctrl (Command on OS X) + Alt + C)

Displays changes made in the currently selected file.

Open (Ctrl (Command on OS X) + O)

Opens the selected resource from the working copy. Files are opened with an internal editor or an external application associated with that file type, while folders are opened with the default file system browsing application (Windows Explorer on Windows, Finder on OS X, etc).

Open with...

Submenu that allows you to open the selected resource either with Oxygen XML Editor or with another application.

Show in Explorer/Show in Finder

Opens the parent directory of the selected working copy file and selects the file.

Expand All (Ctrl (Command on OS X) + Alt + X)

Displays all descendants of the selected folder. The same behavior is obtained by double-clicking a collapsed folder.

Refresh (F5)

Re-scans the selected resources recursively and refreshes their status in the working copy view.

Synchronize (Ctrl (Command on OS X) + Shift + S)

Connects to the repository and determines the working copy and repository changes made to the selected resources. The application switches to **Modified** view mode if the ***Always switch to 'Modified' mode*** *option* is selected.

Update (Ctrl (Command on OS X)+ U)

Updates the selected resources to the *HEAD* revision (latest modifications) from the repository. If the selection contains a directory, it will be updated depending on its depth.

Update to revision/depth

Allows you to update the selected resources from the working copy to an earlier revision from the repository. You can also select the update *depth* for the current folder. You can find out more about the *depth* term in the *sparse checkouts* section.

Commit

Collects the outgoing changes from the selected resources in the working copy and allows you to choose exactly what resources to commit. A directory will always be committed recursively. Unversioned resources will be deselected by default. In the **Commit** dialog box you can also enter a comment before sending your changes to the repository.

Revert (Ctrl (Command on OS X) + Shift + V)

Undoes all local changes for the selected resources. It does not contact the repository and the files are obtained from Apache Subversion™ pristine copy. It is available only for modified resources. See *Revert your changes* for more information.

Override and Update

Drops any outgoing change and replaces the local resource with the HEAD revision. This action is available on resources with outgoing changes, including conflicting ones. See the *Revert your changes* section.

Override and Commit

Drops any incoming changes and sends your local version of the resource to the repository. This action is available on conflicting resources. For more information see *Drop incoming modifications*.

Mark Resolved (Ctrl (Command on OS X) + Shift + R)

Instructs the Subversion system that you resolved a conflicting resource. For more information, see *Merge conflicts*.

Mark as Merged (Ctrl (Command on OS X) + Shift + M)

Instructs the Subversion system that you resolved the pseudo-conflict by merging the changes and you want to commit the resource. Read the *Merge conflicts* section for more information about how you can solve the pseudo-conflicts.

Create patch (Ctrl (Command on OS X) + Alt + P)

Allows you to create a file containing all the differences between two resources, based on the `svn diff` command. To read more about creating patches, see *the section about patches*.

Compare with:

- **Latest from HEAD (Ctrl (Command on OS X) + Alt + H)** - Performs a 3-way diff operation between the selected file and the *HEAD* revision from the repository and displays the result in the **Compare view**. The common ancestor of the 3-way diff operation is the *BASE* version of the file from the local working copy.
- **BASE revision (Ctrl (Command on OS X) + Alt + C)** - Compares the working copy file with the BASE revision file (the so-called *pristine copy*).
- **Revision (Ctrl (Command on OS X) + Alt + R)** - Displays the **History view** that contains the log history of that resource.
- **Branch/Tag** - Opens the **Compare with Branch/Tag** dialog box that allows you to specify *another file from the repository* (**To URL** field) to compare with the working copy file. You can specify the revision of the repository file by choosing between **HEAD revision** or specific **Other revision**.

  Tip: To compare with a file that was deleted, moved, or replaced, you need to specify the original URL (before the file was removed) and use a *peg revision* at the end (for example, URL`@rev1234`).
- **Each other** - Compares two selected files with each other.

These *compare* actions are available only if the selected resource is a file.

Replace with:

- **Latest from HEAD** - Replaces the selected resources with their versions from the *HEAD* revision of the repository.
- **BASE revision** - Replace the selected resources with their versions from the pristine copy (the BASE revision).

Note: In some cases it is impossible to replace the currently selected resources with their versions from the *BASE/HEAD* revision:

- For the **Replace with BASE revision** action, the resources being unversioned or added have no *BASE* revision, and thus cannot be replaced. However, they will be deleted if the action is invoked on a parent folder. The action will never work for missing folders or for obstructing files (folders being obstructed by a file), since you cannot recover a tree of folders
- For the **Replace with latest from HEAD** action, you must be aware that there are cases when resources will be completely deleted or reverted to the BASE revision and then updated to a HEAD revision to avoid conflicts. These cases are:
  - The resource is *unversioned*, *added*, *obstructed*, or *modified*.
  - The resource is affected by a `svn:ignore` or `svn:externals` property that is locally added on the parent folder and not yet committed to the repository.

Show History (Ctrl (Command on OS X) + H)

Displays the **History view** where the log history for the selected resource will be presented. For more details about resource history, see the sections about *the resource history view* and *requesting the history for a resource*.

Show Annotation (Ctrl + Shift + A (Command + Shift + A on OS X))

Opens the **Show Annotation** dialog box that computes *the annotations for a file and displays them in the **Annotations** view*, along with the history of the file in the **History** view.

Revision Graph (Ctrl (Command on OS X) + G)

This action allows you to see the graphical representation history of a resource. For more details about the revision graph of resources, see *Revision Graph*.

Copy URL Location (Ctrl (Command on OS X) + Alt + U)

Copies the encoded URL of the selected resource from the Working Copy to the clipboard.

Mark as copied

You can use this action to mark an item from the working copy as a copy of an other item under *version control*, when the copy operation was performed outside of an SVN client. The **Mark as copied** action is available when you select two items (both the new item and source item), and it depends on the state of the source item.

Mark as moved

You can use this action to mark an item from the working copy as being moved from another location of the working copy, when the move operation was performed outside of an SVN client. The **Mark as moved** action is available when you select two items from different locations (both the new item and the source item that is usually reported as *missing*), and it depends on the state of the source item.

Mark as renamed

You can use this action to mark an item from the working copy as being renamed outside of an SVN client. The **Mark as renamed** action is available when you select two items from the same directory (both the new item and the source item that is usually reported as *missing*), and it depends on the state of the source item.

Copy to

Copies the currently selected resource to a specified location.

Move to Ctrl + M (Command + M on OS X)

Moves the currently selected resource to a specified location.

Rename (F2)

As with the move command, a copy of the original resource will be made with the new name and the original will be marked as deleted. Note that you can only rename one resource at a time.

Delete (Delete)

Schedules selected items for deletion upon the next commit and removes them from the disk. Depending on the state of each item, you are prompted to confirm the operation.

New:

- **New File** - Creates a new file inside the selected folder. The newly created file will be added under version control only if the parent folder is already versioned.
- **New Folder (Ctrl (Command on OS X)+ Shift + F)** - Creates a child folder inside the selected folder. The newly created folder will be added under version control only if its parent is already versioned.
- **New External Folder (Ctrl (Command on OS X) + Shift + W)** - This operation allows you to add a new external definition on the selected folder. An external definition is a mapping of a local directory to *a URL of a versioned directory*, and ideally a particular revision, stored in the `svn:externals` property of the selected folder.

  Tip: You can specify a particular revision of the external item by using a *peg revision* at the end of the URL (for example, URL`@rev1234`). You can also use peg revisions to access external items that were deleted, moved, or replaced.

  The URL used in the external definition format can be relative. You can specify the repository URL that the external folder points to by using one of the following relative formats:

  - **../** - Relative to the URL of the directory that the `svn:externals` property is set.
  - **^/** - Relative to the root of the repository in which the `svn:externals` property is versioned.
  - **//** - Relative to the scheme of the URL of the directory that the `svn:externals` property is set.
  - **/** - Relative to the root URL of the server in which the `svn:externals` property is versioned.

  Important: To change the target URL of an external definition, or to delete an external item, do the following:

  1. Modify or delete the item definition found in the `svn:externals` property that is set on the parent folder.
  2. For the change to take effect, use the **Update** operation on the parent folder of the external item.

  Note: Syncro SVN Client does not support definitions of local relative external items.

Add to "svn:ignore" (Ctrl (Command on OS X) + Alt + I)

Allows you to add files that should not participate in the *version control* operations inside your working copy . This action can only be performed on resources not under *version control*. It actually modifies the value of the `svn:ignore` property in the parent directory of the resource. Read more about this in the *Ignore Resources Not Under Version Control* section.

Add to version control (Ctrl (Command on OS X) + Alt + V)

Allows you to add resources that are not under *version control*. For further details, see *Add Resources to Version Control* section.

Remove from version control

Schedules selected items for deletion from repository upon the next commit. The items are not removed from the file system after committing.

Clean up (Ctrl (Command on OS X) + Shift + C)

Performs a maintenance cleanup operation on the selected resources from the working copy. This operation removes the Subversion maintenance locks that were left behind. This is useful when you already know where the problem originated and want to fix it as quickly as possible. It is only active for resources under *version control*.

Locking:

- **Scan for locks (Ctrl (Command on OS X) + L)** - Contacts the repository and recursively obtains the list of locks for the selected resources. A dialog box containing the locked files and the lock description will be displayed. This is only active for resources under *version control*. For more details see *Scanning for locks*.
- **Lock (Ctrl (Command on OS X) + K)** - Allows you to lock certain files that need exclusive access. You can write a comment describing the reason for the lock and you can also force (*steal*) the lock. This action is active only on files under *version control*. For more details on the use of this action see *Locking a file*.

- **Unlock (Ctrl (Command on OS X) + Alt + K)** - Releases the exclusive access to a file from the repository. You can also choose to unlock it by force (*break the lock*).

Show SVN Properties (Ctrl + P (Command + P on OS X))

Brings up the *Properties view* and displays the SVN properties for the selected resource.

Show SVN Information (Ctrl + I (Command + I on OS X))

Provides additional information for the selected resource from the working copy. For more details, go to *Obtain information for a resource*.

Drag and Drop Operations

The structure of the files tree can be changed with drag and drop operations inside the **Working Copy** view. These operations behave in the same way with the **Copy to**/**Move to** operations.

Also, files and folders can be added to the file tree of the view as *unversioned* resources by drag and drop operations from other applications (for example, from Windows Explorer or Mac OS X Finder). In this case, the items from the file system are only copied, without removing them from their original location.

Attention: When you drag items from the working copy to a different application, the performed operation is controlled by that application. This means that the moved items are left as *missing* in the working copy (items are moved in the file system only, but no SVN versioning meta-data is changed).

Assistant Actions

To ensure a continuous and productive work flow, when a view mode has no files to present, it offers a set of guiding actions with some possible paths to follow.

Initially, when there is no working copy configured the **All Files** view mode lists the following two actions:

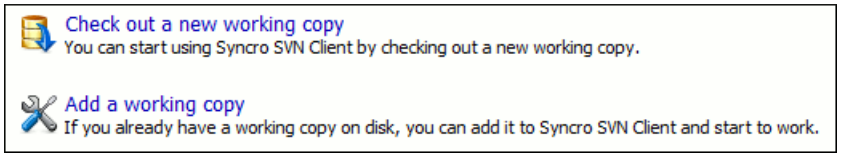


Figure 681: All Files Panel

For **Modified**, **Incoming**, **Outgoing**, **Conflicts** view modes, the following actions may be available, depending on the current working copy state in various contexts:

- **Synchronize with Repository** - Available only when there is nothing to present in the **Modified** and **Incoming** view modes.
- **Switch to Incoming** - Selects the **Incoming** view mode.
- **Switch to Outgoing** - Selects the **Outgoing** view mode.
- **Switch to Conflicts** - Selects the **Conflicts** view mode.
- **Show all changes/incoming/outgoing/conflicts** - Depending on the currently selected view mode, this action presents the corresponding resources after a synchronize operation was executed only on a part of the working copy resources.
- **(Information message)** - Informs you why there are no resources presented in the currently selected view mode.

History View

In Apache Subversion™, both files and directories are versioned and have a history. If you want to examine the history for a selected resource and find out what happened at a certain revision you can use the **History view** that can be accessed from *Repositories view*, *Working Copy view*, *Revision Graph*, or *Directory Change Set view*. From

the **Working copy view** you can display the history of local versioned resources. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

The view consists of four distinct areas:

- The table showing details about each revision, such as revision number, commit date and time, number of changes (more details available in the tooltip), author's name, and a fragment of the commit message.

  Some revisions may be highlighted to emphasize:

  - The current revision of the resource for which the history is displayed - a bold font revision.
  - The last revision in which the content or properties of the resource were modified - blue font revision.

  Note: Both font highlights may be applied for the same revision.
- The complete commit message for the selected revision.
- A tree structure showing the folders where the modified resources are located. You can compress this structure to a more compact form that focuses on the folders that contain the actual modifications.
- The list of resources modified in the selected revision. For each resource, the type of action done against it is marked with one of the following symbols:

  - - A newly created resource.
  - - A newly created resource, copied from another repository location.
  - - The content/properties of the resource were *modified*.
  - - Resource was *replaced* in the repository.
  - - Resource was deleted from the repository.

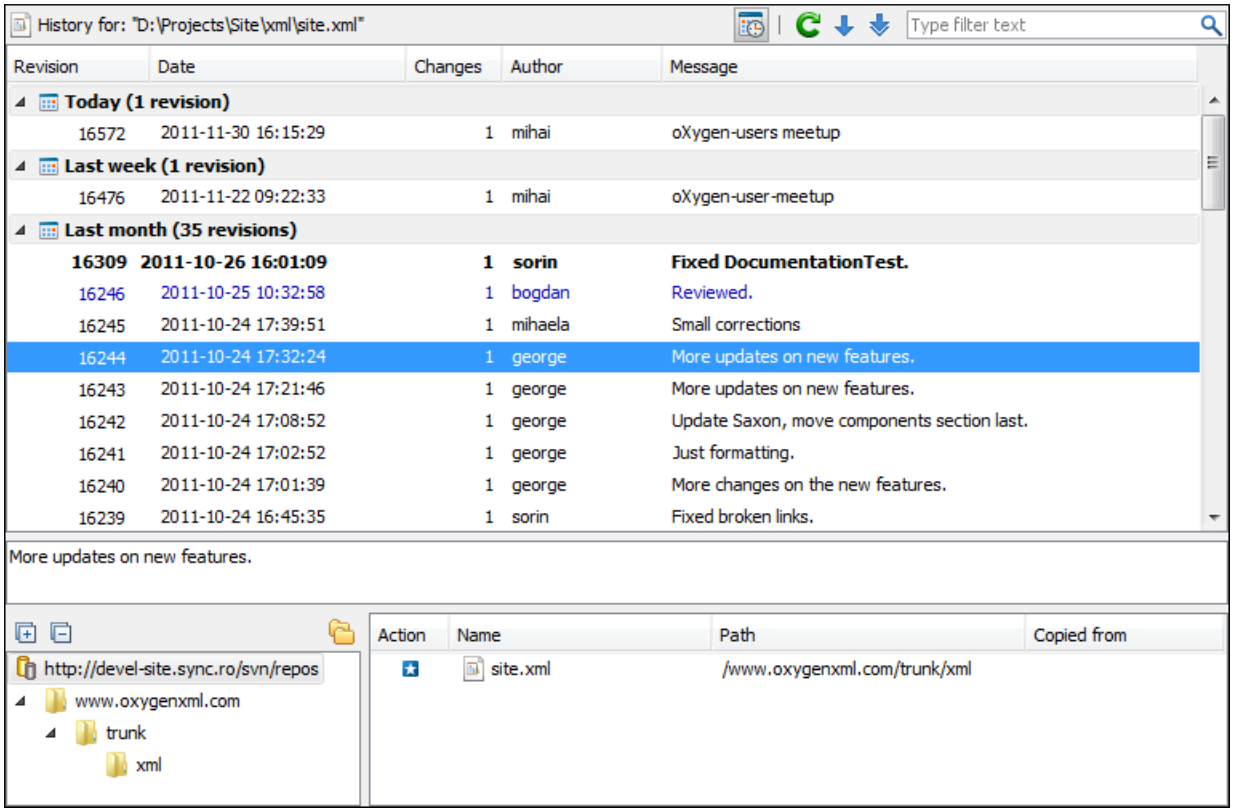


Figure 682: History View

You can group revisions in predefined time frames (today, yesterday, this week, this month), by pressing the **Group by date** button from the toolbar.

History Filter Dialog Box

The **History view** does not always show all the changes ever made to a resource because there may be thousands of changes and retrieving the entire list can take a long time. Normally you are interested in the more

recent ones. That is why you can specify the criteria for the revisions displayed in the **History view** by selecting one of several options presented in the **History** dialog box that is displayed when you invoke the **Show History** action.

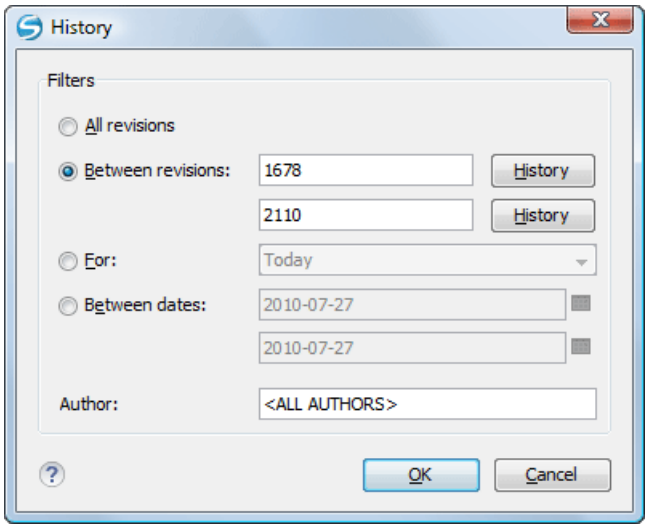


Figure 683: History Filters Dialog Box

Options for the set of revisions presented in the History view are:

- All revisions of the selected resource.
- Only revisions between a start revision number and an end revision number.
- Only revisions added in a period of time (such as today, last week, last month, etc.)
- Only revisions between a start and an end date.
- Only revisions committed by a specified SVN user.

The toolbar of the **History view** has two buttons for extending the set of revisions presented in the view: **Get next 50** and **Get all**.

History Filter Field

When only the history entries that contain a specified substring need to be displayed in the **History** view, the filter field displayed at the top of this view is a useful tool. Just enter the search string in the field next to the **Find** label. Only the items (with an author name, commit message, revision number, or date) that match the search string are kept in the **History** view. When you press the **Search** button, the filter action is executed and the content of the table is updated.

History View Contextual Menu Actions

The **History** view contains the following contextual menu actions:

Compare with working copy

Compares the selected revision with your working copy file. It is available only when you select a file.

Open

Opens the selected revision of the file into the Editor. This is available only for files.

Open with

Displays the **Open with** dialog box to specify the editor in which the selected file will be opened.

Get Contents

Replaces the current version from the working copy with the contents of the selected revision from the history of the file. The *BASE* version of the file is not changed in the working copy so that after this action the file will appear as modified in a synchronization operation, that is newer than the *BASE* version, even if the contents is from an older version from history.

Save as

Allows you to save the contents of a file as it was committed at a certain revision. This option is available only when you access the history of a file.

Copy to

Copies to the repository the item whose history is displayed, using the selected revision. This option is active only when presenting the history for a repository item (URL).

Note: This action can be used to resurrect deleted items also.

Revert changes from this revision

Reverts changes that were made in the selected revisions. The changes are reverted only in your working copy and does not affect the repository items. It does not replace your working copy items with those from the selected revisions. This action is available when the resource history was launched for a local working copy resource.

Note: For items displayed in the **Affected Paths** section that were *added*, *deleted*, or *replaced*, this action has no effect because such changes are considered to be changes to the parent directory. To revert these type of changes, follow these steps:

1. Request the history for the parent directory.
2. Identify the revision that contains the changes you want to revert.
3. Invoke the action on that revision.

Warning: There are instances where the SVN Client is not able to identify the corresponding working copy item for the selected item in the **Affected Paths** section. In this case, the action does not proceed and an error message is displayed. For example, the selected item in the **Affected Paths** section is from a different repository location than the working copy item for which the history is displayed.

Update to revision

Updates your working copy resource to the selected revision. This is useful if you want your working copy to reflect a time in the past. It is best to update a whole directory in your working copy, not just one file. Otherwise, your working copy is inconsistent and you are unable to commit your changes.

Check out

Checks out a new working copy of the directory for which the history is presented, from the selected revision.

Export

Opens *the **Export** dialog box* that allows you to configure options for exporting a folder from the repository to the local file system.

Show Annotation (Ctrl + Shift + A (Command + Shift + A on OS X))

Opens the **Show Annotation** dialog box that computes *the annotations for a file and displays them in the **Annotations** view*, along with the history of the file in the **History** view.

Change

Allows you to change commit data for a file:

- *Author* - Changes the name of the SVN user that committed the selected revision.
- *Message* - Changes the commit message of the selected revision.

When two resources are selected in the **History** view, the contextual menu contains the following actions:

Compare revisions

When the resource is a file, the action compares the two selected revisions using the **Compare** view. When the resource is a folder, the action displays the set of all resources from that folder that were changed between the two revision numbers.

Revert changes from these revisions

Similar to the `svn merge` command, it merges two selected revisions into the working copy resource. This action is only available when the resource history was requested for a working copy item.

For more information about the **History view** and its features, see the *Request history for a resource* and *Using the resource history view* sections.

Directory Change Set View

The result of comparing two reference revisions from the history of a folder resource is a set with all the resources changed between the two revision numbers. The changed resources can be contained in the folder or in a subfolder of that folder. These resources are presented in a tree format. For each changed resource all the revisions committed between the two reference revision numbers are presented.

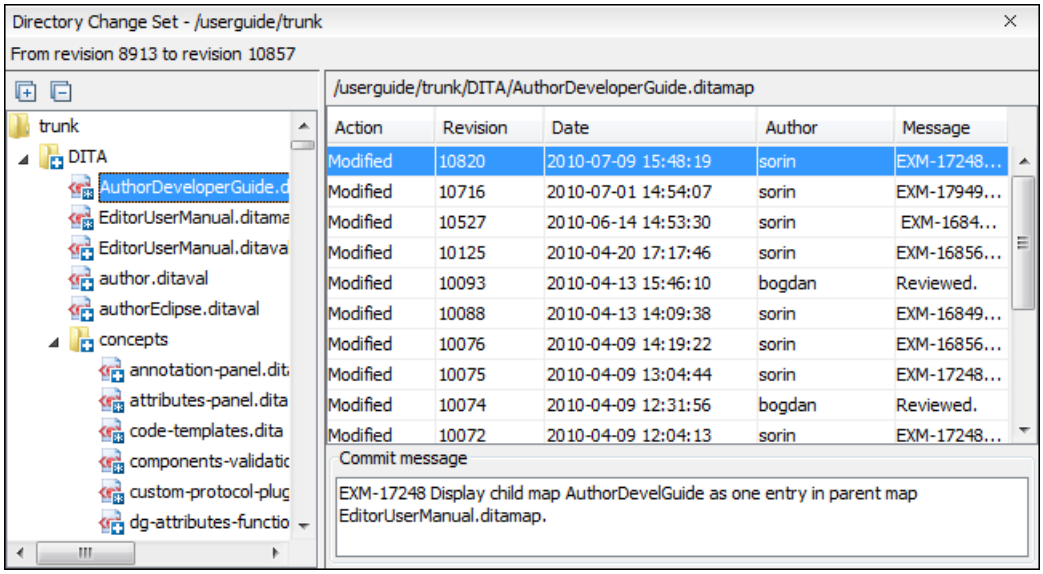


Figure 684: Directory Change Set View

The set of changed resources displayed in the tree is obtained by running the action **Compare revisions** available on the contextual menu of the **History** view when two revisions of a folder resource are selected in the **History** view.

The left side panel of the view contains the tree hierarchy with the names of all the changed resources between the two reference revision numbers. The right side panel presents the list with all the revisions of the resource selected in the left side tree. These revisions were committed between the two reference revision numbers. Selecting one revision in the list displays the commit message of that revision in the bottom area of the right side panel.

Double-clicking a file listed in the left-side tree performs a diff operation between the two revisions of the file corresponding to the two reference revisions. Double-clicking one of the revisions displayed in the right side list of the view performs a diff operation between that revision and the previous one of the same file.

The contextual menu of the right side list contains the following actions:

Compare with previous version

Performs a diff operation between the selected revision in the list and the previous one.

Open

Opens the selected revision in the associated editor type.

Open with

Displays a dialog box with the available editor types and allows you to select the editor type for opening the selected revision.

Save as

Saves the selected file as it was in the selected revision.

Copy to

Copies to the repository the item whose history is displayed, using the selected revision.

Note: This action can be used to resurrect deleted items also.

Check out

Checks out a new working copy of the selected directory, from the selected revision.

Export

Opens *the **Export** dialog box* that allows you to configure options for exporting a folder from the repository to the local file system.

Show Annotation (Ctrl + Shift + A (Command + Shift + A on OS X))

Opens the **Show Annotation** dialog box that computes *the annotations for a file and displays them in the **Annotations** view*, along with the history of the file in the **History** view.

Show SVN Information (Ctrl (Command on OS X) + I)

Provides additional information for a selected resource. For more details, go to *Obtain information for a resource*.

Editor Panel of SVN Client

You can open a file for editing in an internal built-in editor. There are default associations between frequently used file types and the internal editors in *the File Types preferences panel*.

The internal editor can be accessed either from the *Working copy view* or from the *History view*. No actions that modify the content are allowed when the editor is opened with a revision from history.

Only one file at a time can be edited in an internal editor. If you try to open another file it will be opened in the same editor window. The editor provides syntax highlighting for known file types. This means that a different color will be used for each recognized token type found in the file. If the file's content type is unknown you will be prompted to choose the proper way the file should be opened.

After editing the content of the file in an internal editor you can save it to disk by using the **Save** action from the *File* menu or the **Ctrl + S (Command + S on OS X)** key shortcut. After saving your file you can see the file changed status in *the **Working Copy** view*.

If the internal editor associated with a file type is not the XML Editor, then the encoding set in *the preferences for Encoding for non XML files* is used for opening and saving a file of that type. This is necessary because in the case of XML files, the encoding is usually declared at the beginning of the XML file in a special declaration or it assumes the default value UTF-8, but in the case of non-XML files, there is no standard mechanism for declaring the encoding for the file.

Annotations View

Sometimes you need to know not only what was changed in a file, but also who made those changes. The **Annotations** view displays the revision and the author that changed every line in a file. The annotations of a file are computed and this view is opened with the **Show Annotation** action, which is available in the **History** menu, and from the contextual menu of the following views: *the **Repositories** view*, ***Working copy** view*, ***History** view*, and ***Directory Change Set** view*..

This action opens a dialog box that allows you to configure some options for showing the annotations.

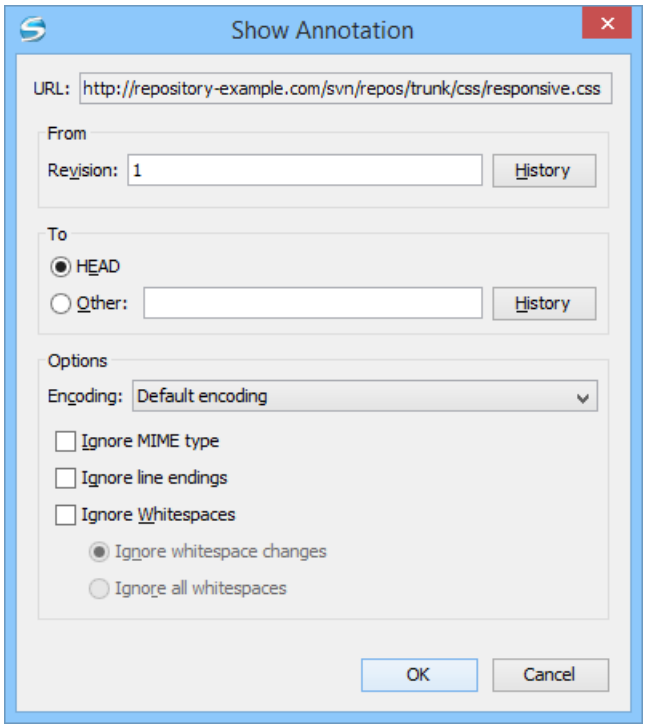


Figure 685: Show Annotation Options Dialog Box

Once you have configured the options and click **OK**, the **Annotations** view is displayed (by default, on the right side of the application). You can click a line in the editor panel where the file is opened to see the revision in which the line was last modified. The same revision is highlighted in the **History view** and you can also see all the lines that were changed in the same revision highlighted in the editor panel. Also, the entries of the **Annotations view** corresponding to that revision are highlighted. Therefore, the **Annotations view**, **History view**, and annotations editor panel are all synchronized. Clicking a line in one of them highlights the corresponding lines in the other two.

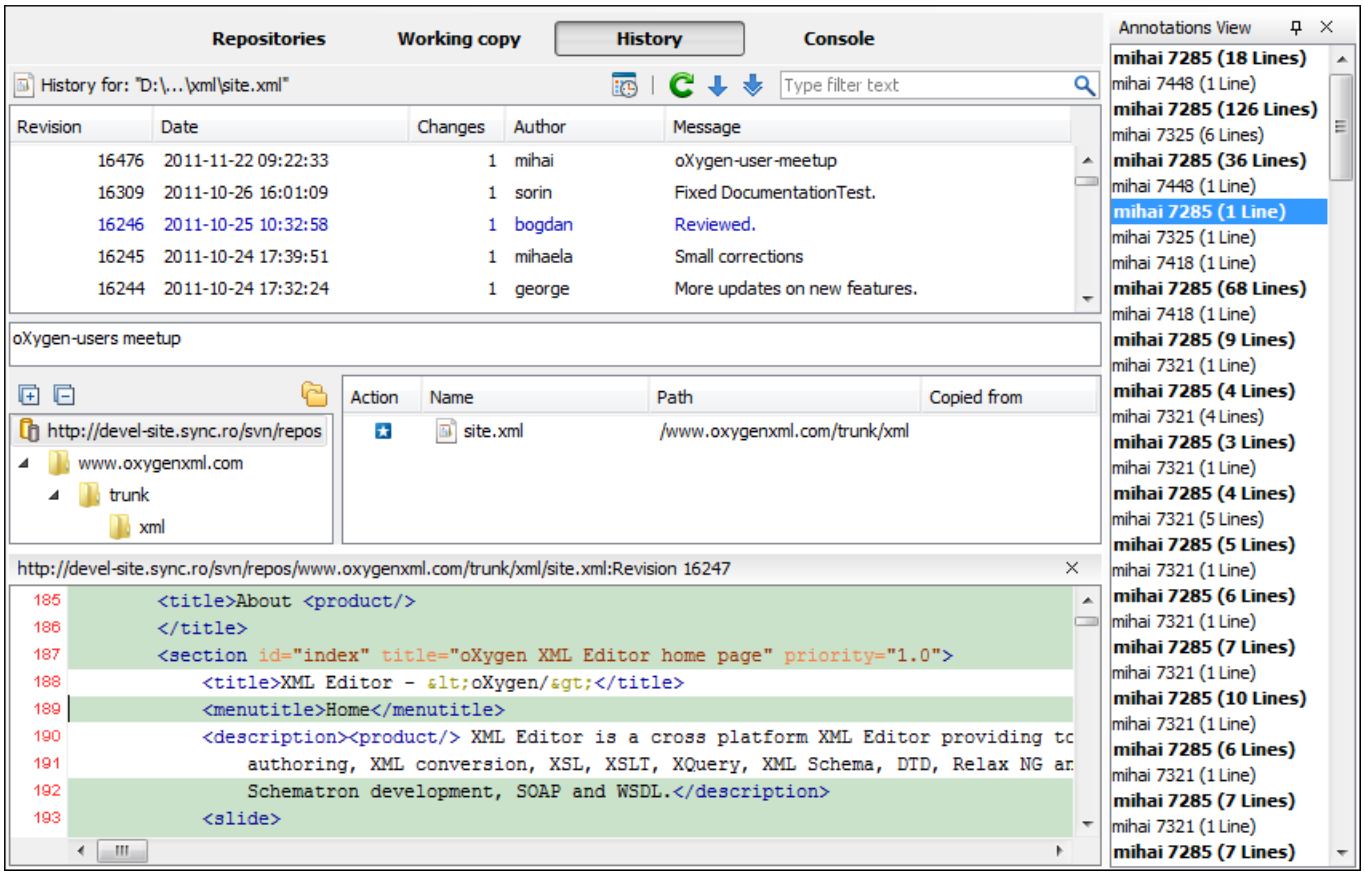


Figure 686: Annotations View

The following options can be configured in the **Show Annotation** dialog box:

From Revision Section

Select the revision from which to start computing the annotation. If you press the **History** button, *the **History** dialog box* is displayed, which allows you to select a revision.

To Revision Section

Select the ending revision by choosing between the **HEAD** revision or specify it in the **Other** text box . If you press the **History** button, *the **History** dialog box* is displayed, which allows you to select a revision.

Encoding

Select the encoding to be used when the annotation is computed. For each line of text, the SVN Client looks through the history of the file to be annotated see when it was last modified, and by whom. It is required that it is in the form of a text file. Therefore, encoding is needed to properly decode and read the file content. By default, the encoding of the operating system is used.

Ignore MIME type

If selected, the file is treated as a text file and ignores what the SVN system infers from the `svn:mime-type` property.

Ignore line endings

If selected, the differences in line endings are ignored when the annotation is computed.

Ignore whitespaces

If selected, it allows you to specify how the whitespace changes should be handled. When selected, you can then choose between two options:

- **Ignore whitespace changes** - Ignores changes in the amount of whitespaces or to their type (for example, when changing the indentation or changing tabs to spaces).

  Note: Whitespaces that were added where there were none before, or that were removed, are still considered to be changes.

- **Ignore all whitespaces** - Ignores all types of whitespace changes.

Tip: Selecting any of these *ignore* options can help you better determine the last time a meaningful change was made to a given line of text.

After you configure the options and press **OK**, the annotations will be computed and the **Annotations** view is displayed, where all the users that modified the selected resource will be presented, along with the specific lines and revision numbers modified by each user.

Note: If the file has a very long history, the computation of the annotation data can take a long time to process.

Compare View

In the Oxygen XML Editor, there are three types of files that can be checked for differences: text files, image files and binary files. For the text files and image files you can use the built-in **Compare** view. This view is automatically opened if you select two files and use the **Compare with** > **Each Other** action in the contextual menu.

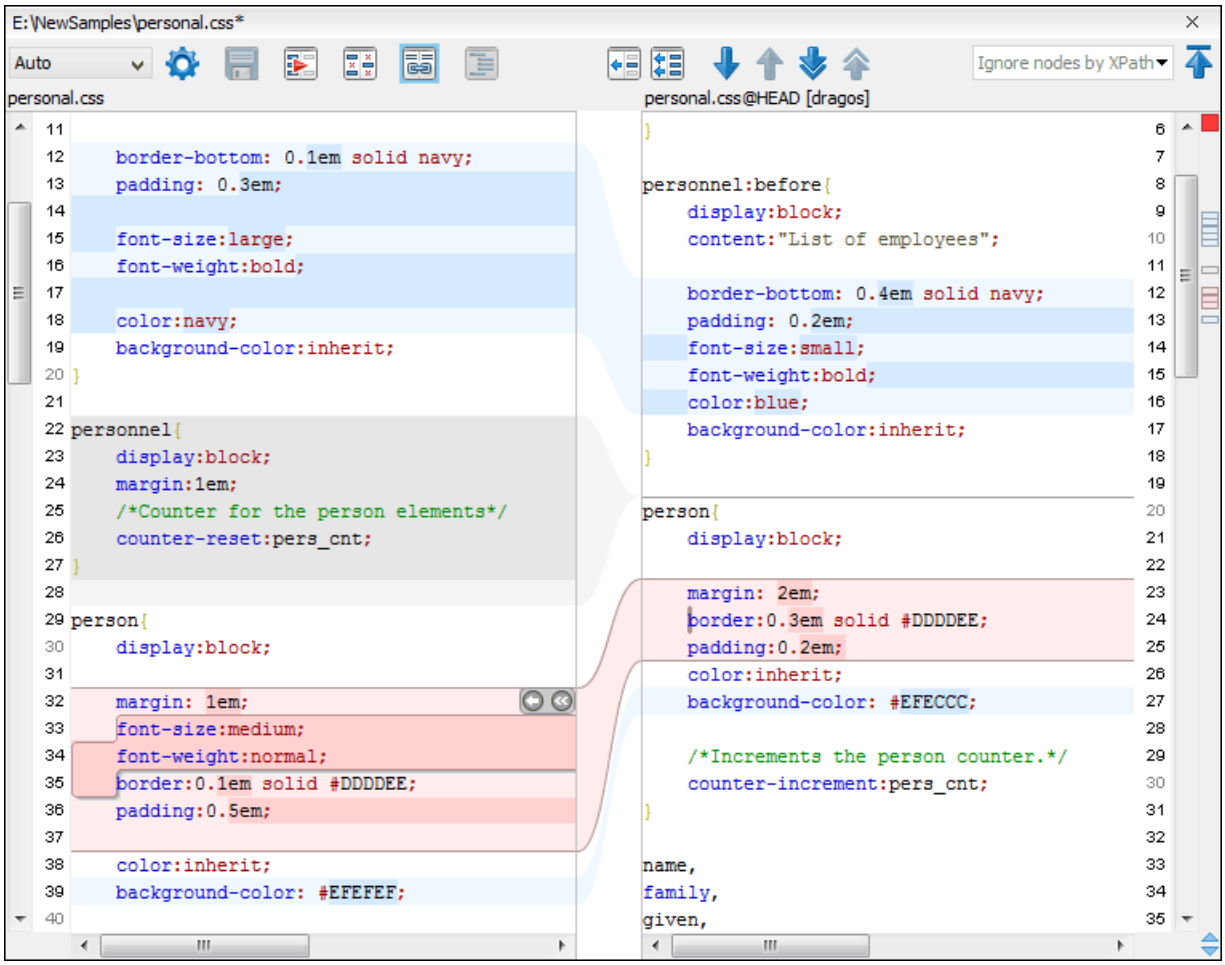


Figure 687: Compare View

At the top of each of the two editors, there are presented the name of the opened file, the corresponding SVN revision number (for remote resources) and the author who committed the associated revision.

When comparing text, the differences are computed using a *line differencing algorithm*. The view can be used to show the differences between two files in the following cases:

- After obtaining the outgoing status of a file with a **Refresh** operation, the view can be used to show the differences between your working file and the pristine copy. In this way you can find out what changes you will be committing.
- After obtaining the incoming and outgoing status of the file with the **Synchronize** operation, you can examine the exact differences between your local file and the *HEAD* revision file.

- You can use the **Compare view** from the **History view** to compare the local file and a selected revision or compare two revisions of the same file.

The Compare view contains two editors. Edits are allowed only in the left editor and only when it contains the working copy file. To learn more about how the view can be used, see *View Differences*.

Compare View Toolbar

The toolbar of the **Compare** view contains the operations that can be performed on the source and target files.

Figure 688: Compare View Toolbar

The following actions are available:

Algorithm

The algorithm to be used for performing a comparison. The following options are available:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Save action

Saves the content of the left editor when it can be edited.

Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.

Ignore Whitespaces

Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces.

Synchronized scrolling

Toggles synchronized scrolling. When toggled on, a selected difference can be seen in both panels.

Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on OS X))

Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.

Note: When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.

Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.

Copy All Changes from Right to Left

Copies all changes from the file in the right panel to the file in the left panel.

Next Block of Changes (Ctrl + Period (Command + Period on OS X))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

Note: A change block groups one or more consecutive lines that contain at least one change.

Previous Block of Changes (Ctrl + Comma (Command + Comma on OS X))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

Next Change (Ctrl + Shift + Period (Command + Shift + Period on OS X))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

Previous Change (Ctrl + Shift + Comma (Command + Shift + M on OS X))

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

Ignore Nodes by XPath

You can use this text field to enter an *XPath expression* to ignore certain nodes from the comparison. It will be processed as XPath version 2.0. You can also enter the name of the node to ignore all nodes with the specified name (for example, if you want to ignore all ID attributes from the document, you could simply enter `@id`). This field is only available when comparing XML documents using the **XML Fast** or **XML Accurate** algorithms.

Note: If an XPath expression is specified in the ***Ignore nodes by XPath*** *option* in the **Diff / File Comparison** preferences page, that one is used as a default when the application is started. If you then enter an expression in this field on the toolbar, this one will be used instead of the default. If you delete the expression from this field, neither will be used.

First Change (Ctrl + B (Command + B on OS X))

Jumps to the first change.

Most of these actions are also available from the *Compare* menu.

Image Preview

You can view your local files by using the built-in **Image Preview** component. The view can be accessed from the *Working copy view* or from the *Repository view*. It can also be used from the *History view* to view a selected revision of a image file.

Only one image file can be opened at a time. If an image file is opened in the *Image preview* and you try to open another one it will be opened in the same window. Supported image types are *GIF*, *JPEG/JPG*, *PNG*, *BMP*. Once the image is displayed in the **Image Preview** panel using the actions from the contextual menu, you can scale the image at its original size (**1:1** action) or scale it down to fit in the view's available area (**Scale to fit** action).

Compare Images View

The images are compared using the **Compare Images** view. This view is automatically opened if you select two image files and use the **Compare with** > **Each Other** action in the contextual menu. The images are presented in the left and right part of the view, scaled to fit the available area. You can use the contextual menu actions to scale the images at their original size or scale them down to fit the view's available area.

The supported image types are: *GIF*, *JPG / JPEG*, *PNG*, *BMP*.

Properties View

The properties view presents Apache Subversion™ properties for the currently selected resource from either the **Working Copy** view or the **Repositories** view. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

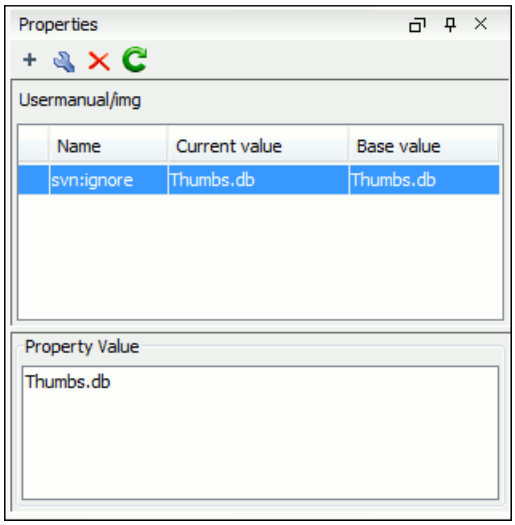


Figure 689: Properties View

The table includes four columns:

- **State** - Can be one of the following:
  - *(empty)* - Normal unmodified property (same current and base values)
  - ** (asterisk)* - Modified property (current and base values are different)
  - *+ (plus sign)* - New property
  - *- (minus sign)* - Removed property
- **Name** - The property name.
- **Current value** - The current value of the property.
- **Base value** - The base (original) value of the property.

`svn:externals` Property

The `svn:externals` property can be set on a folder or a file. In the first case it stores *the URL of a folder from other repository*.

In the second case it stores the URL of a file from other repository. The external file will be added into the working copy as a versioned item. There are a few differences between directory and file externals:

- The path to the file external must be in a working copy that is already checked out. While directory externals can place the external directory at any depth and it will create any intermediate directories, file externals must be placed into a working copy that is already checked out.
- The external file URL must be in the same repository as the URL that the file external will be inserted into; inter-repository file externals are not supported.
- While commits do not descend into a directory external, a commit in a directory containing a file external will commit any modifications to the file external.

The differences between a normal versioned file and a file external:

- File externals cannot be moved or deleted; the `svn:externals` property must be modified instead; however, file externals can be copied.

A file external is displayed as a X in the switched status column.

Toolbar / Contextual Menu

The properties view toolbar and contextual menu contain the following actions:

- **Add a new property** - This button invokes the *Add property* dialog box in which you can specify the property name and value.
- **Edit property** - This button invokes the *Edit property* dialog box in which you can change the property value and also see its original(base) value.

- **Remove property** - This button will prompt a dialog box to confirm the property deletion. You can also specify if you want to remove the property recursively.
- **Refresh** - This action will refresh the properties for the current resource.

Console View

The **Console View** shows the traces of all the actions performed by the application. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Part of the displayed messages mirror the communication between the application and the Apache Subversion™ server. The output is expressed as subcommands to the Subversion server and simulates the Subversion command-line notation. For a detailed description of the Subversion console output read the **SVN User Manual**.

The view has a simple layout, with most of its space occupied by a message area. On its right side, there is a toolbar holding the following buttons:

Clear

Erases all the displayed messages.

Lock scroll

Disables the automatic scrolling when new messages are appended in the view.

The maximum number of lines displayed in the console (length of the buffer) can be modified in the *Preferences* page. By default, this value is set to 100.

Dynamic Help View

Dynamic Help view is a help window that changes its content to display the help section that is specific to the currently selected view. As you change the focused view, you can read a short description of it and its functionality. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

Revision Graph of a SVN Resource

The history of a SVN resource can be watched on a graphical representation of all the revisions of that resource together with the tags in which the resource was included. The graphical representation is identical to a tree structure and very easy to follow.

The graphical representation of a resource history is invoked with the **Revision graph** action available on the right-click menu of a SVN resource in *the Working Copy view* and *the Repository view*.

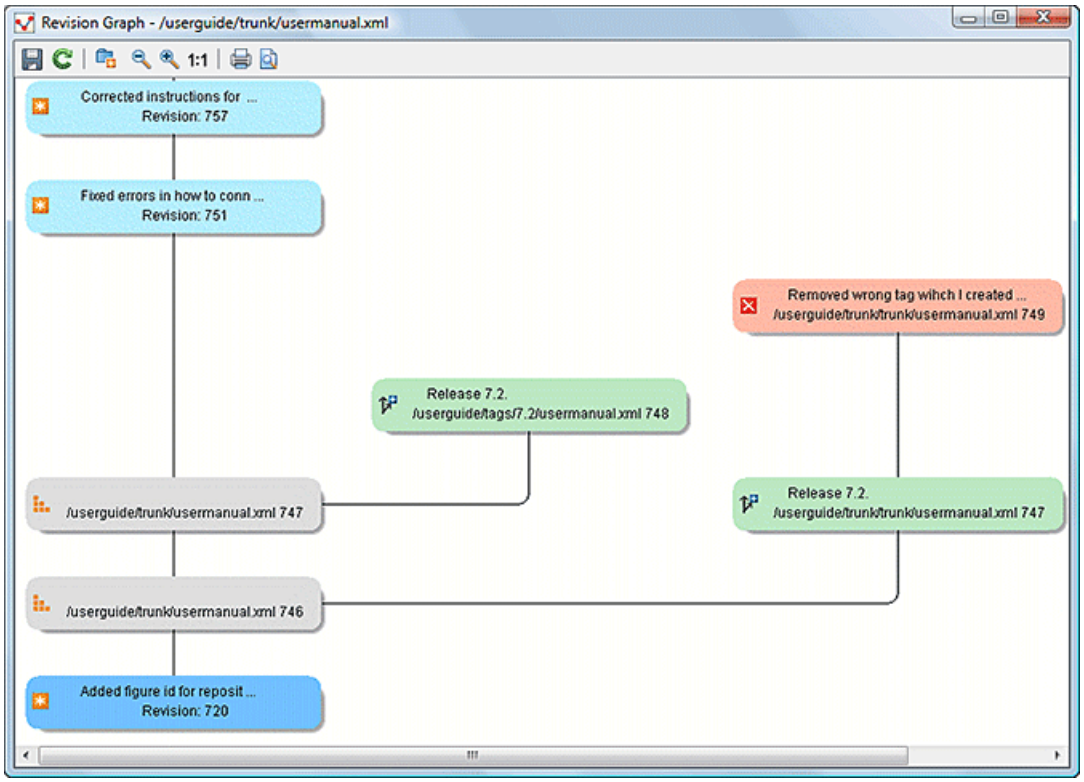


Figure 690: Revision Graph of a File Resource

In every node of the revision graph an icon and the background color represent the type of operation that created the revision represented in that node. The commit message associated with that revision, the repository path, and the revision number are also contained in the node. The tooltip displayed when the mouse pointer hovers over a node specifies the URL of the resource, the SVN user who created the revision of that node, the revision number, the date of creation, the commit message, the modification type and *the affected paths*.

The types of nodes used in the graph are:

Added resource

The icon for a new resource added to the repository and a green background.

Copied resource

The icon for a resource copied to other location (for example, when a SVN tag is created and a green background).

Modified resource

The icon for a modified resource and a blue background.

Deleted resource

The icon for a resource deleted from the repository and a red background.

Replaced resource

The icon for a resource removed and replaced with another one on the repository and a orange background.

Indirect resource

The icon for a revision from where the resource was copied or an indirectly modified resource, that is a directory in which a resource was modified and a gray background. The *Modification type* field of the tooltip specifies how that revision was obtained in the history of the resource.

A directory resource is represented with two types of graphs:

Simplified graph

Lists only the changes applied directly to the directory;

Complete graph

Lists also the indirect changes of the directory resource, that is the changes applied to the resources contained in the directory.

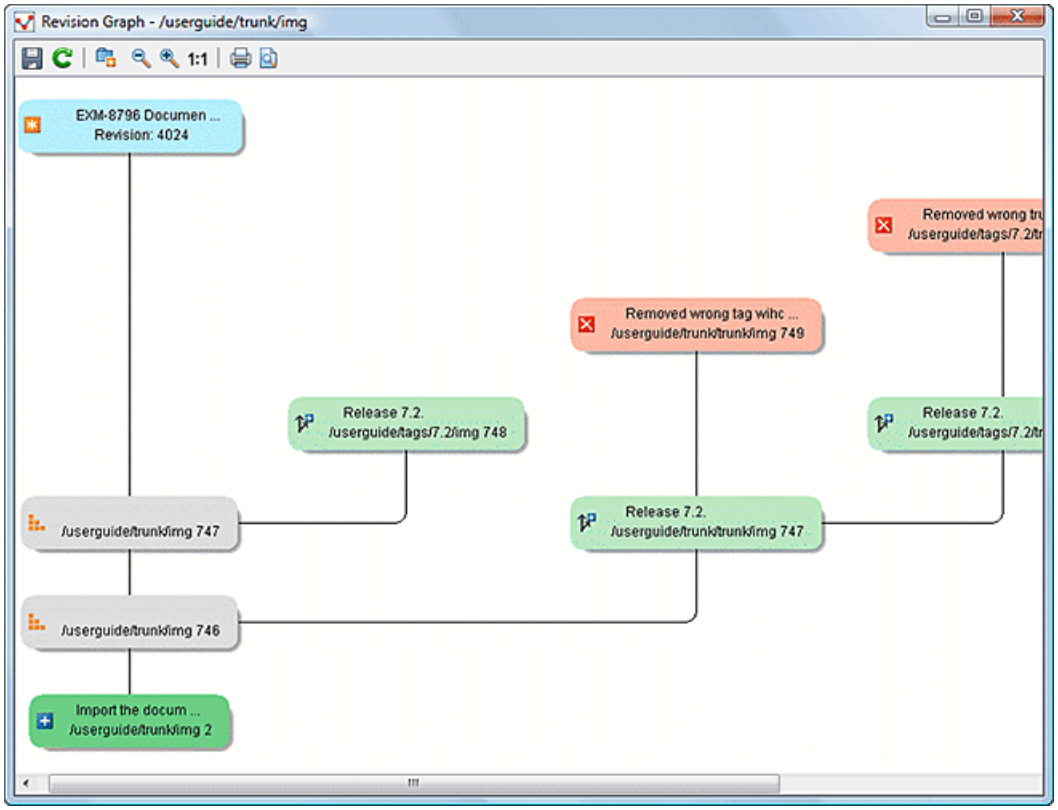


Figure 691: Revision Graph of a Directory (Direct Changes)

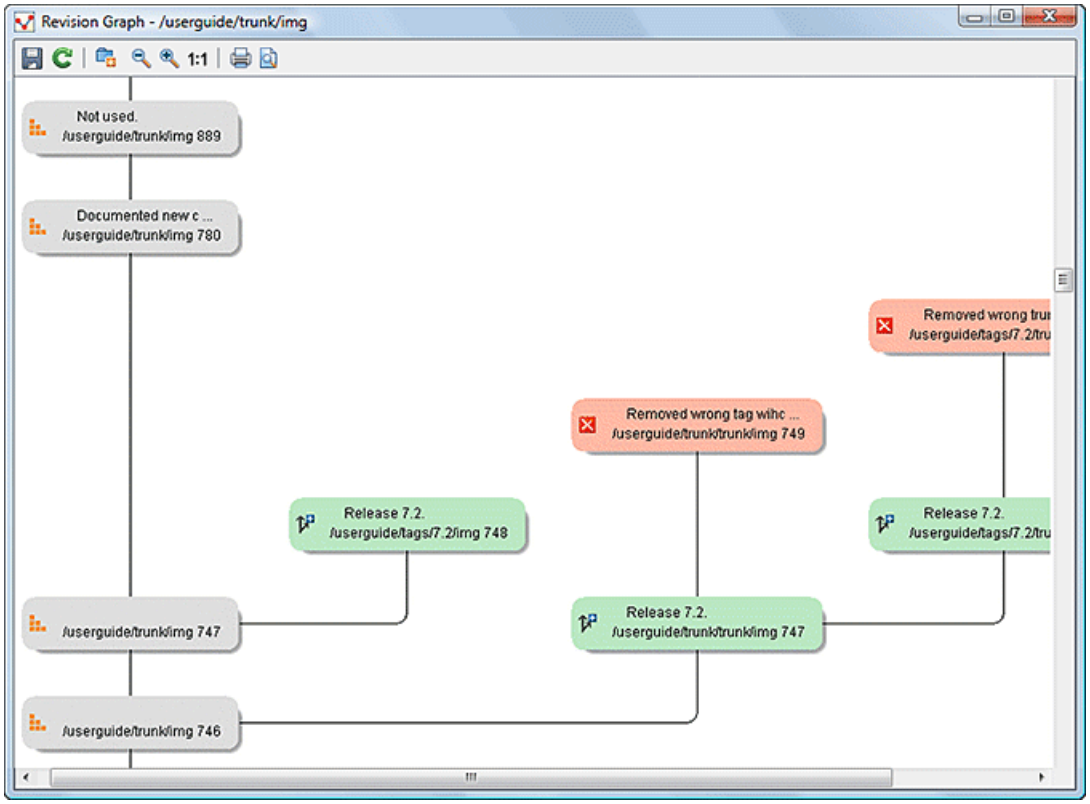


Figure 692: Revision Graph of a Directory (Including Indirect Changes)

The **Revision graph** toolbar contains the following actions:

Save as image

Saves the graphical representation as image. For a large revision graph you have to *set more memory in the startup script*. The default memory size is not enough when there are more than 100 revisions that are included in the graph.

/Show/Hide indirect modifications

Switches between simplified and complete graph.

Zoom In

Zooms in the graph.

Zoom Out

Zooms out the graph. When the font reaches its minimum size, the graph nodes will display only the icons, leading to a very compact representation of the graph.

1:1Reset scale

Resets the graphical scale to a default setting.

Print

Prints the graph.

Print preview

Offers a preview of the graph to allow you to check the information to be printed.

The contextual menu of any of the graph nodes contains the following actions:

Open

Opens the selected revision in the editor panel. Available only for files.

Open with

Opens the selected revision in the editor panel. Available only for files.

Save as

Saves the file for which the revision graph was generated, based on the selected node revision.

Copy to

Copies to the repository the item whose revision graph is displayed, using the selected revision.

Note: This action can be used to resurrect deleted items also.

Compare with HEAD

Compares the selected revision with the HEAD revision and displays the result in the diff panel. Available only for files.

Show History

Displays the history of the resource in *the History view*. Available for both files and directories.

Check out

Checks out the selected revision of the directory. Available only for directories.

Export

Opens *the **Export** dialog box* that allows you to configure options for exporting a folder from the repository to the local file system.

When two nodes are selected in the revision graph of a file the right-click menu of this selection contains only the **Compare** for comparing the two revisions corresponding to the selected nodes. If the resource for which the revision graph was built is a folder then the right-click menu displayed for a two nodes selection also contains the **Compare** action but it computes the differences between the two selected revisions as a set of directory changes. The result is displayed in the *Directory Change Set* view.

Attention: Generating the revision graph of a resource with many revisions may be a slow operation. You should enable caching for revision graph actions so that future actions on the same repository will not request the same data again from the SVN server, which will finish the operation much faster.

Oxygen XML Editor SVN Preferences

The options used in the SVN client are saved and loaded independently from the Oxygen XML Editor options. However, if Oxygen XML Editor cannot determine a set of SVN options to be loaded at startup, some of the preferences are imported from the XML Editor options (such as the License key and HTTP Proxy settings).

There is also an additional set of preferences applied to the SVN client that are set in global SVN files. There are two editing actions available in the **Global Runtime Configuration** submenu of the **Options** menu. These actions, **Edit 'config' file** and **Edit 'servers' file**, contain parameters that act as defaults applied to all the SVN client tools that are used by the same user on their login account.

Entering Local Paths and URLs

The Oxygen XML Editor includes a variety of option configuration pages or wizards that contain text boxes where you specify paths to local resources or URLs of items inside remote repositories. The Oxygen XML Editor provides support in these text boxes to make it easier to specify these paths and URLs.

Local Item Paths

The text boxes used for specifying local item paths support the following:

- *Absolute Paths* - In most cases, the Oxygen XML Editor expects absolute paths for local file system items.
- *Relative Paths* - The Oxygen XML Editor only accepts relative paths in the form ~[`/...`], where ~ is the user home directory.
- *Path Validation* - Oxygen XML Editor validates the path as you type and invalid text becomes red.
- *Drag and Drop* - You can drag files and folders from the file system or other applications and drop them into the text box.
- *Automatic Use of Clipboard Data* - If the text box is empty when its dialog box is opened, any data that is available in the system clipboard is used, provided that it is valid for that text box.

Repository Item URLs

- *Local Repository Paths* - You can use local paths (absolute or relative) to access local repositories. When you use the **Browse** button, the Oxygen XML Editor will convert the file path to a `file://` form of URL, provided that the location is a real repository.
  - *Absolute Paths* - In most cases, the Oxygen XML Editor expects absolute paths for local file system items.
  - *Relative Paths* - The Oxygen XML Editor only accepts relative paths in the form ~[`/...`], where ~ is the user home directory.
- *Peg Revisions* - For URL text boxes found inside dialog boxes where you are pulling information from the repository, you can *use peg revisions at the end of the URLs* (for example, URL`@rev1234`).

  Note: If you try to use a *peg revision* number in a dialog box where you are sending information to the repository then the peg revision number will become part of the name of the item rather than searching for the specified revision. For example, in the URL `http://host/path/inside/repo/item@100`, the item name is considered to be `item@100`.

  Tip: You can even use *peg revisions* with local repository paths. For example, `C:\path\to\local\repo@100` will be converted to `file:///C:/path/to/local/repo@100` and the **Repository browser** will display the content of the local repository as it is at revision `100`.
- *URL Validation* - Oxygen XML Editor validates the URLs as you type and invalid text becomes red. Even paths to local repositories are not accepted unless using the **Browse** button to convert them to valid URLs.
- *Drag and Drop* - You can drag URLs from other applications or text editors and drop them into the URL text box. You can also drag folders that point to local repositories, from the local file system or from other applications, and they are automatically converted to valid `file://` type URLs.
- *Automatic Use of Clipboard Data* - If the URL text box is empty when its dialog box is opened, any data that is available in the system clipboard is used, provided that it is valid for that text box. Even valid local paths will be automatically converted to `file://` type URLs.

Note: The text boxes that are in the form of a combo box also allow you to select previously used URLs, or URLs defined in the **Repositories** view.

Technical Issues

This section contains special technical issues found during the use of Syncro SVN Client.

Authentication Certificates Not Saved

If Syncro SVN Client prompts you to enter the authentication certificate, although you already provided it in a previous session, then you should make sure that your local machine user account has the necessary rights to store certificate files in the *Subversion* configuration folder (write access to *Subversion* folder and all its subfolders). Usually, it is located in the following locations:

- Windows: `[HOME_DIR]\AppData\Roaming\Subversion`
- Mac OS X and Linux: `[HOME_DIR]/.subversion`

Updating Newly Added Resources

When you want to get a resource that is part of a newly created structure of folders from the repository, you need to also get its parent folders.

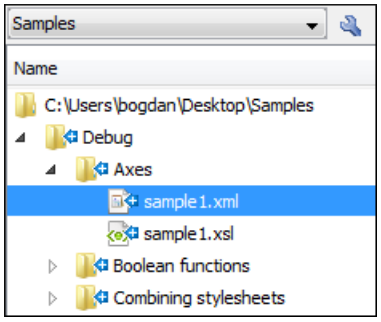


Figure 693: Incoming Changes

Syncro SVN Client allows you to choose how you want to deal with the entire structure from that moment onwards:

Update ancestor directories recursively

This option brings the entire newly added folders structure into your working copy. In this case, the update time depends on the total number of newly incoming resources, because of the full update operation (not updating only selected resource).

Update selected files only (leave ancestor directories empty)

This option brings a skeleton structure composed of the resource's parent folders only, and the selected resource at the end of the operation. All of the parent directories will have depth set to *empty* in your working copy, thus subsequent **Synchronize** operations will not report any remote modifications in those folders. If you need to update the folders to full-depth, you can use the ***Update to revision/depth*** *action* from the working copy view.

Cannot Access a Repository through HTTPS

If you have issues when trying to access a repository through HTTPS protocol, one of the possible causes can be the encryption protocol currently used by the application. This is happening when:

- You are running Oxygen XML Editor with Java 1.6 or older.
- The repository is set to use only one of the SSLv3 or TLSv1 encryption protocols.

To solve this issue, set the ***HTTPS encryption protocols*** *option* to **SSLv3 only** or **TLSv1 only** (depending on the repository configuration).

Accessing Old Items from a Repository

Usually, you point to an item from a repository using a URL. However, sometimes this might not be enough because the URL alone might point to a different item than the one you want and a *peg revision* is needed.

A Subversion repository tracks any change made to its items by using *revisions*, which contain information such as the name of the author who made the changes, the date when they were made, and a number that

uniquely identifies each of them. Over time, an item from a specific location in a repository evolves as a result of committing changes to it. When an item is deleted, its entire life cycle (all changes made to it from the moment it was created) remains recorded in the history of the repository. If a new item is created, with the same name and in the same location of the repository as a previously existing one, then both items are identified by the same URL even though they are located in different time frames. This is when a *peg revision* comes in handy. A *peg revision* is nothing more than a normal revision, but the difference between them is made by their usage. Many of the Subversion commands accept also a peg revision as a way to precisely identify an item in time, beside an *operative revision* (the revision we are interested in, regarding the used command).

Example:

To illustrate an example, consider the following:

- We created a new repository file `config`, identified by the URL `http://host.com/myRepository/dir/config`.
- The file has been created at revision `10`.
- Over time, the file was modified by committing revisions `12, 15, 17`.

To access a specific version of the file identified by the `http://host.com/myRepository/dir/config` URL, we need to use a corresponding revision (the operative revision):

- If we use a revision number less than `10`, an error is triggered, because the file has not been created yet.
- If we use a revision number between `10` and `19`, we will obtain the specific version we are interested in.

  Note: Although the file was modified in revisions `12, 15, 17`, it existed in all revisions between `10` and `19`. Starting with a revision at which the file is modified, it has the same content across all revisions generated in the repository until another revision in which it is modified again.

At this point, we delete the file, creating revision `20`. Now, we cannot access any version of the file, because it does not exist anymore in the latest repository revision. This is due to the fact that Subversion automatically uses the `HEAD` revision as a peg revision (it assumes any item currently exists in the repository if not instructed otherwise). However, using any of the revision numbers from the `10-19` interval (when the file existed) as a peg revision (beside the operative revision), will help Subversion to properly identify the time frame when the file existed, and access the file version corresponding to the operative revision. If we use a revision number greater than `19`, this will also trigger an error.

Continuing our example, suppose that at revision `30` we create a directory called `config` in the same repository location as our deleted file. This means that our new directory will be identified by the same repository address: `http://host.com/myRepository/dir/config`. If we use only this URL in any Subversion command, we will access the new directory. We will also access the same directory if we use any revision number equal with or greater than `30` as peg revision. However, if we are interested in accessing an old version of the previously existing file, then we must use one of the revisions that existed (`10-19`), as a peg revision, similar to the previous case.

Checksum Mismatch Error

A *Checksum Mismatch* error could happen if an operation that sends or retrieves information from the repository to the working copy is interrupted. This means that there is a problem with the synchronization between a local item and its corresponding remote item.

If you encounter this error, try the following:

1. Identify the parent directory of the file that caused the error (the file name should be displayed in the error message).

   Note: If the parent directory is the root of the working copy or if it contains a large amount of items it is recommended that you check out the working copy again, rather than continuing with the rest of this procedure.
2. Identify the *current depth* of that directory.
3. Update the parent directory using the **Update to revision/depth** action that is available from the contextual menu or the **Working copy** menu.

   a. For the **Depth** option, select **This folder only (empty)**.

Warning: If you have files with changes in this directory, those changes could be lost. You should commit your changes or move the files to another directory outside the working copy prior to proceeding with this operation.

4. After clicking **OK** the contents of the directory will be erased and the directory is be marked as having *an empty depth*.
5. Once again, update the same directory using the **Update to revision/depth** action.
   a. This time, for the **Depth** option, select the depth that was previously identified in step 2.
6. If you moved modified files to another directory outside the working copy, move them back to the original location inside the working copy.

If this procedure does not solve the error, you need to check out the working copy again and move possible changes from the old working copy to the new one.

External Tools

A command-line tool can be started in the Oxygen XML Editor user interface as if from the command line of the operating system shell. Oxygen XML Editor offers you the option of integrating such a tool by specifying just the command line for starting the executable file and its working directory. To integrate such a tool, *open the **Preferences** dialog box **(Options > Preferences)*** and go to ***External Tools*** (or select **Configure** from the **Tools** > **External Tools** menu).

The ***External Tools** preferences page* presents a list of the external tools that have been configured. Once *a tool has been configured*, you can open it by selecting it from the **Tools** > **External Tools** menu or from the **External Tools** drop-down menu on the toolbar (the **Tools** toolbar needs to be selected in the ***Configure Toolbars** dialog box*). You can also *assign a keyboard shortcut* to be used to launch the tool.

If the external tool is applied on one of the files opened in Oxygen XML Editor, the ***Save all files before calling external tools** option* (in the **Open/Save** preference page) should be selected so that all edited files are automatically saved when an external tool is applied.

When an external tool is launched, the icon on the toolbar changes to a stop icon ( ) and you can use this button to stop the tool. When the tool has finished running (or you close it), the icon changes back to the original icon ( ).

Note: Even though you can stop the external tool by invoking the stop action while it is running, that does not necessarily mean it will also stop the processes spawned by that external tool. For instance, if you stop an external tool that runs a batch file, the batch may be stopped but without actually stopping the processes that the batch was running at that time.

Example: Integrating the Ant Tool

This is an example procedure for integrating *the Ant build tool* into Oxygen XML Editor:

1. *Download* and *install Apache Ant* on your computer.
2. Test your *Ant* installation from the command-line interface in the directory where you want to use *Ant* from. For example, run the `clean` target of your `build.xml` file `C:\projects\XMLproject\build.xml`:

```
ant clean
```

3. *Open the **Preferences** dialog box **(Options > Preferences)*** and go to **External Tools** (or select **Configure** from the **Tools** > **External Tools** menu).
4. Click the **New** button to create a new external tool entry and enter the following information:
   - **Name** - For example, `Ant tool`.
   - **Working directory** - For example, `C:\projects\XMLproject`.
   - **Command line** - For example, `"C:\projects\XMLproject\ant.bat" clean`.
5. Click **OK** to add the new tool to the list of external tools.
6. Run the tool from **Tools** > **External Tools** > **Ant tool**. You can see the output in the system console:

```
    Started: "C:\projects\XMLproject\ant.bat" clean
    Buildfile: build.xml

    clean:
    [echo] Delete output files.
    [delete] Deleting 5 files from C:\projects\XMLproject

    BUILD SUCCESSFUL
    Total time: 1 second
```

Common Problems

18

Topics:

- *Performance Problems and Solutions*
- *Misc Problems and Solutions*

This section provides a collection of common performance and other types of problems that might be encountered when using Oxygen XML Editor, along with their possible solutions.

Performance Problems and Solutions

This section contains solutions for some common performance problems that may appear when running Oxygen XML Editor.

Related Information:

Editing Documents with Long Lines on page 736

Loading Large Documents on page 734

External Tools on page 1562

Display Problems on Linux or Solaris

Problem

I experience display problems (such as screen freezes) on Linux or Solaris.

Solution

Add the following startup parameter: -Dsun.java2d.pmoffscreen=false.

Out of Memory on External Processes

Problem

Oxygen XML Editor throws an *Out Of Memory* error when trying to generate PDF output with the built-in Apache FOP processor.

Solution

*Open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **XML** > **XSLT-FO-XQuery** > **FO Processors**, and increase the value of the ***Memory available to the built-in FOP** option*.

For external XSL-FO processors, XSLT processors, and external tools, the maximum value of the allocated memory is set in the command line of the tool using the -Xmx parameter set to the Java virtual machine.

Related Information:

FO Processors Preferences on page 129

Custom Engines Preferences on page 133

External Tools Preferences on page 150

Too many nested apply-templates calls Error When Running a Transformation

Problem

I'm getting the error message **Too many nested apply-templates calls** when I try to transform my DocBook file to HTML using default Oxygen XML Editor DocBook to HTML transformation scenario.

Solution

Most likely, this is the result of a masked stack overflow error that can be solved by increasing the stack size (**-Xss**) to 4MB. Try setting a new VM option with the value **-Xss4m**. You can try to slowly increase this to larger values (e.g. **-Xss5m** or **-Xss6m**). Note that this consumes memory on a per thread basis (Oxygen XML Editor can have tens of threads), so using a very large value here can backfire and leave Oxygen XML Editor without memory.

Related Information:

Setting a Java Virtual Machine Parameter when Launching Oxygen XML Editor on page 178

Performance Issues with Large Documents

Problem

The performance of the application slows down considerably over time when editing large documents.

Solution

A possible cause is that the application needs more memory to run properly. You can increase the maximum amount of memory available to Oxygen XML Editor by *setting the -Xmx parameter in a configuration file* that is specific to the platform that runs the application.

Attention: The maximum amount of memory should not be equal to the physical amount of memory available on the machine because in that case the operating system and other applications will have no memory available.

When installed on a multiple user environment, each instance of Oxygen XML Editor will be allocated the amount stipulated in the memory value. To avoid degrading the general performance of the host system, ensure that the amount of memory available is optimally apportioned for each of the expected instances.

Note: When starting Oxygen XML Editor from the icon created on the Start menu or Desktop in Windows (or from the shortcut created on the Linux desktop), the default maximum memory available to the application is set to 40% of the amount of physical RAM, but not more than 700 MB.

When starting Oxygen XML Editor from a command-line script, the default maximum memory is 512 MB.

Misc Problems and Solutions

This chapter presents common problems that may appear when running the application along with solutions for these problems.

'Address Family Not Supported by Protocol Family; Connect' Error

Problem

I have experienced the following error: *"Address Family Not Supported by Protocol Family; Connect"*. How do I solve it?

Solution

This seems to be an IPv6 connectivity problem. By default, the Java runtime used by Oxygen XML Editor prefers to create connections via IPv6, if the support is available. However, even though it is available in appearance, IPv6 sometimes happens to be configured incorrectly on some systems.

A quick solution for this problem is to set the `java.net.preferIPv4Stack` Java property to `true` (`java.net.preferIPv4Stack=true`), by following this procedure:

1. Create a file named `custom_commons.vmoptions` and add on a single line `-Djava.net.preferIPv4Stack=true`. Then save the file and copy it to the Oxygen XML Editor installation folder (may need admin access).
2. Restart Oxygen XML Editor.
3. Make sure the procedure was successful by going to **Help** > **About** > **System properties** and check that the value of the `java.net.preferIPv4Stack` property is `true`.

Alignment Issues of the Main Menu on Linux Systems Based on Gnome 3.x

Problem

On some Linux systems based on Gnome 3.x (Ubuntu 11.x, 12.x), the main menu of Oxygen XML Editor has alignment issues when you navigate it using your mouse.

Solutions

This is a known problem caused by Java SE 6 1.6.0_32 and earlier. You can resolve this problem using the latest Java SE 6 JRE from Oracle. To download the latest version, go to *http://www.oracle.com/technetwork/java/javase/downloads/index.html*.

To bypass the JRE bundled with Oxygen XML Editor, go to the installation directory of Oxygen XML Editor and rename or move the `jre` folder. If Oxygen XML Editor does not seem to locate the system JRE, either set the JAVA_HOME environment variable to point to the location where you have installed the JRE, or you can simply copy that folder with the JRE to the installation directory and rename it to `jre` to take the place of the bundled JRE.

Archive Distribution Fails to Run on Mac OS 10.12 (Sierra)

Problem

For versions prior to 18.1, the classic archive distributions of Oxygen XML Editor (`.zip` or `.tar.gz`) fail to run on Mac OS 10.12 (Sierra).

Solution

This happens because the archives get quarantined and Mac OS 10.12 (Sierra) treats quarantined apps differently than older versions and isolates them from their parent folder at launch. If Oxygen XML Editor was already installed when you upgraded to Mac OS 10.12 (Sierra), there are no problems.

If Oxygen XML Editor was not already installed, or you need to reinstall an older version (prior to version 18.1), the quarantine flag must be removed for the entire content of the Oxygen XML Editor installation directory or the individual applications. To resolve this issue, follow these steps:

1. Open a *Terminal* window and change the directory to the folder where Oxygen XML Editor is located.
2. Run the following command:

   ```
   xattr -dr com.apple.quarantine oxygen/
   ```

   where "oxygen" is the actual name of the Oxygen XML Editor directory.

   If Oxygen XML Editor is in a location that requires administrator privileges for write access, you need to run the command from an administrator account and prefix the command with `sudo`. You will then be prompted to enter your password.

Cannot Associate Oxygen XML Editor With a File Type on My Windows Computer

Problem

I cannot associate the Oxygen XML Editor application with a file type on my Windows computer by right clicking a file in Windows Explorer, selecting **Open With** > **Choose Program** and browsing to the file oxygen19.0.exe. When I select the file oxygen19.0.exe in the Windows file browser dialog box, the Oxygen XML Editor application is not added to the list of applications in the **Open With** dialog box. What can I do?

Solution

The problem is due to some garbage Windows registry entries remained from versions of Oxygen XML Editor older than version 9.0. Uninstall all your installed versions of Oxygen XML Editor and run a registry cleaner application for cleaning these older entries. After that just reinstall your current version of Oxygen XML Editor and try again to create the file association.

Cannot Connect to SVN Repository from Repositories View

Problem

I cannot connect to a SVN repository from the **Repositories** view of SVN Client. How can I find more details about the error?

Solution

First check that you entered the correct URL of the repository in the **Repositories** view. Also check that a SVN server is running on the server machine specified in the repository URL and is accepting connections from SVN clients. You can check that the SVN server accepts connections with the command line SVN client from CollabNet.

If you try to access the repository with a `svn+ssh` URL also check that a SSH server is running on port 22 on the server machine specified in the URL.

If the above conditions are verified and you cannot connect to the SVN repository please generate a logging file on your computer and send the logging file to support@oxygenxml.com. For generating a logging file you need to create a text file called `log4j.properties` in the install folder with the following content:

```
log4j.rootCategory= debug, R2

log4j.appender.R2=org.apache.log4j.RollingFileAppender
log4j.appender.R2.File=logging.log
log4j.appender.R2.MaxFileSize=12000KB
log4j.appender.R2.MaxBackupIndex=20
log4j.appender.R2.layout=org.apache.log4j.PatternLayout
log4j.appender.R2.layout.ConversionPattern=%r %p [ %t ] %c - %m%n
```

Restart the application, reproduce the error, close the application and send the file `logging.log` generated in the install directory to support@oxygenxml.com.

Cannot Open XML Files in Internet Explorer

Problem

Before installing Oxygen XML Editor I had no problems opening XML files in Internet Explorer. Now when I try to open an XML file in Internet Explorer it opens the file in Oxygen XML Editor. How can I load XML files in Internet Explorer again?

Solution

XML files are opened in Oxygen XML Editor because Internet Explorer uses the Windows system file associations for opening files and you associated XML files with Oxygen XML Editor in the installer panel called **File Associations**. Therefore, Internet Explorer opens XML files with the associated Windows application.

By default, the association with XML files is disabled in the Oxygen XML Editor installer panel called **File Associations**. When you enabled it, the installer displayed a warning message about the effect that you experience right now.

For opening XML files in Internet Explorer, again you have to set Internet Explorer as the default system application for XML files (for example by right-clicking an XML file in Windows Explorer, selecting **Open With** > **Choose Program**, selecting IE in the list of applications and selecting the checkbox **Always use the selected program**). Also, you have to run the following command from a command line:

```
wscript revert.vbs
```

where `revert.vbs` is a text file with the following content:

```
function revert()
  Set objShell = CreateObject("WScript.Shell")
  objShell.RegWrite "HKCR\.xml\", "xmlfile", "REG_SZ"
  objShell.RegWrite "HKCR\.xml\Content Type", "text/xml", "REG_SZ"
end function

revert()
```

Compatibility Issue Between Java and Certain Graphics Card Drivers

Problem

Under certain settings, a compatibility issue can appear between Java and some graphics card drivers, which results in the text from the editor (in **Author** or **Text** mode) being displayed garbled.

Solution

If you encounter this problem, update your graphics card driver. Another possible workaround is, *open the **Preferences** dialog box (**Options** > **Preferences**)*, go to **Appearance** > **Fonts**, and set the value of the ***Text antialiasing** option* to ON.

Note: If this workaround does not resolve the problem, set the ***Text antialiasing** option* to other values.

Crash at Startup on Windows with an Error About the `nvoglv32.dll` File

Problem

I try to start Oxygen XML Editor on Windows but it crashed with an error message about “Fault Module Name: nvoglv32.dll”. What is the problem?

Solution

It is an OpenGL driver issue that can be avoided by creating an empty file called `opengl32.dll` in the Oxygen XML Editor install folder (if you start Oxygen XML Editor with the shortcut created by the installer on the Start menu or on Desktop) or in the subfolder `bin` of the home folder of the Java virtual machine that runs Oxygen XML Editor (if you start Oxygen XML Editor with the `oxygen.bat` script). The home folder of the Java virtual machine that runs Oxygen XML Editor is the value of the `java.home` property that is available in the **System properties** tab of the **About** dialog box (opened from the **Help** > **About** menu.

Damaged File Associations on OS X

Problem

After upgrading OS X and Oxygen XML Editor, it is no longer associated to the appropriate file types (such as XML, XSL, XSD, etc.) How can I create the file associations again?

Solution

The upgrade damaged the file associations in the LaunchService Database on your OS X machine. You can rebuild the LaunchService Database with the following procedure. This will reset all file associations and will rescan the entire file system searching for applications that declare file associations and collecting them in a database used by Finder.

1. Find all the Oxygen XML Editor installations on your hard drive.
2. Delete them by dragging them to the Trash.
3. Clear the Trash.
4. Unpack the Oxygen XML Editor installation kit on your desktop.
5. Copy the contents of the archive into the folder `/ Applications / Oxygen`.
6. Run the following command in a Terminal:

   ```
   /System/Library/Frameworks/CoreServices.framework/Versions/A/Frameworks/
   LaunchServices.framework/Versions/A/Support/lsregister -kill -r -domain local -
   domain system -domain user
   ```
7. Restart Finder with the following command:

   ```
   killall Finder
   ```
8. Create an XML or XSD file on your desktop. It should have the Oxygen XML Editor icon.
9. Double-click the file.
10. Accept the confirmation.

Result: When you start Oxygen XML Editor, the file associations should work correctly.

Details to Submit in a Request for Technical Support Using the Online Form

Problem

What details should I add to my request for technical support on the online form in the product website?

Solution

When completing a request for Technical Support using the online form, include as many details as possible about your problem. For problems where a simple explanation may not be enough for the Technical Support team to reproduce or address the issue (such as server connection errors, unexpected delays while editing a document, an application crash, etc.), you should generate a log file and attach it to the problem report. In the case of a crash, you should also attach the crash report file generated by your operating system.

If the text content of an XML document you want to send to the support team contains sensitive or private information, you can use the ***Randomize XML text content*** *action* to create filler content. Before using this action, you need to copy the initial XML resources and save them in a separate folder. Otherwise, you might lose your original information.

To generate an Oxygen XML Editor log file, follow these steps:

1. Create a text file called `log4j.properties` in the application installation folder, with the following content:

   ```
   log4j.rootCategory= debug, R2

   log4j.appender.R2=org.apache.log4j.RollingFileAppender
   log4j.appender.R2.File=${user.home}/Desktop/oxygenLog/oxygen.log
   log4j.appender.R2.MaxFileSize=12000KB
   log4j.appender.R2.MaxBackupIndex=20
   log4j.appender.R2.layout=org.apache.log4j.PatternLayout
   log4j.appender.R2.layout.ConversionPattern=%r %p [ %t ] %c - %m%n
   ```
2. Restart the application.
3. Reproduce the error.
4. Close the application.
5. Delete the `log4j.properties` file because it might cause performance issues if you leave it in the installation folder.

Important: The logging mode may severely decrease the performance of the application. Therefore, please do not forget to delete the `log4j.properties` file when you are done with the procedure.

The resulting log file is named `oxygen#.log` (for example, `oxygen.log`, `oxygen.log.1`, `oxygen.log.2`, etc.) and is located in the `Desktop\oxygenLog` folder.

DITA Map Transformation Fails (Cannot Connect to External Location)

Problem

DITA map transformation fails because it cannot connect to an external location.

Solution

The transformation is run as an external Ant process so you can continue using the application as the transformation unfolds. All output from the process appears in the **DITA Transformation** tab.

The HTTP proxy settings are used for the Ant transformation, so if the transformation fails because it cannot connect to an external location, you can check the *the **Proxy** preferences page*

DITA PDF Transformation Fails

Problem

The DITA to PDF transformation fails.

Solution

To generate the PDF output, Oxygen XML Editor uses the DITA Open Toolkit.

If your transformation fails you can detect some of the problems that caused the errors by running *the **Validate and Check for Completeness** action*. Depending on the options you select when you run it, this action reports errors such as topics referenced in other topics but not in the *DITA map*, broken links, and missing external resources.

You can analyze the **Results** tab of the DITA transformation and search for messages that contain text similar to `[fop] [ERROR]`. If you encounter this type of error message, edit the transformation scenario you are using and set the **clean.temp** parameter to **no** and the **retain.topic.fo** parameter to **yes**. Run the transformation, go to the temporary directory of the transformation, open the `topic.fo` file and go to the line indicated by the error. Depending on the XSL FO context try to find the DITA topic that contains the text that generates the error.

If none of the above methods helps you, go to **Help** > **About** > **Components** > **Frameworks** and check what version of the DITA Open Toolkit you are using. Copy the whole output from the DITA OT console output and either report the problem on the DITA User List or to support@oxygenxml.com.

DITA to CHM Transformation Fails

Oxygen XML Editor uses the DITA Open Toolkit and the HTML Help compiler (part of the Microsoft HTML Help Workshop) to transform DITA content into *Compiled HTML Help* (or *CHM* in short).

It is a good practice to validate the *DITA map* before executing the transformation scenario. To do so, run *the **Validate and Check for Completeness** action*. Depending on the selected options, this action reports errors, such as topics referenced in other topics (but not in the *DITA map*), broken links, and missing external resources.

However, the execution of the transformation scenario may still fail. Reported errors include the following:

Problem: Cannot Open File

```
[exec] HHC5010: Error: Cannot open "fileName.chm". Compilation stopped.
```

Solution: Cannot Open File

This error occurs when the CHM output file is opened and the transformation scenario cannot rewrite its content. To solve this issue, close the CHM help file and run the transformation scenario again.

Problem: Compilation Failed

```
[exec] HHC5003: Error: Compilation failed while compiling fileName
```

Solution: Compilation Failed

Possible causes of this error are:

- The processed file does not exist. Fix the file reference before executing the transformation scenario again.
- The processed file has a name that contains space characters. To solve the issue, remove any spacing from the file name and run the transformation scenario again.

Related Information:

Compiled HTML Help (CHM) Output Format on page 832

Drag and Drop Without Initial Selection Does Not Work

Problem

When I try to drag with the mouse an unselected file from the ***Project*** *view* or the ***DITA Maps Manager*** *view*, the drag never starts, it only selects the resource. I need to drag the resource again after it becomes selected. As a result any drag and drop without initial selection becomes a two step operation. How can I fix this?

Solution

This is *a bug* present in JVM versions prior to 1.5.0_09. This issue is fixed in 1.5.0_09 and newer versions. See *the installation instructions* for setting a specific JVM version for running the Oxygen XML Editor application.

Gray Window on Linux With the Compiz / Beryl Window Manager

Problem

I try to run Oxygen XML Editor on Linux with the Compiz / Beryl window manager but I get only a gray window that does not respond to user actions. Sometimes the Oxygen XML Editor window responds to user actions but after opening and closing an Oxygen XML Editor dialog or after resizing the Oxygen XML Editor window or a view of the Oxygen XML Editor window the content of this window becomes gray and it does not respond to user actions. What is wrong?

Solution

Sun Microsystems' Java virtual machine *does not support the Compiz window manager and the Beryl one very well*. It is expected that better support for Compiz / Beryl will be added in future versions of their Java virtual machine. You should turn off the special effects of the Compiz / Beryl window manager before starting the Oxygen XML Editor application or switch to other window manager.

Image Appears Stretched Out in the PDF Output

Problem

When publishing XML content (DITA, DocBook, etc.), images are sometimes scaled up in the PDF outputs but are displayed perfectly in the HTML (or WebHelp) output.

Solution

PDF output from XML content is obtained by first obtaining a intermediary XML format called XSL-FO and then applying an XSL-FO processor to it to obtain the PDF. This stretching problem is caused by the fact that all XSL-

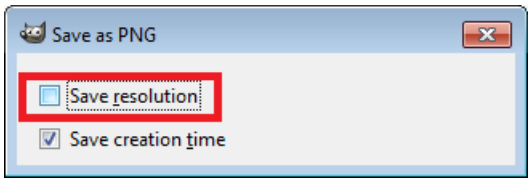
FO processors take into account the DPI (dots-per-inch) resolution when computing the size of the rendered image.

The PDF processor that comes out of the box with the application is the open-source Apache FOP processor. Here is what Apache FOP does when deciding the image size:

1. If the XSL-FO output contains width, height or a scale specified for the image `external-graphic` tag, then these dimensions are used. This means that if in the XML (DITA, DocBook, etc.) you set explicit dimensions to the image they will be used as such in the PDF output.
2. If there are no sizes (width, height or scale) specified on the image XML element, the processor looks at the image resolution information available in the image content. If the image has such a resolution saved in it, the resolution will be used and combined with the image width and height to obtain the rendered image dimensions.
3. If the image does not contain resolution information inside, Apache FOP will look at the FOP configuration file for a default resolution. The FOP configuration file for XSLT transformations that output PDF is located in the *[OXYGEN_INSTALL_DIR]*`/lib/fop.xconf`. DITA publishing uses the DITA Open Toolkit that has the Apache FOP configuration file located in [*DITA-OT-DIR*`/plugins/org.dita.pdf2/fop/conf/fop.xconf`. The configuration file contains two XML elements called `source-resolution` and `target-resolution`. The values set to those elements can be increased (usually a DPI value of 110 or 120 should render the image in PDF the same as in the HTML output).

The commercial **RenderX XEP** XSL-FO processor behaves similarly but as a fallback it uses 120 as the DPI value instead of using a configuration file.

Tip: It is best to save your images without any DPI resolution information. For example, when saving a PNG image in the open-source GIMP image editor, you do not want to save the resolution.



This allows you to control the image resolution from the configuration file for all referenced images.

Increasing the Memory for the Ant Process

Problem

The Ant build process ran out of memory.

Solution

For details about setting custom JVM arguments to the Ant build process see *this section*.

JPEG CMYK Color Space Issues

Problem

JPEG images with CMYK color profile having the color profiles embedded in the image should be properly rendered in the **Author** mode.

Solution

If the color profile information is missing from the JPEG image but you have the ICC file available, you can copy the `profileFileName.icc` to the *[OXYGEN_INSTALL_DIR]*`\lib` directory.

If the color space profile is missing, JPEG images that have the CMYK color space are rendered without taking the color profile into account. The **Unsupported Image Type** message is displayed above the image.

Keyboard Shortcuts Do Not Work

Problem

The keyboard shortcuts listed in the *Menu Shortcut Keys preferences* do not work. What can I do?

Solution

Usually this happens when a special keyboard layout is set in the operating system that generates other characters than the usual ones for the keys of a standard keyboard. For example if you set the extended Greek layout for your keyboard you should return to the default Greek layout or to the English one. Otherwise, the Java virtual machine that runs the application will receive other key codes than the usual ones for a standard keyboard.

Keyboard Language Resets to Default on Windows

Problem

In Windows, I have set a specific language for my keyboard other than the default language and while using Oxygen XML Editor, it keeps getting reset to the default language.

Solution

The default Windows shortcut keys for switching the input language is **Left Alt+Shift** and trying to use various shortcuts in Oxygen XML Editor that involves combinations of those two keys is probably resetting your input language to the default setting if you accidentally press those two keys without a third combination. You can change the Windows shortcut keys that are assigned to the input language by going to the control panel and searching for the **Switch input languages** option. For example, in Windows 10, go to **Control Panel** > **Language** > **Advanced Setting**. In the **Switching input methods** section, click on **Change language bar hot keys**. Click the **Change Key Sequence** button. This opens a dialog box that allows you to switch the shortcut keys for the input language or keyboard layout.

MSXML 4.0 Transformation Issues

Problem

If the latest MSXML 4.0 service pack is not installed on your computer, you are likely to encounter the following error message in the ***Results** panel* when you run a transformation scenario that uses the MSXML 4.0 transformer.

Example Error Message:

```
Could not create the 'MSXML2.DOMDocument.4.0' object.
Make sure that MSXML version 4.0 is correctly installed on the machine.
```

Solution

To fix this issue, go to the Microsoft website and get the latest MSXML 4.0 service pack.

Navigation to the web page was canceled when viewing CHM on a Network Drive

Problem

When viewing a CHM on a network drive, if you only see the TOC and an empty page displaying “Navigation to the web page was canceled” note that this is normal behavior. The Microsoft viewer for CHM does not display the topics for a CHM opened on a network drive.

Solution

As a workaround, copy the CHM file on your local system and view it there.

Out Of Memory Error When Opening Large Documents

Problem

I am trying to open a file larger than 100 MB to edit it in Oxygen XML Editor, but it keeps telling me it runs out of memory (**OutOfMemoryError**). What can I do?

Solution

You should make sure that the minimum limit of document size that enables the support for editing large documents (the value of the ***Optimize loading in the Text edit mode for files over*** *option*) is less than the size of your document. That will enable the optimized support for large documents. If that fails and you still get an Out Of Memory error you should *increase the memory available to Oxygen XML Editor*.

Other tips:

- Make sure that you close other files before opening the large file.
- You can set the default editing mode *in the Preferences dialog box*. The **Text** editing mode uses less memory than other editing modes.
- If the file is too large for the editor to handle, you can *open it in for viewing in* ***Large File Viewer***.

Oxygen XML Editor Crashed on My Mac OS X Computer

Problem

Oxygen XML Editor crashed the Apple Java virtual machine/Oxygen XML Editor could not start up on my OS X computer due to a JVM crash. What can I do?

Solution

Usually it is an incompatibility between the Apple JVM and a native library of the host system. More details are available in the crash log file generated by OS X and reported in the crash error message.

Oxygen XML Editor Takes Several Minutes to Start

Problem

Some anti-virus software can cause Java applications, such as Oxygen XML Editor, to start very slowly due to scanning of compressed archives (such as the *JAR* libraries that all Java applications use).

Solution

A possible solution is to add the Oxygen XML Editor folder to the list of exceptions in the anti-virus software settings.

PDF Processing Fails When Publishing DITA Content

There are cases when publishing DITA content fails when creating a PDF file. This topic lists some common problems and solutions.

Problem: Cannot Save PDF

The FO processor cannot save the PDF at the specified target. The console output contains messages like this:

```
[fop] [ERROR] Anttask - Error rendering fo file:
C:\samples\dita\temp\pdf\oxygen_dita_temp\topic.fo
<Failed to open C:\samples\dita\out\pdf\test.pdf>
Failed to open samples\dita\out\pdf\test.pdf
.............
[fop] Caused by: java.io.FileNotFoundException:
C:\Users\radu_coravu\Desktop\bev\out\pdf\test.pdf
(The process cannot access the file because it is being used by another process)
```

Solution: Cannot Save PDF

Such an error message usually means that the PDF file is already opened in a PDF reader application. The solution is to close the open PDF before running the transformation.

Problem: Table Contains More Cells Than Defined in Colspec

One of the DITA tables contains more cells in a table row than the defined number of *colspec* elements. The console output contains messages like this:

```
[fop] [ERROR] Anttask - Error rendering fo file:
D:\projects\eXml\samples\dita\flowers\temp\pdf\oxygen_dita_temp\topic.fo
<net.sf.saxon.trans.XPathException: org.apache.fop.fo.ValidationException:
The column-number or number of cells in the row overflows the number of
fo:table-columns specified for the table.
(See position 179:-1)>net.sf.saxon.trans.XPathException:
org.apache.fop.fo.ValidationException: The column-number or number of cells
in the row overflows the number of fo:table-columns specified for the table.
(See position 179:-1)
[fop]  at org.apache.fop.tools.anttasks.FOPTaskStarter.renderInputHandler
(Fop.java:657)
[fop]  at net.sf.saxon.event.ContentHandlerProxy.startContent
(ContentHandlerProxy.java:375)
............
[fop] D:\projects\samples\dita\flowers\temp\pdf\oxygen_dita_temp\topic.fo ->
D:\projects\samples\dita\flowers\out\pdf\flowers.pdf
```

Solution: Table Contains More Cells Than Defined in Colspec

To resolve this issue, correct the *colspec* attribute on the table that caused the issue. To locate the table that caused the issue:

1. Edit the transformation scenario and set the parameter *clean.temp* to *no*.
2. Run the transformation, open the `topic.fo` file in Oxygen XML Editor, and look in it at the line specified in the error message `(See position 179:-1)`.
3. Look around that line in the `XSL-FO` file to find relevant text content that you can use (for example, with the **Find/Replace in Files** action in the ***DITA Maps Manager*** *view*) to find the original DITA topic for which the table was generated.

Problem: Broken Link

There is a broken link in the generated `XSL-FO` file. The PDF is generated but contains a link that is not working. The console output contains messages like this:

```
[fop] 1248 WARN [ main ] org.apache.fop.apps.FOUserAgent -
Page 6: Unresolved ID reference "unique_4_Connect_42_wrongID" found.
```

Solution: Broken Link

To resolve this issue:

1. Use the **Validate and Check for Completeness** action available in the ***DITA Maps Manager*** *view* to find such problems.
2. If you publish to PDF using a `DITAVAL` filter, select the same DITAVAL file in the **DITA Map Completeness Check** dialog box.
3. If the **Validate and Check for Completeness** action does not discover any issues, edit the transformation scenario and set the *clean.temp* parameter to *no*.
4. Run the transformation, open the `topic.fo` file in Oxygen XML Editor, and search for the *unresolved ID references* (for example: `unique_4_Connect_42_wrongID`).
5. Look in the `XSL-FO` file to find relevant text content that you can use (for example, with the **Find/Replace in Files** action in the ***DITA Maps Manager*** *view*) to find the original DITA topic for which the table was generated.

Scroll Function of my Notebook Trackpad is Not Working

Problem

I got a new notebook (Lenovo Thinkpad™ with Windows) and noticed that the scroll function of my trackpad is not working in Oxygen XML Editor.

Solution

It is a problem with the Synaptics™ trackpads that can be fixed by adding the following lines to the `C:\Program Files\Synaptics\SynTP\TP4table.dat` file:

```
*,*,oxygen19.0.exe,*,*,*,WheelStd,1,9
*,*,oxygenAuthor19.0.exe,*,*,*,WheelStd,1,9
*,*,oxygenDeveloper19.0.exe,*,*,*,WheelStd,1,9
*,*,syncroSVNClient.exe,*,*,*,WheelStd,1,9
*,*,diffDirs.exe,*,*,*,WheelStd,1,9
*,*,diffFiles.exe,*,*,*,WheelStd,1,9
```

Segmentation Fault Error on Mac OS X

Problem

On my Mac OS X machine the application gives a *Segmentation fault* error when I double-click the application icon. Sometimes it gives no error but it does not start. What is the problem?

Solution

Make sure you have the latest Java update from the Apple website installed on your Mac OS X computer. If installing the latest Java update does not solve the problem, copy the file `JavaApplicationStub` from the `/System/Frameworks/JavaVM.framework` folder to the `Oxygen.app/Contents/MacOS` folder. For browsing the `.app` folder you have to **(CMD+click)** the Oxygen XML Editor icon and select **Show Package Contents**.

Set Specific JVM Version on Mac OS X

Problem

How do I configure Oxygen XML Editor to run with the version X of the Apple Java virtual machine on my Mac OS X computer?

Solution

Oxygen XML Editor uses the first JVM from the list of preferred JVM versions set on your Mac computer that has a version number 1.6.0 or higher. You can move your desired JVM version up in the preferred list by dragging it with the mouse on a higher position in the list of JVMs available in the **Java Preferences** panel that is opened from **Applications** > **Utilities** > **Java** > **Java Preferences**.

Signature Verification Failed Error on a Resource from Documentum

Problem

When I try to open/edit a resource from Documentum, I receive the following error:

`signature verification failed: certificate for All-MB.jar.checksum not signed by a certification authority`.

Solution

The problem is that the certificates from the Java Runtime Environment 1.6.0_22 or later no longer validate the signatures of the UCF *JARS*.

Set the `-Drequire.signed.ucf.jars=false` parameter, as explained in the *Setting a Parameter in the Launcher Configuration File / Startup Script* topic.

Special Characters are Replaced with a Square in Editor

Problem

My file was created with other application and it contains special characters (such as é, ©, ®, etc.) Why does Oxygen XML Editor display a square for these characters when I open the file in Oxygen XML Editor?

Solution

You must set a font able to render the special characters in the ***Font** preferences*. If it is a text file you must set also *the encoding used for non XML files.* If you want to set a font that is installed on your computer but that font is not accessible in the **Font** preferences that means the Java virtual machine is not able to load the system fonts, probably because it is not a True Type font. It is a problem of the Java virtual machine and a possible solution is to copy the font file in the `[JVM_DIR]/lib/fonts` folder. `[JVM_DIR]` is the value of the property *java.home* that is available in the **System properties** tab of the **About** dialog box that is opened from menu **Help** > **About**.

Syntax Highlight Not Available in Eclipse Plugin

Problem

I associated the `.ext` extension with Oxygen XML Editor in Eclipse. Why does an `.ext` file opened with the Oxygen XML Editor plugin not have syntax highlight?

Solution

Associating an extension with Oxygen XML Editor in Eclipse versions 3.6-3.8, 4.2-4.6 requires three steps:

1. Associate the `.ext` extension with the Oxygen XML Editor plugin..
   a. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **General** > **Editors** > **File Associations**.
   b. Add `*.ext` to the list of file types.
   c. Select `*.ext` in the list by clicking it.
   d. Add Oxygen XML Editor to the list of **Associated editors** and make it the default editor..
2. Associate the `.ext` extension with the **Oxygen XML** content type.
   a. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **General** > **Content Types**.
   b. Add `*.ext` to the **File associations** list for the **Text** > **XML** > **Oxygen XML Editor** content type.
3. Press the **OK** button in the Eclipse preferences dialog box.

Result: Now when an `*.ext` file is opened the icon of the editor and the syntax highlight should be the same as for XML files opened with the Oxygen XML Editor plugin.

TocJS Transformation Does not Generate All Files for a Tree-Like TOC

Problem

The *TocJS* transformation of a *DITA map* does not generate all the files needed to display the tree-like table of contents.

Solution

To get a complete set of output files, follow these steps:

1. Run the *XHTML* transformation on the same *DITA map*. Make sure the output gets generated in the same output folder as for the *TocJS* transformation.
2. Copy the content of *`DITA-OT-DIR`*`/plugins/com.sophos.tocjs/basefiles` folder in the transformation output folder.

3. Copy the *DITA-OT-DIR*`/plugins/com.sophos.tocjs/sample/basefiles/frameset.html` file in the transformation's output folder.
4. Edit `frameset.html` file.
5. Locate element `<frame name="contentwin" src="concepts/about.html">`.
6. Replace `"concepts/about.html"` with `"index.html"`.

Topic References Outside the Main DITA Map Folder

Problem

Referencing a DITA topic, *map*, or binary resource (for example, image) that is located outside of the folder where the main *DITA map* is located leads to problems when publishing the content using the DITA Open Toolkit.

Solution

The DITA-OT does not handle it well when references are outside the directory where the published *DITA map* is found. By default, it does not even copy the referenced topics to the output directory.

To solve this, you have the following options:

1. Create another *DITA map* that is located in a folder path above all referenced folders and reference from it the original *DITA map*. Then transform this *DITA map* instead.
2. Edit the transformation scenario and in the **Parameters** tab edit the **fix.external.refs.com.oxygenxml** parameter. This parameter is used to specify whether or not the application tries to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references. The fix has no impact on your edited DITA content. The allowed values are "false" and "true". The default value is false.

   Important: The **fix.external.refs.com.oxygenxml** parameter is only supported when the DITA OT transformation process is started from Oxygen XML Editor.

Wrong Highlights of Matched Words in a Search in User Manual

Problem

When I do a keyword search in the User Manual that is included with the Oxygen XML Editor application, the search highlights the wrong word in the text. Sometimes the highlighted word is several words after the matched word. What can I do to get correct highlights?

Solution

This does not happen when Oxygen XML Editor runs with a built-in Java virtual machine, that is a JVM that was installed by Oxygen XML Editor in a subfolder of the installation folder (for example, on Windows and Linux when installing Oxygen XML Editor with the installation wizard specific for that platform). When Oxygen XML Editor runs from an All Platforms installation, it uses whatever JVM was found on the host system, which may be incompatible with the JavaHelp indexer used for creating the built-in User Manual. Such a JVM may offset the highlight of the matched word with several characters, usually to the right of the matched word. To see the highlight exactly on the matched word, it is recommended to install the application with the specific installation wizard for your platform (available only for Windows and Linux).

XML Document Takes a Long Time to Open

Problem

Oxygen XML Editor takes a long time to open a document.

Solution

It takes longer to open an XML document if the whole content of your document is on a single line or if the document size is very large.

If the content is on a single line, you can speed up loading by selecting the ***Format and indent the document on open*** *option* (in the **Format** preferences page).

If the document is very large (above 30 MB), make sure that the value of the ***Optimize loading in the Text edit mode for files over*** *option* (in the **Open/Save** preferences page) is greater than the size of your document.

If that fails and you get an Out Of Memory error (**OutOfMemoryError**) you can *increase the memory available to Oxygen XML Editor.*

XSLT Debugger Is Very Slow

Problem

When I run a transformation in the **XSLT Debugger** *perspective* it is very slow. Can I increase the speed?

Solution

If the transformation produces HTML or XHTML output you should *disable rendering of output in the XHTML output view* during the transformation process. To view the XHTML output result do one of the following:

- Run the transformation in the **Editor** *perspective* and make sure the ***Open in Browser/System Application*** *option* is selected.
- Run the transformation in the **XSLT Debugger** *perspective*, save the text output area to a file, and use a browser application for viewing it (for example Firefox or Internet Explorer).

DITA Authoring and Publishing

19

Topics:

- *Working with DITA Maps*
- *Working with DITA Topics*
- *Working with Markdown Documents in DITA*
- *Working with Keys*
- *Reusing DITA Content*
- *Linking in DITA*
- *Publishing DITA Output*
- *DITA Profiling / Conditional Text*
- *DITA Open Toolkit Support*
- *DITA Specialization Support*
- *Master Files Support in DITA*
- *Metadata*
- *Migrating MS Office Documents to DITA*
- *DITA 1.3 Support*

DITA is an XML standard, an architectural approach, and a writing methodology, developed by technical communicators for technical communicators. It provides a standardised architectural *framework* for a common structure for content that promotes the consistent creation, sharing, and re-use of content.

Some of the benefits of using DITA include the following:

- **Flexibility** - DITA is a topic-based architecture and it offers flexibility in content organization.
- **Modularity** - DITA allows for content reuse that saves time and reduces the number of modifications.
- **Structured Authoring** - DITA offers a standardized, methodological approach that helps to reduce authoring time and improve consistency.
- **Single-Source Publishing** - DITA provides the ability to change content in one place and have the change propagate everywhere.
- **Multiple Output Formats** - DITA supports multiple types of output.
- **Inheritance** - The DITA inheritance model makes it easy to specialize topics or elements within topics and you only have to define how the element is different from its immediate ancestor.
- **Process Automation** - DITA offers various ways to automate processes, such as with index or glossary production, output delivery, validation, and more.
- **Specialization** - DITA allows you to define your own information types and semantic elements/attributes to suit the needs of your particular content model.
- **Multi-Lingual** - DITA is a translation-friendly structure that supports numerous languages and text encodings.
- **Conditional Profiling** - DITA supports conditional text processing and profiling to filter content in the publishing stage.

This chapter is designed to be a guide to help content authors who use DITA. It also presents the Oxygen XML Editor features that are specific to working with DITA documents and concepts.

DITA Resources

For more general information and technical details about working with DITA, refer to the following resources:

- *The DITA Specifications*.
- *The DITA Style Guide Best Practices for Authors*.

- Various sample DITA topics and maps can be found in the *[OXYGEN_INSTALL_DIR]*/samples/dita folder.

  Related Information:
  DITA Topics Document Type (Framework) on page 794
  DITA Map Document Type (Framework) on page 804

Working with DITA Maps

In the DITA standard architecture you create documents by collecting topics into maps.

DITA Maps

A *DITA map* organizes a set of topics into a hierarchy. In most output formats, the structure of the map becomes the structure of the table of contents. Oxygen XML Editor provides support for *creating* and *managing DITA maps* through the ***DITA Maps Manager***. There are also specialized types of *DITA maps*, such as a *bookmap*, which is intended for creating the structure of a book.

Submaps

You do not have to create an entire publication using a single map. It is generally good practice to break up a large publication into several smaller *submaps* that are easier to manage. You can reuse submaps in multiple publications by including them in each of the main maps. The ***DITA Maps Manager*** provides support for easily creating and managing submaps.

Opening a DITA Map

There are several ways to open a *DITA map* and you can choose to open it in the ***DITA Maps Manager*** or in the XML Editor. Use any of the following methods to open a map:

- To open a submap in its own tab in the **DITA Maps Manager**, simply double-click it (or right-click it and select **Open**).
- To open a map in the XML editor from the **DITA Maps Manager**, right-click it and select **Open Map in Editor**.
- Drag a *DITA map* file from your system browser and drop it in the XML editor. This will open the map in the editor.
- If you open a file with the `.ditamap` or `.bookmap` extension (from the ***Project*** *view* or a system browser), a dialog box is opened that offers you the choice of opening it in the XML editor or in the **DITA Maps Manager**.

  Note: If you select the **Do not show the dialog again** option, it will always be opened in the method that you choose and you will not be asked in the future. However, you can reset this by selecting **Always ask** for the ***When opening a map*** *option in the* ***DITA*** *preferences page*.
- To open a map in the **DITA Maps Manager**, you can right-click a map file in the ***Project*** *view* and select **Open with** > **DITA Maps Manager**.
- If you have a *DITA map* file open in the XML editor, you can open it in the **DITA Maps Manager** by right-clicking the title tab and selecting **Open in DITA Maps Manager View**.

Chunking DITA Maps

By default, many output types place a single topic on each output page. In some cases you may want to *output multiple topics as a single output page (also known as chunking)*. To support this, Oxygen XML Editor provides an ***Edit Properties*** *dialog box* that allows you to easily configure the attributes of a topic to control how your table of contents and topics are rendered in the output.

Validating a Map

You should *validate your maps* to make sure that the individual topics are valid and that the relationships between them are working. Oxygen XML Editor provides a validation function for *DITA maps* that performs a comprehensive validation of a map and its topics.

To watch a video on DITA editing , go to *https://www.oxygenxml.com/demo/DITA_Editing.html*.

Related Information:

DITA Map Document Type (Framework) on page 804
DITA Map Author Mode Actions on page 805

DITA Maps Manager

Oxygen XML Editor provides a view for managing and editing *DITA maps*. The **DITA Maps Manager** view presents a *DITA map* as a Table of Contents. It allows you to navigate the topics and maps, make changes, and apply transformation scenarios to obtain various output formats. By default, it is located to the left of the main editor. If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu.

The **DITA Maps Manager** includes a variety of useful actions to help you edit and organize the structure of your *DITA maps* and topics. The actions that are available and their functions depend on the type of nodes that are selected in the **DITA Maps Manager**. If you select multiple sibling nodes, the result of the actions will be applied to all the selected nodes. If you select multiple nodes that are not on the same hierarchical level, the actions will be applied to the parent node and the child nodes will inherit certain attributes from the parent node.

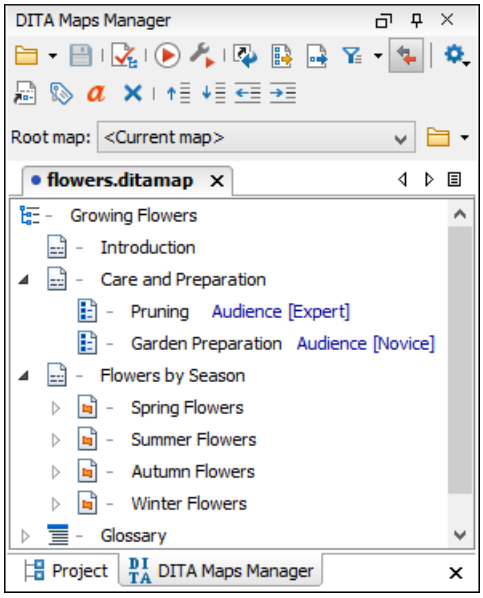


Figure 694: DITA Maps Manager View

Opening Maps in the DITA Maps Manager

The **DITA Maps Manager** view supports multiple open maps at the same time, with each one presented in its own tab. To open a *DITA map* in the **DITA Maps Manager**, use any of the following methods:

- To open a submap in its own tab, simply double-click it (or right-click it and select **Open**).
- If you open a file with a `.ditamap` or `.bookmap` extension (from the ***Project** view* or a system browser), a dialog box is opened that offers you the choice of opening it in the **DITA Maps Manager** or the XML editor.

  Note: If you select the **Do not show the dialog again** option, it will always be opened in the method that you choose and you will not be asked in the future. However, you can reset this by selecting **Always ask** for the ***When opening a map** option in the **DITA** preferences page*.
- Right-click a map file in the ***Project** view* and select **Open with** > **DITA Maps Manager**.
- If you have a *DITA map* file open in the XML editor, you can right-click the title tab and select **Open in DITA Maps Manager View**.

If your map references other *DITA maps*, they will be shown, expanded, in the **DITA Maps Manager** view and you will be able to navigate their content. To edit the submaps and their content, you need to open each referenced map separately.

Drag and Drop in the DITA Maps Manager

You can move topics or nodes within the same map, or other maps, by dragging and dropping them into the desired position. You can arrange the nodes by dragging and dropping one or more nodes at a time. You can arrange multiple topics by dragging them while pressing the **Ctrl** or **Shift** key. Drop operations can be performed before, after, or as child of the targeted node.

Drag and drop operations include:

Copy

Select the nodes you want to copy and start dragging them. Before dropping them in the appropriate place, press and hold the **Ctrl** key. The mouse pointer changes to a symbol to indicate that a copy operation is being performed.

Move

Select the nodes you want to move and drag and drop them in the appropriate place.

Promote (Ctrl + Alt + LeftArrow (Command + Alt + LeftArrow on OS X))/Demote (Ctrl + Alt + RightArrow (Command + Alt + RightArrow on OS X))

You can move nodes between child and parent nodes by using the **Promote** (**Ctrl + Alt + LeftArrow (Command + Alt + LeftArrow on OS X)**) and **Demote** (**Ctrl + Alt + RightArrow (Command + Alt + RightArrow on OS X)**) operations.

DITA Maps Manager Toolbar

The toolbar includes the following actions (also available in the**DITA Maps** menu) and their availability depend on the nodes that are selected:

Note: If multiple nodes are selected, the availability of the actions depends on the nodes that are selected.

Open Drop-down Menu

You can use this drop-down menu to open new *DITA maps* or to reopen recently viewed maps. The drop-down menu contains the following:

- List of recently viewed *DITA maps* that can be selected to reopen them.
- **Clear history** - Clears the history list of the recently viewed *DITA maps*.
- **Open** - Allows you to open the map in the ***DITA Maps Manager*** *view*. You can also open a map by dragging it from the file system explorer and dropping it into the ***DITA Maps Manager*** *view*.
- **Open URL** - Displays the **Open URL** dialog box that allows you to access any resource identified through a URL (defined by a protocol, host, resource path, and an optional port). The following actions are available in this drop-down menu:
  - **Browse for local file** - Opens a local file browser dialog box, allowing you to select a local *DITA map*.
  - **Browse for remote file** - Displays the ***Open URL*** *dialog box* that allows you to open a remotely stored *DITA map*.
  - **Browse for archived file** - Displays the ***Archive Browser*** that allows you to browse the content of an archive and choose a *DITA map*.
  - **Browse Data Source Explorer** - Opens the ***Data Source Explorer*** that allows you to browse the data sources defined in the *Data Sources preferences page*.

    Tip: You can open the **Data Sources** preferences page by using the **Configure Database Sources** shortcut from the **Open URL** dialog box.
  - **Search for file** - Displays the ***Open/Find Resource*** *dialog box* that allows you to search for a *DITA map*.

Save (Ctrl (Meta on Mac OS)+S)

Saves the current *DITA map*.

Validate and Check for Completeness

Checks the validity and integrity of the map.

Apply Transformation Scenario(s)

Applies the DITA Map transformation scenario that is associated with the current map.

Configure Transformation Scenario(s)

Opens the ***Configure Transformation Scenarios(s)*** *dialog box* where you can edit or create transformation scenarios or *associate a DITA Map transformation scenario* with the current map.

Refresh References

You can use this action to manually trigger a refresh and update of all referenced documents. This action is useful when the referenced documents are modified externally. When they are modified and saved from Oxygen XML Editor, the *DITA map* is updated automatically.

Open Map in Editor with Resolved Topics

Opens the *DITA map* in the main editor area with content from all topic references, expanded in-place. Content from the referenced topics is presented as read-only and you have to use the contextual menu action **Edit Reference** to open the topic for editing.

Tip: If you want to print the expanded content, you should consider changing the **Styles** drop-down to **+ Print ready**.

Open Map in Editor

For complex operations that cannot be performed in the simplified **DITA Maps Manager** view (for instance, editing a relationship table) you can open the map in the main editing area.

Note: You can also use this action to open referenced *DITA maps* in the **Editor**.

Profiling/Conditional Text Drop-down Menu

This drop-down menu contains the following actions:

- **Show Profiling Colors and Styles** - Select this option to turn on conditional styling. To configure the colors and styles *open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Editor** > **Edit modes** > **Author** > **Profiling/Conditional Text** > **Colors and Styles**.
- **Show Profiling Attributes** - Select this option to display the values of the profiling attributes at the end of the titles of topic references. When selected, the values of the profiling attributes are displayed in both the **DITA Maps Manager** view and in the **Author** view.
- **Show Excluded Content** - Controls if the content filtered out by a particular condition set is hidden or grayed-out in the editor area and in the ***Outline*** and **DITA Maps Manager** views. When this option is selected, the content filtered by the currently applied condition set is grayed-out. To show only the content that matches the currently applied condition set, deselect this option.
- **Profiling Settings** - Opens the preferences page for adding and editing the profiling conditions that you can apply in the **DITA Maps Manager** view and the **Author** mode editing pane. When a *profiling condition set* is applied, the keys that are defined in the *DITA map* are gathered by filtering out the excluded content.

Link with Editor

Toggles the synchronization between the file path of the current editor and the selected topic reference in the **DITA Maps Manager** view.

Note: This button is toggled off automatically when you move to a **Debugger** *perspective*.

Settings

Show extended toolbar

Toggles whether or not the extended toolbar will be displayed in the **DITA Maps Manager** toolbar.

Show root map toolbar

Toggles whether or not the ***Root map*** *option* will be displayed in the **DITA Maps Manager** toolbar.

Show topic titles

Toggles how topics are presented in the **DITA Maps Manager**. If selected, the title of each topic is shown. Otherwise, the file path (value of the `href` attribute) for each topic is shown.

Root map

Specifies a master *DITA map* (*root map*) that Oxygen XML Editor uses to define a hierarchical structures of submaps and to establish a *key space* that defines the keys that are propagated throughout the entire map structure. For more information, see *Selecting a Root Map* on page 1591.

Browse Drop-down menu

You can use this drop-down menu to browse for *root maps* with the following choices:

- **Browse for local file** - Opens a local file browser dialog box, allowing you to select a local *root map*.
- **Browse for remote file** - Displays the ***Open URL*** *dialog box* that allows you to select a remotely stored *root map*.
- **Browse for archived file** - Displays the ***Archive Browser*** that allows you to browse the content of an archive and choose a *root map*.
- **Browse Data Source Explorer** - Opens the ***Data Source Explorer*** that allows you to browse the data sources defined in the *Data Sources preferences page*.

  Tip: You can open the **Data Sources** preferences page by using the **Configure Database Sources** shortcut from the **Open URL** dialog box.
- **Search for file** - Displays the ***Find Resource*** *dialog box* to search for a *root map*.

The following additional actions are displayed in the toolbar when the **Show extended toolbar** option is selected in the **Settings** menu:

Insert Topic Reference

Opens *the* ***Insert Reference*** *dialog box* that allows you to insert references to targets such as topics, maps, topic sets, or key definitions.

Edit Properties

Opens the **Edit Properties** dialog box that allows you to configure the properties of a selected node. For more details about this dialog box, see *Edit Properties Dialog Box* on page 1603.

Edit Attributes

Opens a small in-place editor that allows you to edit the attributes of a selected node. You can find more details about this action in the *Attributes View in Author Mode* on page 224 topic.

Delete

Deletes the selected node.

Move Up

Moves the selected node up within the *DITA map* tree.

Move Down

Moves the selected node down within the *DITA map* tree.

Promote(Alt + LeftArrow)

Moves the selected node up one level to the level of its parent node.

Demote(Alt + RightArrow)

Moves the selected node down one level to the level of its child nodes.

Contextual Menu of the DITA Maps Manager

Root Map

The following actions can be invoked from the contextual menu on the *root map* of an opened *DITA map*:

Open Map in Editor

For complex operations that cannot be performed in the simplified **DITA Maps Manager** view (for instance, editing a relationship table) you can open the map in the main editing area.

Open Map in Editor with Resolved Topics

Opens the *DITA map* in the main editor area with content from all topic references, expanded in-place. Content from the referenced topics is presented as read-only and you have to use the contextual menu action **Edit Reference** to open the topic for editing.

Export DITA Map

Allows you to choose a destination for exporting the *DITA map*.

Find Unreferenced Resources

Allows you to search for orphaned resources that are not referenced in the *DITA maps*.

Edit Properties

Opens the **Edit Properties** dialog box that allows you to configure the properties of a selected node. For more details about this dialog box, see *Edit Properties Dialog Box* on page 1603.

Fast Create Topics

Opens the ***Fast Create Topics*** *dialog box* that allows you to quickly create multiple skeleton topics at once and you can specify their hierarchical structure within the *DITA map*.

Append Child submenu

Container sub-menu for a number of actions that create a map node as a child of the currently selected node:

- **New** - Opens a dialog box that allows you to configure some options for *inserting a new topic*.
- **Reference** - Inserts a reference to a topic file. You can find more details about this action in the *Inserting References* topic.
- **Reference to the currently edited file** - Inserts a reference to the currently edited file. You can find more details about this action in the *Inserting References* topic.
- **Key Reference** - Opens an ***Insert Key Definition*** *dialog box* that allows you to insert a key reference.
- **Key Reference with Keyword** - Opens a simplified ***Insert Key Definition*** *dialog box* that allows you to define a key and a value inside a *keyword*.
- A set of actions that open the ***Insert Reference*** *dialog box* that allow you to insert various reference specializations (such as **Anchor Reference**, **Glossary Reference**, **Map Reference**, **Navigation Reference**, **Topic Group**, **Topic Head**, **Topic Reference**, **Topic Set**, **Topic Set Reference**).

Search References

Searches all references to the current topic in the entire *DITA map*.

Refactoring submenu

The following actions are available from this submenu:

Convert Markdown to DITA Topic (Available for Markdown documents)

Opens a dialog box that allows you to configure options for *converting the Markdown document into a DITA topic*.

Rename resource

Allows you to *change the name of a resource linked in the edited DITA map*.

Move resource

Allows you to *change the location on disk of a resource linked in the edited DITA map*.

XML Refactoring

Opens *the XML Refactoring tool wizard* that presents refactoring operations to assist you with managing the structure of your XML documents.

Other XML Refactoring Actions

For your convenience, the last 5 *XML Refactoring tool operations* that were finished or previewed will also appear in this submenu.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace content across multiple files.

Check Spelling in Files

Allows you to *spell check multiple files*.

Paste

Allows you to paste content from the clipboard into the *DITA map*.

Paste Before

Pastes the content of the clipboard (only if it is a part of the *DITA map*) before the currently selected *DITA map* node.

Paste After

Pastes the content of the clipboard (only if it is a part of the *DITA map*) after the currently selected *DITA map* node.

Expand All

Allows you to expand the entire *DITA map* structure.

Collapse All

Allows you to collapse the entire *DITA map* structure.

Child Nodes

The following actions are available when the contextual menu is invoked on a child node of a *DITA map* (submaps need to be opened in the **DITA Maps Manager** to access these actions since they are in a read-only state in the parent map):

Note: If multiple nodes are selected, the availability of the actions depends on the nodes that are selected.

Open

Opens in the editor the resources referenced by the nodes that you select.

Edit Properties

Opens the **Edit Properties** dialog box that allows you to configure the properties of a selected node. For more details about this dialog box, see *Edit Properties Dialog Box* on page 1603.

Fast Create Topics

Opens the ***Fast Create Topics*** *dialog box* that allows you to quickly create multiple skeleton topics at once and you can specify their hierarchical structure within the *DITA map*.

Append Child submenu

Container sub-menu for a number of actions that create a map node as a child of the currently selected node:

- **New** - Opens a dialog box that allows you to configure some options for *inserting a new topic*.
- **Reference** - Inserts a reference to a topic file. You can find more details about this action in the *Inserting References* topic.
- **Reference to the currently edited file** - Inserts a reference to the currently edited file. You can find more details about this action in the *Inserting References* topic.
- **Key Reference** - Opens an ***Insert Key Definition*** *dialog box* that allows you to insert a key reference.
- **Key Reference with Keyword** - Opens a simplified ***Insert Key Definition*** *dialog box* that allows you to define a key and a value inside a *keyword*.

- A set of actions that open the ***Insert Reference*** *dialog box* that allow you to insert various reference specializations (such as **Anchor Reference**, **Glossary Reference**, **Map Reference**, **Navigation Reference**, **Topic Group**, **Topic Head**, **Topic Reference**, **Topic Set**, **Topic Set Reference**).

Insert Before submenu

Container sub-menus for a number of actions that create a map node as a sibling of the currently selected node, above the current node in the map:

- **New** - Opens a dialog box that allows you to configure some options for *inserting a new topic*.
- **Reference** - Inserts a reference to a topic file. You can find more details about this action in the *Inserting References* topic.
- **Reference to the currently edited file** - Inserts a reference to the currently edited file. You can find more details about this action in the *Inserting References* topic.
- **Key Reference** - Opens an ***Insert Key Definition*** *dialog box* that allows you to insert a key reference.
- **Key Reference with Keyword** - Opens a simplified ***Insert Key Definition*** *dialog box* that allows you to define a key and a value inside a *keyword*.
- A set of actions that open the ***Insert Reference*** *dialog box* that allow you to insert various reference specializations (such as **Anchor Reference**, **Glossary Reference**, **Map Reference**, **Navigation Reference**, **Topic Group**, **Topic Head**, **Topic Reference**, **Topic Set**, **Topic Set Reference**).

Insert After submenu

Container sub-menus for a number of actions that create a map node as a sibling of the currently selected node, below the current node in the map:

- **New** - Opens a dialog box that allows you to configure some options for *inserting a new topic*.
- **Reference** - Inserts a reference to a topic file. You can find more details about this action in the *Inserting References* topic.
- **Reference to the currently edited file** - Inserts a reference to the currently edited file. You can find more details about this action in the *Inserting References* topic.
- **Key Reference** - Opens an ***Insert Key Definition*** *dialog box* that allows you to insert a key reference.
- **Key Reference with Keyword** - Opens a simplified ***Insert Key Definition*** *dialog box* that allows you to define a key and a value inside a *keyword*.
- A set of actions that open the ***Insert Reference*** *dialog box* that allow you to insert various reference specializations (such as **Anchor Reference**, **Glossary Reference**, **Map Reference**, **Navigation Reference**, **Topic Group**, **Topic Head**, **Topic Reference**, **Topic Set**, **Topic Set Reference**).

Search References

Searches all references to the current topic in the entire *DITA map*.

Refactoring submenu

The following actions are available from this submenu:

Convert Markdown to DITA Topic (Available for Markdown documents)

Opens a dialog box that allows you to configure options for *converting the Markdown document into a DITA topic*.

Rename resource

Allows you to *change the name of a resource linked in the edited DITA map*.

Move resource

Allows you to *change the location on disk of a resource linked in the edited DITA map*.

XML Refactoring

Opens *the XML Refactoring tool wizard* that presents refactoring operations to assist you with managing the structure of your XML documents.

Other XML Refactoring Actions

For your convenience, the last 5 *XML Refactoring tool operations* that were finished or previewed will also appear in this submenu.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace content across multiple files.

Check Spelling in Files

Allows you to *spell check multiple files*.

Cut

Deletes the currently selected node and copies it to the clipboard.

Copy

Copies the currently selected node to the clipboard.

Paste

Allows you to paste content from the clipboard into the *DITA map*.

Paste Before

Pastes the content of the clipboard (only if it is a part of the *DITA map*) before the currently selected *DITA map* node.

Paste After

Pastes the content of the clipboard (only if it is a part of the *DITA map*) after the currently selected *DITA map* node.

Delete

Deletes the currently selected node from the *DITA map*.

Organize

Allows you to organize the *DITA map* with the several submenu actions:

- **Move Up** - Moves the selected node up within the *DITA map* tree.
- **Move Down** - Moves the selected node down within the *DITA map* tree.
- **Promote**(**Alt + LeftArrow**) - Moves the selected node up one level to the level of its parent node.
- **Demote**(**Alt + RightArrow**) - Moves the selected node down one level to the level of its child nodes.

Expand All

Allows you to expand the entire *DITA map* structure.

Collapse All

Allows you to collapse the entire *DITA map* structure.

Other Nodes

The following actions are available when the contextual menu is invoked from a *map* node that is not an immediate child node of the *root map* or other special nodes (such as relationship tables):

Note: If multiple nodes are selected, the availability of the actions depends on the nodes that are selected.

Open

Opens in the editor the resources referenced by the nodes that you select.

Open Map in Editor (available when invoking on a submap)

Opens the currently selected *DITA map* in the editor.

Open parent DITA map (available when invoking on a topic reference or a submap reference)

Opens the parent *DITA map* of the currently selected reference in the **DITA Maps Manager**.

Edit Attributes (only available for relationship table nodes)

Opens a small in-place editor that allows you to edit the attributes of a selected node. You can find more details about this action in the *Attributes View in Author Mode* on page 224 topic.

Edit Profiling Attributes (only available for relationship table nodes)

Allows you to change the *profiling attributes* defined on the selected node.

Search References

Searches all references to the current topic in the entire *DITA map*.

Refactoring submenu

The following actions are available from this submenu:

Convert Markdown to DITA Topic (Available for Markdown documents)

Opens a dialog box that allows you to configure options for *converting the Markdown document into a DITA topic*.

Rename resource

Allows you to *change the name of a resource linked in the edited DITA map*.

Move resource

Allows you to *change the location on disk of a resource linked in the edited DITA map*.

XML Refactoring

Opens *the XML Refactoring tool wizard* that presents refactoring operations to assist you with managing the structure of your XML documents.

Other XML Refactoring Actions

For your convenience, the last 5 *XML Refactoring tool operations* that were finished or previewed will also appear in this submenu.

Find/Replace in Files

Opens the ***Find/Replace in Files*** *dialog box* that allows you to find and replace content across multiple files.

Check Spelling in Files

Allows you to *spell check multiple files*.

Copy

Copies the currently selected node to the clipboard.

Expand All

Allows you to expand the entire *DITA map* structure.

Collapse All

Allows you to collapse the entire *DITA map* structure.

To watch our video demonstration about the **DITA Maps Manager** view, go to *https://www.oxygenxml.com/demo/DITA_Maps_Manager.html*.

Related Information:

DITA Map Validation and Completeness Check on page 1609
DITA Map Author Mode Actions on page 805
Finding and Replacing Text in Multiple Files on page 476

Creating a Map

To create a *DITA map*, *subject scheme map*, *bookmap*, or other types of *DITA maps*, follow these steps:

1. Go to **File** > **New**. If you want the *map* to be a submap, you can create it the same way by right-clicking the place in the current map where you want to add it (in the ***DITA Maps Manager***) and selecting **New** from the **Append Child**, **Insert Before**, or **Insert After** submenu.
   A **New** document dialog box is opened that allows you to select a document type from various folders.
2. Select one of the **DITA Map** templates from the **Framework templates** folder.
3. Click the **Create** button.
4. Select whether you want to open the map in the ***DITA Maps Manager*** or the **Editor**.

5. Save the map using the **Save** button on the toolbar of the ***DITA Maps Manager*** *view*.

Selecting a Root Map

Oxygen XML Editor allows you to select a *root map* (a master *DITA map*) that defines a hierarchical structures of submaps and establishes a *key space* that defines the keys used in all the other *DITA maps* and topics in the project. Specifying the correct *root map* helps to prevent validation problems when you work with `keyrefs` and also acts as the foundation for content completion. All the *keys* that are defined in a *root map* are available in the submaps that are contained within the *root map*.

There are several ways to select or change the *root map*:

- The easiest method is to use the ***Root map*** *drop-down menu* in the ***DITA Maps Manager*** toolbar to select the appropriate *root map*.
- If you insert a *key reference* using the **Cross Reference** action from the **Link** drop-down menu (from the toolbar or **Link** submenu of the contextual menu) and keys are not gathered from the expected *DITA map*, you can change the *root map* by using the **Change Root Map** link in the **Choose Key** dialog box that is opened when you click the **Choose Key Reference** button.
- If you insert a *content key reference* or *key reference* using the **Reuse Content** action (from the toolbar, **DITA** menu, or **Reuse** submenu of the contextual menu) and keys are not gathered from the expected *DITA map*, you can change the *root map* by using the **Change Root Map** link in the **Choose Key** dialog box that is opened when you click the **Choose Key Reference** button.

To watch our video demonstration about the DITA *root map* support, go to *https://www.oxygenxml.com/demo/DITA_Root_Map.html*.

Creating DITA Submaps

You can break up a large *DITA map* into more manageable pieces by creating submaps. A submap is simply a *DITA map* that is included by another *DITA map*. There is no separate markup for a submap.

For example, if you are creating a book, you might use one submap for each chapter of the book. If you are reusing a set of topics in multiple publications, you might collect them into a map and reuse the map as a submap in multiple other maps, rather than referencing the topics individually from the new maps.

You add a submap to a map the same way that you would *add a new topic or insert an exiting topic into a map*, except you choose a map rather than a topic to create or add. When adding a submap to a map make sure that you use a `mapref` element or a `topicref` element with the `format` attribute set to `ditamap`. In most cases, Oxygen XML Editor takes care of this for you.

Adding a Submap to a Map

To add a submap to a map:

1. Right-click the place in the current map where you want to add the new submap.
2. To insert the submap as a child of the selected node, select **Append Child** > **New**. To insert the submap as a sibling to the current node, select **Insert After** > **New** or **Insert Before** > **New**.

   Step Result: This opens a ***New*** *file template dialog box* that allows you to select the type of document and assists you with naming it.
3. Select the type of map in one of the folders inside the **DITA Map** folder and give it a name (the file type should be `.ditamap`).
4. Click **Create** to insert the submap.

You can also manage and move submaps the same as you would with topics. For more information, see *Managing DITA Maps* on page 1592.

Creating a Bookmap in DITA

If you want to create a traditional book in DITA, you can use a *bookmap* to organize your topics into a book. A DITA *bookmap* is a specialized type of map, intended for creating output that is structured like a book. A *bookmap* allows you to add book-specific elements such as `frontmatter`, `part`, `chapter`, `appendix`, and `backmatter`

to the map. How these book-specific elements are processed for publication is up to the processing script for each media. See the *DITA documentation* for details.

You can find additional support for creating books in DITA in the DITA for Publishers plugin, which is included with Oxygen XML Editor.

To create a book in DITA using a *bookmap*, follow these steps:

1. Create a new *bookmap* (**File** > **New** > **Framework templates** > **DITA Map** > **map** > **Bookmap**). If you want the *bookmap* to be a submap, you can create it the same way by right-clicking the place in the current map where you want to add it (in the ***DITA Maps Manager***) and selecting **New** from the **Append Child**, **Insert Before**, or **Insert After** submenus.
2. Create the structure of your book by adding the appropriate book sections and defining containers for chapters and any appendices. To add sections to a *bookmap*, or children to a section, right-click the *bookmap* or section icon and choose any of the reference actions in the **Append child** menu. The selections offered in the menu will adjust depending on the element they are applied to. Consult the *DITA documentation* to fully understand the structure of a DITA *bookmap* and where to create each element.
3. Create special elements such as an *index* and *table of contents*. The index and table of contents will be generated by the build process, based on the content of the map and the topics it points to.
4. *Add topics* to your chapters to add content to your book. You may find it easier to manage if you *use submaps* to create the content of your chapters. This keeps your *bookmap* from becoming long and difficult to manage.

Managing DITA Maps

You may want to manage your *DITA maps* in a variety of ways, including:

- Change the order and nesting of topics in a map.
- Add topics to a map.
- Insert various types of references in a map.
- Find, move, or rename resources in a map.
- Change other properties of the items in a map.
- Use the Edit Properties dialog box to manage attributes, keys, metadata, or add profiling to any section of a map.

This section includes various topics that describe how you can manage *DITA maps* and resources.

Change the Order of Topics in DITA Maps

You can change the order and nesting of the topics in a map in several ways:

- By dragging and dropping topics within the ***DITA Maps Manager***.
- By highlighting a topic in the ***DITA Maps Manager***, holding down the **Alt** key, and pressing the arrow keys.
- By showing the extended ***DITA Maps Manager*** toolbar (press the **Settings** icon on the ***DITA Maps Manager*** toolbar and select the extended toolbar) and then using the node moving buttons ( ) on the toolbar to move topics around in the map.

To understand how to organize topics in a *DITA map* using the ***DITA Maps Manager***, you can examine and experiment with the sample map called `flowers.ditamap`, located in the *`[OXYGEN_INSTALL_DIR]`*`/samples/dita` folder.

Adding Topics to a DITA Map

When you are working in DITA, there are several approaches that you can use to create topics and maps. You can start by first creating topics and then assembling your finished topics into one or more documents by creating one or more maps, or you can start by creating a map and then adding new topics to it as you work.

The topics-first approach is generally more appropriate if you intend to do a lot of content reuse, as it encourages you to think of each topic as an independent unit that can be combined with other topics in various ways. The map-first approach will be more familiar to you if you are used to creating books or manuals as a whole. Oxygen XML Editor supports both approaches.

A *DITA map* organizes content hierarchically, so you can add a topic as a child of the map root element or as a child or sibling of any item already in the map. Therefore, the first step to adding a topic to a map is always to choose the place it will be inserted into the map.

Adding Existing Topics to a Map

At the XML-level, a topic is added to a map by adding a reference to the map that points to the topic. There are a variety of reference types that you can use. The default type is the `topicref` element. See the *DITA documentation* for the full range of reference elements and their uses. Oxygen XML Editor provides several tools for inserting reference elements into a map:

Using the Insert Reference Dialog Box

The ***Insert Reference** dialog box* allows you to create various reference types and configure the most commonly used attributes. You can open the **Insert Reference** dialog box with any of the following methods:

- Right-click an item in the current map where you want to add the reference, select **Append Child**, **Insert Before**, or **Insert After** and select the type of reference to enter.
- If the topic you want to add is currently open in the editor, you can right-click an item in the current map where you want to add the reference and select **Reference to the currently edited file**.
- Selecting an item in the map and click the **Insert Reference** button from the ***DITA Maps Manager*** toolbar.
- Select **Insert Reference** from the **DITA Maps** menu.

Dragging and Dropping a File into the DITA Maps Manager

You can add a topic to a *DITA map* by dragging and dropping the file into the ***DITA Maps Manager***. You can drag and drop files from any of the following:

- Your OS file system explorer.
- The ***Project** view*.
- The ***Open/Find Resource** view*.

Adding topics this way will not open the **Insert Reference** dialog box, but you can adjust all the same properties by invoking the contextual menu from the topic and selecting **Edit Properties**.

Adding a New Topic to a Map

To add a new topic to a map, follow these steps:

1. In the ***DITA Maps Manager***, right-click the node in the current map where you want to add the new topic.
2. Select one of the following actions:
   - **Append Child** > **New** - Select this action to insert the new topic as a child of the selected node. This action opens a ***New** file dialog box* that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click **Create**.
   - **Insert Before** > **New** - Select this action to insert the new topic as a sibling to the current node, before it. This action opens a ***New** file dialog box* that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click **Create**.
   - **Insert After** > **New** - Select this action to insert the new topic as a sibling to the current node, after it. This action opens a ***New** file dialog box* that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click **Create**.
   - **Duplicate** - Select this action to create a copy of the selected topic and insert it as a sibling. This action opens a dialog box that allows you to choose the file name and location for the newly created copy of the topic. After you have selected the name and path for your new topic, click **OK**.

   Step Result: The new topic is now referenced (as a `topicref`) in the *DITA map* at the location where you inserted it and the new topic is opened in the editor.
3. Save the *DITA map*.

Adding Multiple Skeleton Topics at Once

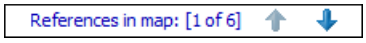
Oxygen XML Editor includes a feature in the ***DITA Maps Manager*** that allows you to quickly create multiple skeleton topics at once and you can specify their hierarchical structure within the *DITA map*.

To access this feature, right-click a node in the **DITA Maps Manager** where you want the new topics to be inserted and select **Fast Create Topics**. This opens the **Fast Create Topics** dialog box where you can configure the structure for the new topics.

For more information, see *Fast Create Multiple DITA Topics* on page 1620.

Adding Multiple References to the Same Topic in a Map

Oxygen XML Editor allows you to reuse entire topics by adding multiple references to the same topic in a *DITA map*. Whenever multiple references to the same topic are detected in the context of the current map in the ***DITA Maps Manager***, an indicator will appear in the top-right corner of the **Author** mode editor that shows the number of times the topic is referenced in the *DITA map*. It also includes navigation arrows that allow you to jump to the next or previous reference.



Remove Topics from a Map

You can remove topics from a map in a number of ways. Some ways to remove a topic from a map include:

- Highlight the topic and press **Delete** or **Backspace**.
- Highlight the topic and click the ✖ **Delete** button on the ***DITA Maps Manager*** extended toolbar.

Related Information:

Fast Create Multiple DITA Topics on page 1620

Moving and Renaming Resources

You can move or rename resources on disk directly from Oxygen XML Editor and you have the option of updating all the references to the moved or renamed resources. For DITA resources (such as topics and maps), you can do this from the ***DITA Maps Manager*** *view*. For non-DITA resources (such as folders, images, html files, audio, video, text files, Markdown documents), you can do this from the ***Project*** *view*.

Moving or Renaming DITA Resources (Topics or Maps)

To move or rename normal DITA resources (such as topics or maps), use one of the following actions available in the **Refactoring** submenu of the contextual menu when invoked on the resource in the ***DITA Maps Manager*** *view*:

Refactoring > Move resource

This action allows you to change the location of a resource linked in the edited *DITA map*, using the **Move resource** dialog box. This dialog box contains the following options:

- **Destination** - Specifies the target location of the edited resource.
- **File name** - Allows you to change the name of the edited resource.
- **Update references** - Select this checkbox to update all references of the file in the edited *DITA map* and in the files referenced from the *DITA map*, preserving the *completeness* of the *DITA map*.
- **Preview** - Select this button to display a preview of the changes Oxygen XML Editor is about to make.
- **Move** - Moves the edited resource in the target location on disk.
- **Cancel** - Cancels the **Move resource** operation. No changes are applied.

Refactoring > Rename resource

This action allows you to change the name of a resource linked in the edited *DITA map*, using the **Rename resource** dialog box. This dialog box contains the following options:

- **New name** - Presents the current name and allows you to change it.
- **Update references** - Select this checkbox to update all references of the file in the edited *DITA map* and in the files referenced from the *DITA map*, preserving the *completeness* of the *DITA map*.

- **Preview** - Select this button to display a preview of the changes Oxygen XML Editor is about to make.
- **Rename** - Executes the **Rename resource** operation.
- **Cancel** - Cancels the **Rename resource** operation. No changes are applied.

Note: If a *root DITA map* is not defined, the move and rename actions are executed in the context of the current *DITA map*.

Moving or Renaming Non-DITA Resources and Updating the References to Them

To move or rename non-DITA resources (such as folders, images, html files, audio, video, text files, Markdown documents), you can simply follow the procedures described in *Moving/Renaming Resources in the Project View* on page 283. However, this approach will not give you the option to update the references to the moved or renamed resources.

To perform move or rename operation on non-DITA resources while also updating all the references to them, use the following sets of procedures:

1. Enable *Master Files* support and add your *root DITA map* to the *Master Files* folder by following the procedure found here: *How to Enable Master Files Support in DITA* on page 1717.
2. Move or rename resources and update the references to them by following the procedure found here: *Moving or Renaming Non-DITA Resources and Updating the References to Them* on page 1717.

Related Information:

Master Files Support in DITA on page 1716

Finding Resources Not Referenced in DITA Maps on page 1595

Finding Resources Not Referenced in DITA Maps

Over the course of time large projects can accumulate a vast amount of resources from a variety of sources. Especially in organizations with a large number of content authors or complex project structures, organizing the project resources can become a challenge. Over time a variety of actions can cause resources to become orphaned from *DITA maps*. To assist you with organizing project resources, Oxygen XML Editor includes an action, **Find Unreferenced Resources**, that searches for orphaned resources that are not referenced in *DITA maps*.

To perform this search, open the *DITA map* in the ***DITA Maps Manager***, invoke the contextual menu on the *DITA map*, and select **Find Unreferenced Resources**. This action can also be selected from the **DITA Maps** menu. This action opens the **Find Unreferenced Resources** dialog box, which allows you to specify some search parameters:

- **DITA Maps** - Provides a list of *DITA maps* to be included in the search and allows you to **Add** maps to the list or **Remove** them.
- **Folders** - Provides a list of folders to be included in the search and allows you to **Add** or **Remove** specific folders.
- **Filters** - Provides three combo boxes that allow you to filter the search to include or exclude certain files or folders:
  - **Include files** - Allows you to filter specific files to include in the search.
  - **Exclude files** - Allows you to filter specific files to exclude from the search.
  - **Exclude folders** - Allows you filter specific folders to exclude from the search.

    Note: In any of the filter combo boxes you can enter multiple filters by separating them with a comma and you can use the *?* and *** wildcards. Use the drop-down arrow to select a previously used filter pattern.

Inserting References in DITA Maps

A *DITA map* may contain various types of references. The targets of the references can be a variety of references, such as chapters, maps, topics, topic sets, or key definitions. You can insert references to such targets with the ***Insert Reference*** *dialog box*.

This section explains how to insert and configure references (such as topic references, topic groups, topic headings, and key definitions) in a *DITA map*.

Insert Reference Dialog Box

The **Insert Reference** dialog box allows you to insert and configure references in *DITA maps*. There are numerous types of references that can be inserted into maps. They include references to topics, other maps, glossary terms, and keys. You can also use this dialog box to configure the attributes of a reference, add profiling or metadata, and define keys.

To open the **Insert Reference** dialog box, use one of the following methods:

- Select **Reference**, **Reference to the currently edited file**, or any of the other specific reference actions that are available from the **Append Child**, **Insert Before**, and **Insert After** submenus when invoking the contextual menu in the ***DITA Maps Manager***.
  - To insert the reference as a child of the current node, select the reference from the **Append Child** submenu.
  - To insert the reference as a sibling of the current node, below the current node in the map, select the reference from the **Insert After** submenu.
  - To insert the reference as a sibling of the current node, above the current node in the map, select the reference from the **Insert Before** submenu.

  Note: The content of these submenus depends on the node that is selected in the *DITA map* tree when the contextual menu is invoked. For example, if the selected node is a topic reference (`topicref`), its possible child nodes include the following elements: `anchorref`, `chapter`, `keydef`, `mapref`, `topicgroup`, `topichead`, `topicref`, `topicset`, and `topicsetref`.
- Click the **Insert Reference** button on the **DITA Maps Manager** extended toolbar. This action will insert the reference as a sibling of the current node (below the current node in the map).
- Select **Insert Reference** from the **DITA Maps** menu. This action will insert the reference as a sibling of the current node (below the current node in the map).

For the **Reference** or **Reference to the currently edited file** actions, a **Reference type** drop-down list is displayed at the top of the **Insert Reference** dialog box and you can select the type of reference you want to insert. Depending on the place where the reference will be inserted, Oxygen XML Editor will propose only valid reference types . When you change the reference type, the fields in the various tabs of the dialog box are reconfigured depending upon the availability of the associated attributes. For the other reference actions in the **Append Child**, **Insert Before**, and **Insert After** submenus, the reference type is automatically chosen based upon the invoked action and you cannot change it.

The main section of the dialog box includes the following tabs: **Target**, **Keys**, **Attributes**, **Metadata**, and **Profiling**.

Target Tab

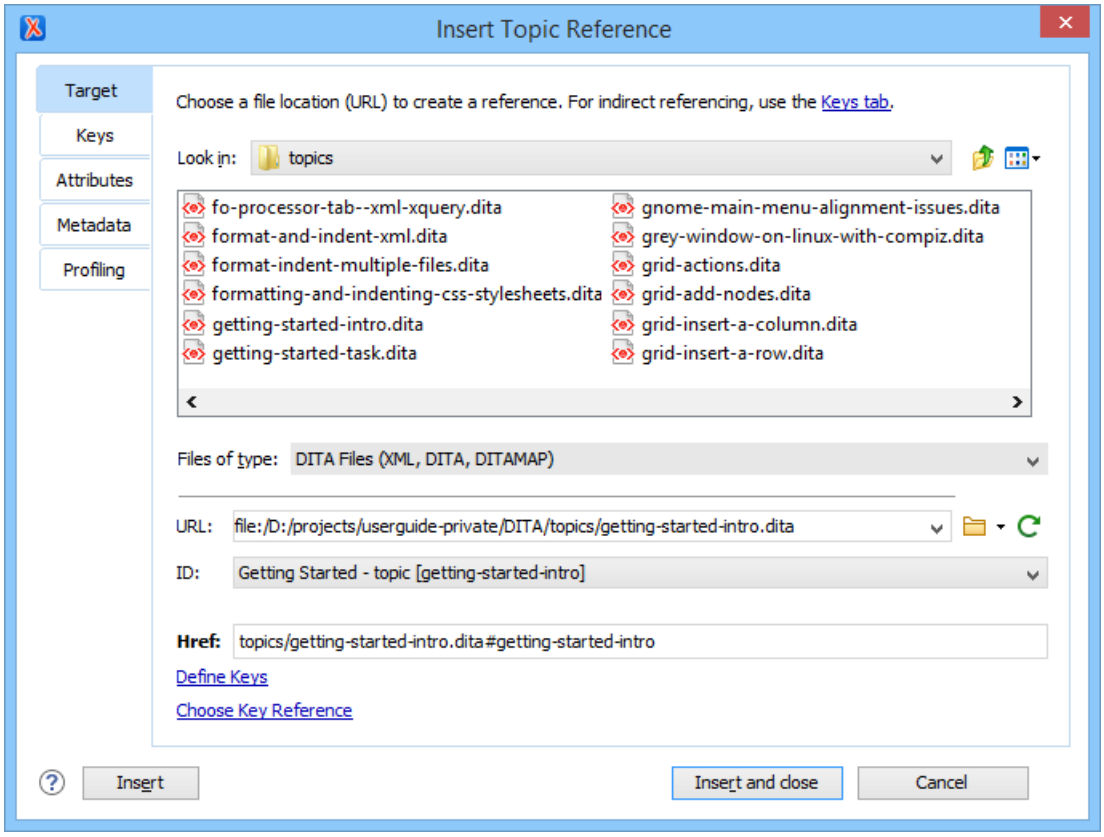


Figure 695: Insert Reference Dialog Box - Target Tab

The **Target** tab of the **Insert Reference** dialog box allows you to specify information about the target reference. It includes the following sections and fields:

Choose a file location section

You can browse for and select the source target file by using the **Look in** drop-down list, browsing tools, or file window in this section. You can use the **Files of type** drop-down menu to narrow the list of possible file types that will be displayed.

URL

Displays the path to the target and allows you to select or change it by using the combo box or browsing tools.

ID

The drop-down list displays all of the target elements that are available for the selected target URL.

Href

The selected target automatically modifies this value to point to the corresponding `href` attribute of the target element.

Note: If the **Reference type** is a **Navigation Reference**, the **Href** field is changed to **Mapref**, since a `navref` element requires a `mapref` attribute instead.

Keys Tab

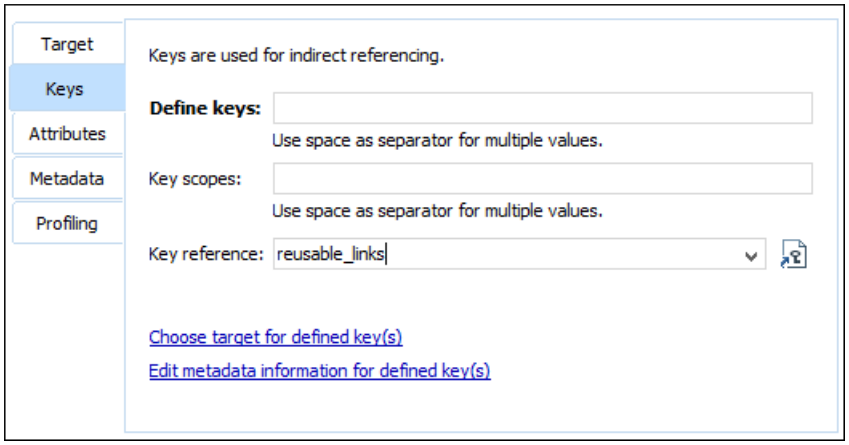


Figure 696: Insert Reference Dialog Box - Keys Tab

The **Keys** tab allows you to use and *define keys* for indirect referencing. For more information, see *Working with Keys* on page 1652. This tab includes the following:

Define keys

Use this text field to define the `keys` attribute for the target.

Key scopes [This option is only available if the *Built-in DITA-OT 2.x (with DITA 1.3 support)* option is selected in the DITA preferences page]

Use this text field to define or edit the value of a *`keyscope` attribute*. *Key scopes* allow you to specify different sets of key definitions for different map branches.

Key reference

Instead of using the **Target** tab to select a file that contains the target reference, you can reference a key definition by using this text field. Use the **Choose key reference** button to access the list of keys that are already defined in the current *root map*.

Attributes Tab

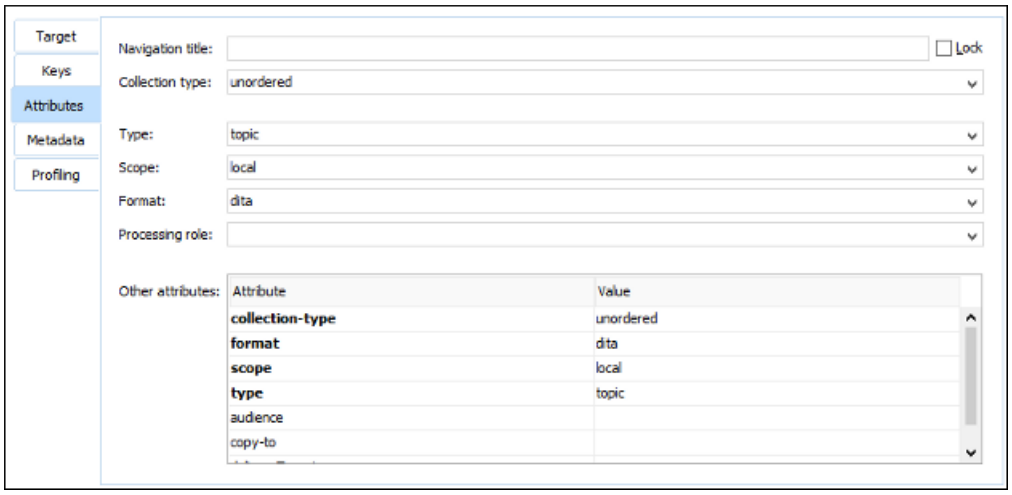


Figure 697: Insert Reference Dialog Box - Attributes Tab

The **Attributes** tab of the **Insert Reference** dialog box allows you to insert and edit attribute values for the target reference. This tab includes the following sections and actions:

Navigation title

This text field allows you to specify a custom navigation title for the target reference. If you want this attribute to always be populated with a detected value (based on the specifications for the target file), select the **Navigation title** checkbox for the ***Always fill values for attributes*** *option in the* ***DITA*** *preferences page*. You can enforce the use of the specified title by selecting the **Lock** checkbox.

Tip: You can also select the ***Prefer navigation title for topicref rendering*** *option in the* ***DITA*** *preferences page* to always enforce the use of the `navtitle` value rather than selecting this **Lock** option on individual topics.

Collection type

This drop-down list allows you to select the `collection-type` attribute to create hierarchical linking between topics in a *DITA map* (for example, `unordered`, `sequence`, `choice`, `family`, `-dita-use-conref-target`).

Type

Allows you to select a `type` attribute (such as `topic`, `task`, `concept`, etc.) for the target element. If you want this attribute to always be populated with a detected value (based on the specifications for the target file), select the **Type** checkbox for the ***Always fill values for attributes*** *option in the* ***DITA*** *preferences page*.

Scope

This property corresponds to the `scope` attribute of the target element. It is populated automatically, based on the selected file type, unless its value for the selected target file is the same as the default attribute value. If you want this attribute to always be populated with a detected value based on the specifications (regardless of the default value), select the **Scope** checkbox for the ***Always fill values for attributes*** *option in the* ***DITA*** *preferences page*.

Format

This property corresponds to the `format` attribute of the target element. It is populated automatically, based on the selected file type, unless its value for the selected target file is the same as the default attribute value. If you want this attribute to always be populated with a detected value based on the specifications (regardless of the default value), select the **Format** checkbox for the ***Always fill values for attributes*** *option in the* ***DITA*** *preferences page*.

Processing Role

This drop-down list allows you to set the `processing-role` attribute to one of the allowed values for DITA reference elements (for example, `resource-only`, `normal`, `-dita-use-conref-target`).

Other attributes table

This table contains the attributes that are available for the selected reference. You can use this table to insert or edit the values of any of the listed attributes. Clicking a cell in the **Value** column allows you to use the combo box to enter, edit, or select attribute values.

Metadata Tab

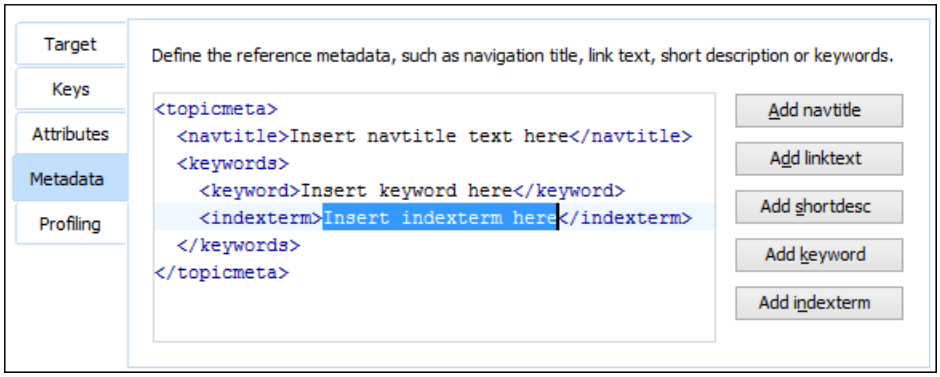


Figure 698: Insert Reference Dialog Box - Metadata Tab

The **Metadata** tab allows you to add metadata elements to the target reference. Use the buttons on the right side of the tab to insert specific metadata elements (you can add the following metadata elements: `navtitle`,

`linktext`, `shortdesc`, `keyword`, `indexterm`). The metadata elements are inserted inside a `topicmeta` element. The editing window allows you to easily insert and modify the content of the metadata that will be inserted.

Profiling Tab

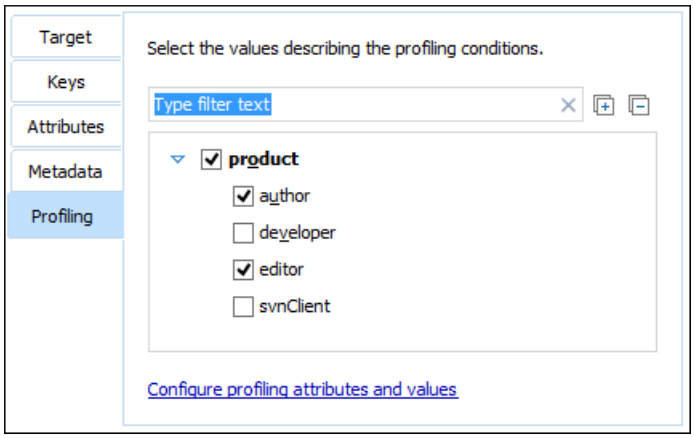


Figure 699: Insert Reference Dialog Box - Profiling Tab

The **Profiling** tab allows you to select or change profiling attributes for the selected reference. This tab displays profiling attributes and their values as determined by the following:

- If your *root map* references a DITA *subject scheme map* that defines values for the profiling attributes, those values are used.
- If your project defines *project-level* configuration values for the *profiling attributes*, those values are used.
- If Oxygen XML Editor defines *global-level* configuration values for the *profiling attributes*, they are used.
- Otherwise, a basic default set of profiling attributes and values are used.

When you modify a selection of values in this tab, the change will also automatically be reflected in the **Attributes** tab. For more information, see *DITA Profiling / Conditional Text* on page 1698.

Finalizing Your Insert Reference Configuration

Once you click **Insert** or **Insert and close**, the configured reference is added in the map.

Tip: You can easily insert multiple references by keeping the **Insert Reference** dialog box opened, using the **Insert** button.

Related Information:

DITA Profiling / Conditional Text on page 1698
Working with Keys on page 1652

Inserting Topic Headings

The `topichead` element provides a title-only entry in a navigation map, as an alternative to the fully-linked title provided by the `topicref` element.

You can insert a topic heading by doing the following:

- Select **Topic Head** from the **Append Child**, **Insert Before**, or **Insert After** submenus when invoking the contextual menu in the ***DITA Maps Manager*** *view*.
- *Open the DITA map in the XML editor* and select the **Insert Topic Heading** action from the main toolbar (or from the **Insert** submenu of the contextual menu).

Those actions open the ***Insert Topic Head*** *dialog box* that allows you to easily insert a `topichead` element. A **Navigation title** (`navtitle` attribute) is required but other attributes can also be specified from this dialog box (such as **Type**, **Scope**, **Format**, etc.)

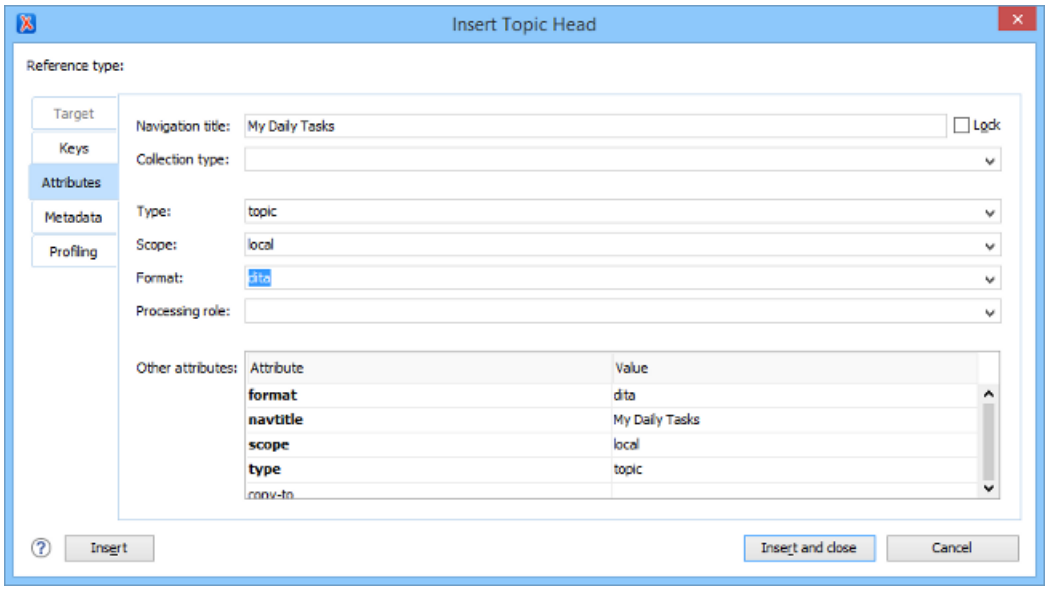


Figure 700: Insert Topic Heading Dialog Box

Related Information:

Insert Reference Dialog Box on page 1596

Inserting Topic Groups

The `topicgroup` element identifies a group of topics (such as a concepts, tasks, or references) or other resources. A `topicgroup` can contain other `topicgroup` elements, allowing you to express navigation or table-of-contents hierarchies, as well as implying relationships between the containing `topicgroup` and its children. You can set the collection-type of a container `topicgroup` to determine how its children are related to each other. Relationships end up expressed as links in the output (with each participant in a relationship having links to the other participants by default).

You can insert a topic group by doing the following:

- Select **Topic Group** from the **Append Child**, **Insert Before**, or **Insert After** submenus when invoking the contextual menu in the ***DITA Maps Manager*** *view*.
- *Open the DITA map in the XML editor* and select the **Insert Topic Group** action from the main toolbar (or from the **Insert** submenu of the contextual menu).

Those actions open the ***Insert Topic Group*** *dialog box* that allows you to easily insert a `topicgroup` element and various attributes can be specified (such as **Collection type**, **Type**, **Scope**, **Format**, etc.)

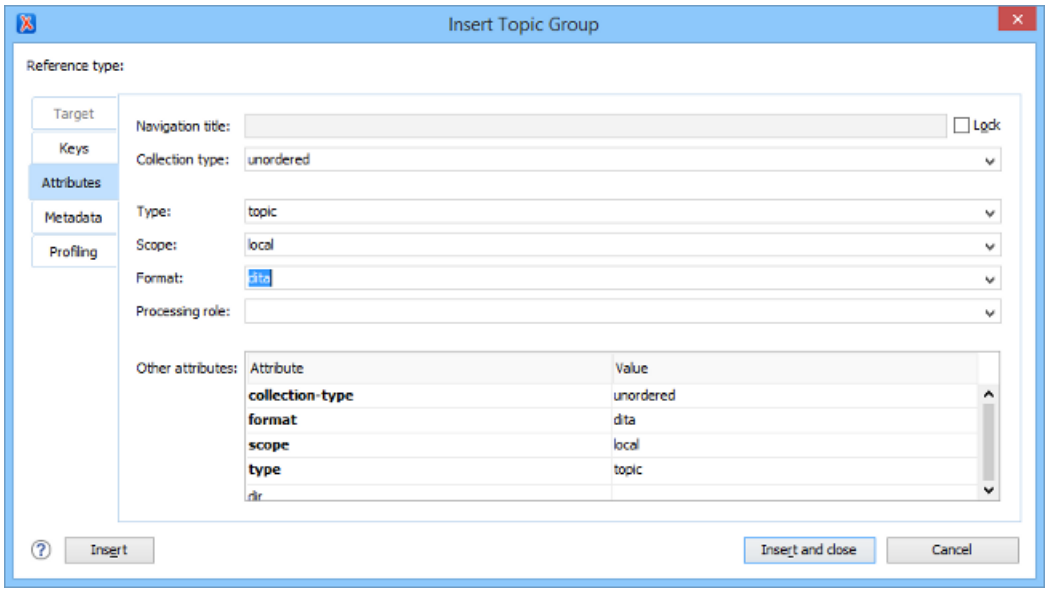


Figure 701: Insert Topic Group Dialog Box

Related Information:

Insert Reference Dialog Box on page 1596

Defining Keys in DITA Maps

DITA uses *keys* to insert content that may have different values in various circumstances. Keys provide the means for indirect referencing in DITA. This can make it easier to manage and to reuse content. In DITA, keys are defined in maps and can then be reused and referenced throughout the whole structure of the map.

The following example is a *DITA map* that defines various values for the `product` key:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA Map//EN" "map.dtd">
<map>
  <!-- product name -->
  <keydef keys="product" product="basic">
    <topicmeta>
      <keywords>
        <keyword>Basic Widget</keyword>
      </keywords>
    </topicmeta>
  </keydef>
  <keydef keys="product" product="pro">
    <topicmeta>
      <keywords>
        <keyword>Professional Widget</keyword>
      </keywords>
    </topicmeta>
  </keydef>
  <keydef keys="product" product="enterprise">
    <topicmeta>
      <keywords>
        <keyword>Enterprise Widget</keyword>
      </keywords>
    </topicmeta>
  </keydef>
</map>
```

Note: The profiling of the names is now contained in the map, where it only has to occur once to reuse throughout the whole map structure.

Key Definition with a Keyword

To insert a *key definition with a keyword* in a *DITA map*, follow these steps:

1. Open the *DITA map* in the ***DITA Maps Manager***.
2. Invoke the contextual menu and select **Key Definition** from the **Append Child**, **Insert Before**, or **Insert After** submenu (depending on where you want to insert the *key definition* in the *DITA map*). This opens an **Insert Key Definition** dialog box.

3. Go to the **Keys** tab and enter the name of the key in the **Define keys** field.
4. Go to the **Metadata** tab and click **Add keyword**. In the editing window enter the key value inside the `keyword` element.

   Note: You can profile the key by using the **Profiling** tab and other attributes can also be defined in the **Attributes** tab.
5. Once you are done configuring the *key definition*, click **Insert and close**.

Alternatively, there is a simplified method that can be used if you simply want to define a key with a value inside a *keyword*, without configuring any profiling or other attributes. To use the simplified method, follow these steps:

1. Open the *DITA map* in the ***DITA Maps Manager***.
2. Invoke the contextual menu and select **Key Definition with Keyword** from the **Append Child**, **Insert Before**, or **Insert After** submenu (depending on where you want to insert the *key definition* in the *DITA map*). This opens a simplified **Insert Key Definition** dialog box.
3. Enter the name of the key in the **Key** field and its value in the **Keyword** field.
4. Click **Insert and close** to finalize the operation.

Key Definition with a Target

To insert a *targeted key definition* (for example, to target a resource such as an image or topic) in a *DITA map*, follow these steps:

1. Open the *DITA map* in the ***DITA Maps Manager***.
2. Invoke the contextual menu and select **Key Definition** from the **Append Child**, **Insert Before**, or **Insert After** submenu (depending on where you want to insert the *key definition* in the *DITA map*). This opens an **Insert Key Definition** dialog box.
3. Go to the **Keys** tab and enter the name of the key in the **Define keys** field.
4. Go to the **Target** tab and select a target resource (such as an image or topic).

   Note: You can profile the key by using the **Profiling** tab and other attributes can also be defined in the **Attributes** tab.
5. Once you are done configuring the *targeted key definition*, click **Insert and close**.

Related Information:

Working with Variable Text in DITA on page 1668
Working with Keys on page 1652

Edit Properties Dialog Box

The ***DITA Maps Manager*** *view* includes a feature that allows you to view and edit the properties of a selected node. The **Edit properties** action is available on both the **DITA Maps Manager** toolbar and in the contextual menu. This action is also available in the contextual menu when you edit a *DITA map* document in **Author** mode. The action opens the **Edit Properties** dialog box and it includes several tabs with various functions and fields that are initialized with values based upon the node for which the action was invoked.

Note: If you select multiple sibling nodes and invoke the **Edit properties** action, only the **Profiling** tab will be available and your modifications in that tab will be applied to all the selected nodes. If you select multiple nodes that are not on the same hierarchical level, the other tabs will also be available and your modifications will be applied to the parent node (the child nodes will inherit the attributes of the parent node).

You can use the **Edit Properties** dialog box to modify or define attributes, metadata, profiling, or keys in *DITA maps* or topics. You can also use it to modify the title of *root maps*.

At the top of the **Edit Properties** dialog box, the **Reference type** drop-down list displays the type of the selected node and it depends on the node for which the action was invoked.

The main section of the dialog box includes the following tabs: **Target**, **Keys**, **Attributes**, **Metadata**, and **Profiling**. The availability of the tabs and their functions depend on the selected node. For example, if you invoke the action on a *root map*, only the **Attributes**, **Metadata**, and **Profiling** tabs are accessible and the **Title** property can be configured. Also, if you select multiple nodes, only the **Profiling** tab is available.

Target Tab

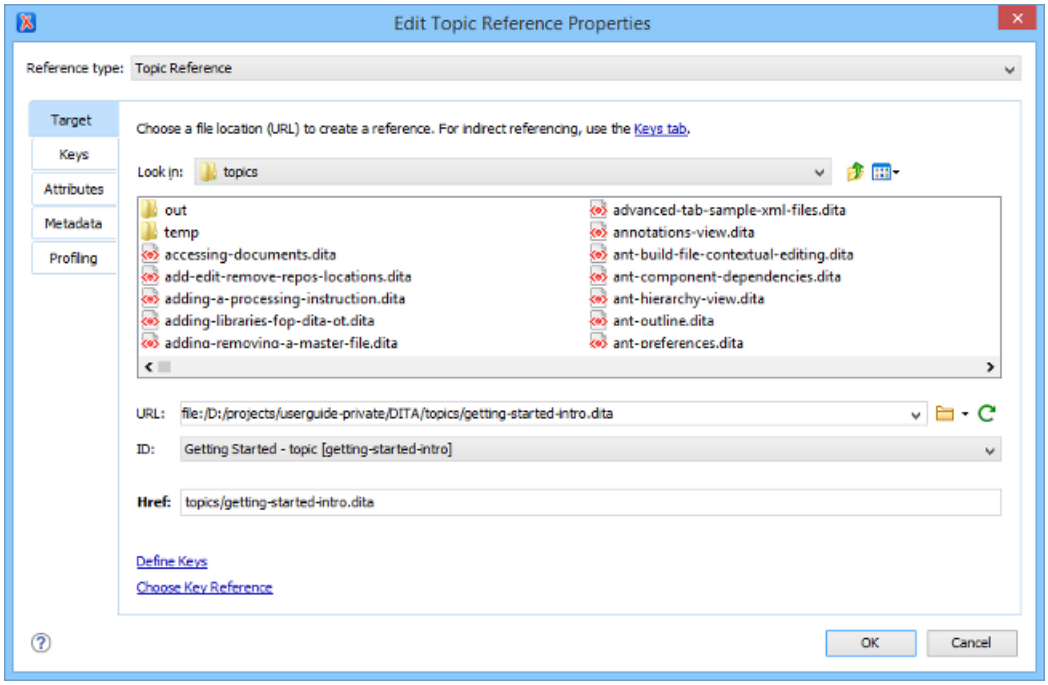


Figure 702: Edit Properties Dialog Box - Target Tab

The **Target** tab of the **Edit Properties** dialog box displays information about the target node on which the action was invoked and allows you to change the target. It includes the following sections and fields:

Choose a file location section

You can browse for and select the source target file by using the **Look in** drop-down list, browsing tools, or file window in this section. You can use the **Files of type** drop-down menu to narrow the list of possible file types that will be displayed.

URL

Displays the path to the target and allows you to select or change it by using the combo box or browsing tools.

ID

The drop-down list displays all of the target elements that are available for the selected target URL.

Href

The selected target automatically modifies this value to point to the corresponding `href` attribute of the target element.

Note: If the **Reference type** is a **Navigation Reference**, the **Href** field is changed to **Mapref**, since a `navref` element requires a `mapref` attribute instead.

Keys Tab

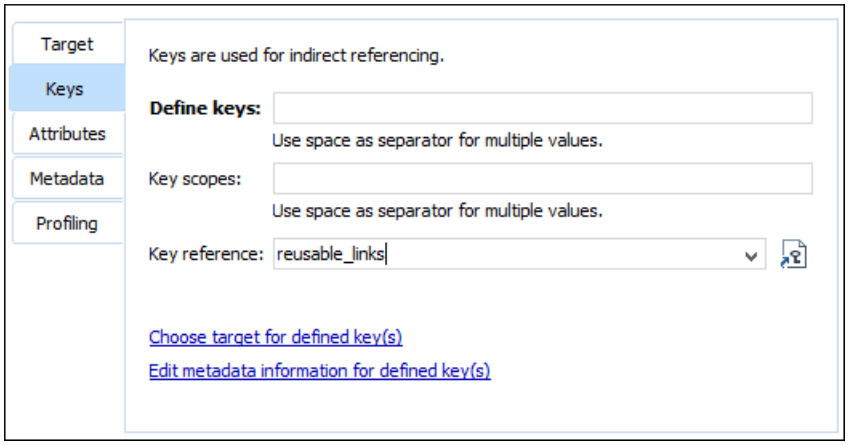


Figure 703: Edit Properties Dialog Box - Keys Tab

The **Keys** tab allows you to use and *define keys* for indirect referencing. For more information, see *Working with Keys* on page 1652. This tab includes the following:

Define keys

Use this text field to define the `keys` attribute for the target.

Key scopes [This option is only available if the *Built-in DITA-OT 2.x (with DITA 1.3 support)* option is selected in the DITA preferences page]

Use this text field to define or edit the value of a *`keyscope` attribute*. *Key scopes* allow you to specify different sets of key definitions for different map branches.

Key reference

Use this combo box (or the **Choose key reference** button) to select a key that is already defined in the *root map*.

Attributes Tab

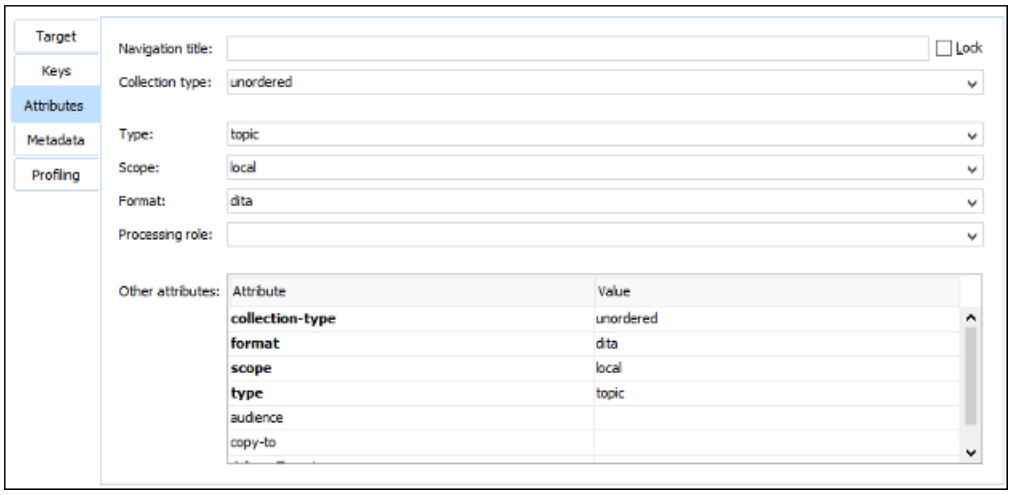


Figure 704: Edit Properties Dialog Box - Attributes Tab

The **Attributes** tab of the **Edit Properties** dialog box allows you to insert and edit attribute values for the target node for which the action was invoked.

If the target is a *root map*, the tab displays the title of the map. You can change it in the **Title** text field and assign it to an **Attribute**, **Element**, or **All**.

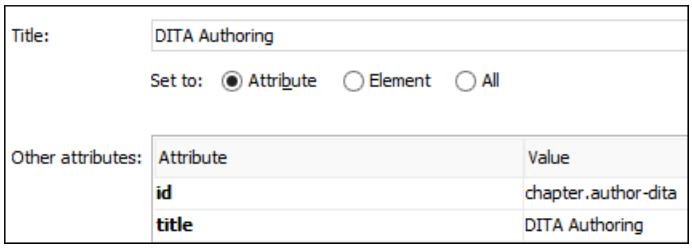


Figure 705: Attributes Tab for a Root Map

For other types of targets, the tab includes the following sections and fields that can be used to edit the attributes of the target:

Navigation title

This text field allows you to specify a custom navigation title for the target reference. If you want this attribute to always be populated with a detected value (based on the specifications for the target file), select the **Navigation title** checkbox for the ***Always fill values for attributes*** *option in the* ***DITA*** *preferences page*. You can enforce the use of the specified title by selecting the **Lock** checkbox.

Tip: You can also select the ***Prefer navigation title for topicref rendering*** *option in the* ***DITA*** *preferences page* to always enforce the use of the `navtitle` value rather than selecting this **Lock** option on individual topics.

Collection type

This drop-down list allows you to select the `collection-type` attribute to create hierarchical linking between topics in a *DITA map* (for example, `unordered`, `sequence`, `choice`, `family`, `-dita-use-conref-target`).

Type

Allows you to select a `type` attribute (such as `topic`, `task`, `concept`, etc.) for the target element. If you want this attribute to always be populated with a detected value (based on the specifications for the target file), select the **Type** checkbox for the ***Always fill values for attributes*** *option in the* ***DITA*** *preferences page*.

Scope

This property corresponds to the `scope` attribute of the target element. It is populated automatically, based on the selected file type, unless its value for the selected target file is the same as the default attribute value. If you want this attribute to always be populated with a detected value based on the specifications (regardless of the default value), select the **Scope** checkbox for the ***Always fill values for attributes*** *option in the* ***DITA*** *preferences page*.

Format

This property corresponds to the `format` attribute of the target element. It is populated automatically, based on the selected file type, unless its value for the selected target file is the same as the default attribute value. If you want this attribute to always be populated with a detected value based on the specifications (regardless of the default value), select the **Format** checkbox for the ***Always fill values for attributes*** *option in the* ***DITA*** *preferences page*.

Processing Role

This drop-down list allows you to set the `processing-role` attribute to one of the allowed values for DITA reference elements (for example, `resource-only`, `normal`, `-dita-use-conref-target`).

Other attributes table

This table contains the attributes that are available for the selected reference. You can use this table to insert or edit the values of any of the listed attributes. Clicking a cell in the **Value** column allows you to use the combo box to enter, edit, or select attribute values.

Metadata Tab

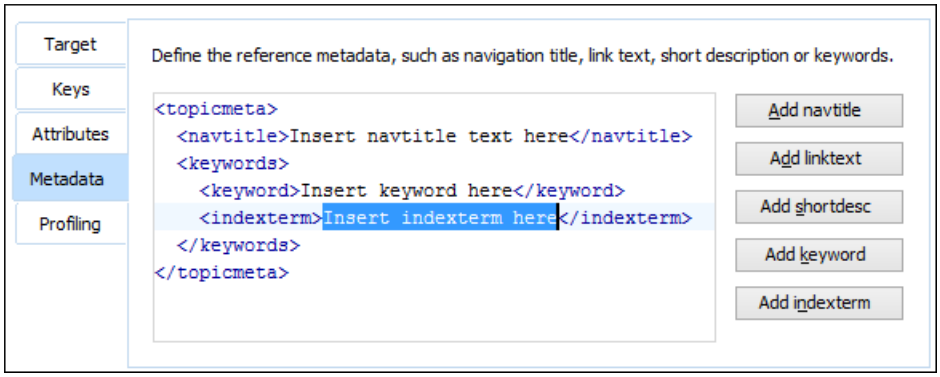


Figure 706: Edit Properties Dialog Box - Metadata Tab

The **Metadata** tab allows you to add metadata elements to the target node. Use the buttons on the right side of the tab to insert specific metadata elements (you can add the following metadata elements: `navtitle`, `linktext`, `shortdesc`, `keyword`, `indexterm`). The metadata elements are inserted inside a `topicmeta` element. The editing window allows you to easily insert and modify the content of the metadata that will be inserted.

Profiling Tab

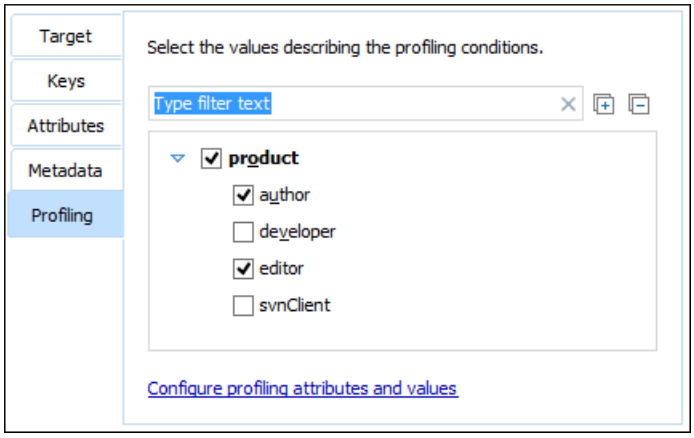


Figure 707: Edit Properties Dialog Box - Profiling Tab

The **Profiling** tab allows you to select or change profiling attributes for the selected target nodes. This tab displays profiling attributes and their values as determined by the following:

- If your *root map* references a DITA *subject scheme map* that defines values for the profiling attributes, those values are used.
- If your project defines *project-level* configuration values for the *profiling attributes*, those values are used.
- If Oxygen XML Editor defines *global-level* configuration values for the *profiling attributes*, they are used.
- Otherwise, a basic default set of profiling attributes and values are used.

If you have a large list of profiling attributes, you can use the text filter field to search for attributes or values, and you can expand or collapse attributes by using the **Expand All/** **Collapse All** buttons to the right of the text filter or the arrow button to the left of the profiling attribute name..

When you modify a selection of values in this tab, the change will also automatically be reflected in the **Attributes** tab. For more information, see *DITA Profiling / Conditional Text* on page 1698.

Note: If you invoke the **Edit properties** action on a selection of multiple nodes that have different values for the same profiling attribute, a conflict panel will be displayed in the **Profiling** tab and you can choose between the following actions for resolving it:

- **Keep** - Preserves the current attribute values.
- **Change Now** - Allows you to edit the selection of values in this **Profiling** tab and the changes will be applied to all the selected nodes.

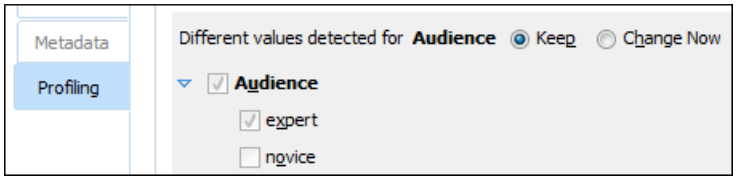


Figure 708: Profiling Conflict Panel

Finalizing Your Modifications

Once you click **OK**, all your changes are applied to the target node.

Related Information:

DITA Profiling / Conditional Text on page 1698
Working with Keys on page 1652

Creating a Table of Contents in DITA

In DITA, the order and hierarchy of the table of contents of a document is based directly on *the DITA map that defines the document*. In many media, the creation of a table of contents, based on the map, is automatic. For example, you do not have to do anything special to create a table of contents in WebHelp output.

In other media, you need to tell DITA where the table of contents should occur. For example, in a book you need to tell DITA where to place the table of contents in the structure of the book, and if you want to generate other common content lists, such as a list of figures or tables. You do this by *using a bookmap to define your book*, and adding the appropriate elements to the `frontmatter`.

To configure a table of contents and other book lists in a *bookmap*, follow these steps:

1. Open your *bookmap* in the ***DITA Maps Manager***.
2. If your *bookmap* is empty, right-click the *bookmap* and select **Append Child** > **Frontmatter**. If your *bookmap* already has nodes added to it, right-click the first node within *bookmap* and select **Insert Before** > **Frontmatter**. The ***Insert Reference*** *dialog box* appears.
3. Click **Insert and Close** to insert the `frontmatter` element.
4. Right-click the `frontmatter` element and create a `booklist` element using **Append child** > **Book Lists**.
5. Use the same steps to create a `toc` element and to add other `booklist` elements, such as `tablelist`.

Creating an Index in DITA

In DITA, indexes are created from `indexterm` elements. You can insert index term elements:

- In the header of a topic. In paginated media, such as a printed book or a PDF, this results in an index entry that points to the page in which the topic starts, even if it is not the page in which the indexed term occurs.
- In the `topicref` element in a map that references the topic. This applies those index terms to that topic only when used in that map, allowing you to index topics differently in various publications. In paginated media, index entries point to the page in which the topic starts.
- In the body of a topic. In paginated media, this results in an index entry that points to the page in which the `indexterm` element occurs, even if that is not the page in which the topic starts.

To add index terms to the text of a topic of the topic header, *create the elements, as you normally would, in Oxygen XML Editor*. To add index terms to a map, open the map in the editor and add the elements, as you normally would, in a topic.

In some media, indexes will be generated automatically when index entries are found in the source. For other media, such as books, you may need to tell DITA where to place the index. For instance, to add an index to a *bookmap*, you need to add an `indexlist` element to the `backmatter` of the book.

1. Open your *bookmap* in the ***DITA Maps Manager***.

2. Right-click the *bookmap* and select **Append Child** > **Backmatter**.
   The ***Insert Reference*** *dialog box* appears.
3. Click **Insert and Close** to insert the `backmatter` element.
4. Right-click the `backmatter` element and create a `booklists` element using **Append Child** > **Book Lists**.
5. Use the same steps to create an `indexlist` element.

CAUTION: Adding `index` entries and an `indexlist` to your project creates an instruction to the DITA publishing routines to create an index. There is no guarantee that all DITA output types or third-party customizations obey that instruction or create the index the way you want it. Modifying the output may be necessary to get the result you want.

Resolving Topic References Through an XML Catalog

There are situations where you want to resolve URIs with an *XML Catalog*:

- You customized your *DITA map* to reference topics using URIs instead of local paths.
- You have URI content references in your DITA topic files and you want to map them to local files when the map is transformed.

In such situations, you have to add the catalog to Oxygen XML Editor. The ***DITA Maps Manager*** *view* will solve the displayed topic refs through the added *XML catalog*, as will as *DITA map* transformations (for PDF output, XHTML output, etc.)

To add an *XML catalog* to the DITA *framework*, follow these steps:

1. Create an *XML catalog* using the guidelines described in *Working with XML Catalogs* on page 484.
2. *Open the* ***Preferences*** *dialog box* ***(Options*** *>* ***Preferences)*** and go to **Document Type Association**.
3. Select the **DITA** document type and use the **Edit**, **Duplicate**, or **Extend** button to open a ***Document type*** *configuration dialog box*.
4. Go to the ***Catalogs*** *tab*.
5. Click on the **Add** button to open a dialog box that allows you to add your created *XML Catalog* to the list.
6. After adding your catalog, click **OK**. You may need to reopen any currently edited files that use the new catalog or run *a manual* ***Validate*** *action* for the changes to take effect.

   Note: You could also add your created catalog to the list of global catalogs in the ***XML Catalog*** *preferences* page.

Chunking DITA Topics

By default, when a *DITA map* is published to an online format, each topic becomes a separate page in the output. In some cases, you may want to combine multiple source topics into one output page. For instance, you may want to combine several types of information into a single page, or you may have chosen to create many small DITA topics for reuse purposes but feel they are too small to be useful to a reader by themselves. This is referred to as *chunking*.

To chunk DITA topics, you set the chunking attribute on the `topicref` that contains the sub-topics in a *DITA map*. There are several values that you can set on the chunking attribute (for example, `by-topic` or `to-content`). See the *DITA documentation* for full details. To achieve the effects you want in your topics and table of contents, you may also need to set the `toc` and `collection-type` attributes on the sub-topics or container topic to suitable values. See the *DITA documentation* for details.

You can set the `collection-type` attribute on your topics using the **Edit Properties** action in the ***DITA Maps Manager***. To set the `toc` and `chunk` attributes, you must open the map file in the editor and add or edit the attributes directly (double-click the map icon in the ***DITA Maps Manager*** to open the map in the editor).

DITA Map Validation and Completeness Check

You should validate your *DITA maps* regularly to make sure that your maps and topics are valid, and all of the relationships between them are working. Changing one topic, image, or piece of metadata may create errors in references that rely on them. You may not discover these problems all at once. Validate your map to catch all of

these kinds of problems. The longer you wait between validating your maps, the more difficult it may be to detect and correct any errors you find.

Validating a DITA Map

To validate a DITA, follow these steps:

1. In the ***DITA Maps Manager*** *view*, make sure that the tab that holds your *root map* is selected and that the **Root map** selection is set either to the name of your *root map* or to `<current map>`.
2. It is a good practice to refresh your *DITA map* before running the validation process. To do so, select the *DITA map* in the **DITA Maps Manager** view and click **Reload (F5)**.
3. Click the **Validate and Check for Completeness** button on the **DITA Maps Manager** toolbar to open the ***DITA Map Completeness Check*** *dialog box*.
4. If you are using profiling, check the **Use DITAVAL filters** box and select the appropriate option.
5. Select any other options you want to check.
6. Click **Check** to run the validation process.

Validation Process

The validation process of a *DITA map* includes the following:

- Verifies that the file paths of the topic references are valid. For example, if an `href` attribute points to an invalid file path, it is reported as an error in the message panel at the bottom of the editor.
- Validates each referenced topic and map. Each topic file is opened and validated against the appropriate DITA DTD. If another *DITA map* is referenced in the main one, the referenced *DITA map* is verified recursively, applying the same algorithm as for the main map.
- If errors or warnings are found, they are displayed in a separate message pane at the bottom of the editor and clicking them takes you to the location of the error or warning in the file where it was found.

DITA Map Completeness Check Dialog Box

The **DITA Map Completeness Check** dialog box allows you to configure the *DITA map* validation.

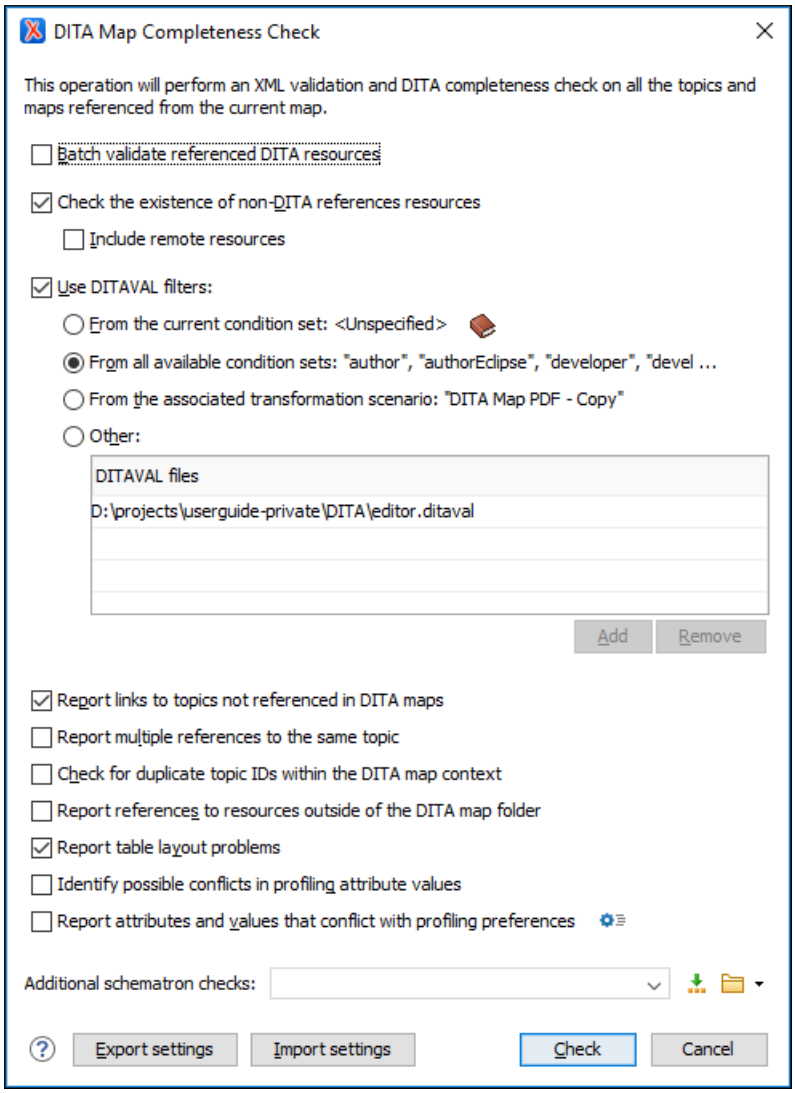


Figure 709: DITA Map Completeness Check Dialog Box

You can configure the validation process with the following options that are available in the **DITA Map Completeness Check** dialog box:

Batch validate referenced DITA resources

This option specifies the level of validation that applies to referenced DITA files:

- If the checkbox is left unchecked (default setting), the DITA files will be validated using the rules defined in the DTD or XML Schema declared in the document.
- If the checkbox is selected, the DITA files will be validated using rules defined in their associated *validation scenario*.

Check the existence of non-DITA references resources

Extends the validation of referenced resources to non-DITA files.

Include remote resources

Select this option if you want to check that remote referenced binary resources (such as images, movie clips, ZIP archives) exist at the specified location.

Use DITAVAL filters

The content of the map is filtered by applying a *profiling condition set* before validation. You can choose between the following options:

- **From the current condition set** - The map is filtered using the condition set currently applied in the ***DITA Maps Manager*** *view*. Clicking the **Details** icon opens a topic in the Oxygen XML Editor User Guide that explains how to create a profiling condition set.
- **From all available condition sets** - For each available condition set, the map content is filtered using that set before validation.
- **From the associated transformation scenario** - The filtering condition set is specified explicitly as a DITAVAL file in the current transformation scenario associated with the *DITA map*.
- **Other DITAVAL files** - For each DITAVAL file from this list, the map content is filtered using the DITAVAL file before validation. Use the **Add** or **Remove** buttons to configure the list. The **Add** button opens a dialog box that allows you to select a local or remote path to a DITAVAL file. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.

Report links to topics not referenced in DITA maps

Checks that all the topics referenced by other topics are also linked in the *DITA map*.

Report multiple references to the same topic

If selected, it will report warnings when a topic is referenced multiple times in the *DITA map*, unless a unique `copy-to` attribute is used on the `topicref` element for any topic that is referenced multiple times.

For example, it will **not** report a warning if there is a topic referenced twice, but the second `topicref` has a `copy-to` attribute set:

```
<topicref href="topic.dita"/>
.....
 <topicref href="topic.dita" copy-to="topic2.dita"/>
```

On the other hand, it **will** report a warning if there is a topic referenced twice and none of the reference-type elements has a `copy-to` attribute set or both of them have the `copy-to` attribute set to the same value:

```
<topicref href="topic.dita" copy-to="topic2.dita"/>
......
 <topicref href="topic.dita" copy-to="topic2.dita"/>
```

Check for duplicate topic IDs within the DITA map context

Checks for multiple topics with the same ID in the context of the entire map.

Report references to resources outside of the DITA map folder

If selected, it will report any references to DITA resources that are located outside the *main DITA map* folder.

Report table layout problems

Looks for table layout problems. The types of errors that may be reported include:

- If a row has fewer cells than the number of columns detected.
- For a *CALS* table, if a cell has a vertical span greater than the available rows count.
- For a *CALS* table, if the number of `colspecs` is different than the number of columns detected from the table `cols` attribute.
- For a *CALS* table, if the number of columns detected from the table `cols` attribute is different than the number of columns detected in the table structure.
- For a *CALS* table, if the value of the `cols`, `rowsep`, or `colsep` attributes are not numeric.
- For a *CALS* table, if the `namest`, `nameend`, or `colname` attributes point to an incorrect column name.

Identify possible conflicts in profile attribute values

When the profiling attributes of a topic contain values that are not found in parent topic profiling attributes, the content of the topic is overshadowed when generating profiled output. This option reports these possible conflicts.

Report attributes and values that conflict with profiling preferences

Looks for profiling attributes and values that are not defined in the ***Profiling / Conditional Text*** *preferences page* (you can click the **Profiling Preferences** button to open this preferences page). It also checks if profiling attributes defined as *single-value* have multiple values set in the searched topics.

Additional schematron checks

Allows you to select a Schematron file that Oxygen XML Editor will use for the validation of DITA resources. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.

Export settings

Allows you to export the settings assigned in this dialog box to an XML file that you can share with other users or use on other systems.

Import settings

Allows you to import settings for this dialog box from an XML file that was created by the **Export settings** action.

Check

Use the **Check** button to begin the validation process. The options that you choose in this dialog box are preserved between sessions.

Related Information:

DITA Maps Manager on page 1582

DITA Map Validation and Completeness Check from the Command-Line Interface

Important: This command-line script requires a special **oXygen XML Scripting License** and does not work with a normal Oxygen XML Editor license. For more information and to obtain an **oXygen XML Scripting License** license, see the *oXygen XML Scripting Web Page*.

You can execute a ***Validate and Check for Completeness*** *operation* with a script ran from the command-line interface. The operation can be executed by running the following script:

- **Windows:**

```
validateCheckDITA.bat -i mainDitamapPath -s settingsFile -r reportFile
```

- **Linux:**

```
sh validateCheckDITA.sh -i mainDitamapPath -s settingsFile -r reportFile
```

- **Mac OS X:**

```
sh validateCheckDITAMac.sh -i mainDitamapPath -s settingsFile -r reportFile
```

The script is located in the Oxygen XML Editor installation folder and can be integrated in an external batch process launched from the command-line interface. The script accepts a variety command line arguments that allow you to configure the validation.

The accepted arguments are as follows:

mainDitamapPath

The path to the main *DITA map* file to validate. This is a required parameter.

settingsFile

The options file for the validate and completeness check. This file can be obtained from the ***DITA Map Completeness Check*** *dialog box* by using the **Export Settings** action. If this parameter is missing, the default options for the validation will be used.

reportFile

The file where the validation report will be saved. If this parameter is missing, the report will be displayed in the console.

Note: When using this script, some of the options that appear in the ***DITA Map Completeness Check*** *dialog box* are ignored. The ignored options include:

- ***Batch validate referenced DITA resources***
- ***Use DITAVAL filters: From the current condition set***
- ***Use DITAVAL filters: From all available condition sets***
- ***Use DITAVAL filters: From the associated transformation scenario***

- ***Report attributes and values that conflict with profiling preferences***

Tip: You can use DITAVAL filter files with the script by specifying one or more `ditaval` files in the ***Use DITAVAL filters: Other DITAVAL files*** *option* in the **DITA Map Completeness Check** dialog box.

Result: The validation result is displayed in the command line interface. This includes whether or not the validation was successful, the number of issues, and the path to the report file (if the reportFile parameter was used), or the actual content of the validation result if the reportFile parameter was not used.

Restriction: When using this script, the Saxon 9 XSLT processor used to apply *custom Schematron checks* is not licensed and therefore the Schematron cannot call external Java extensions.

DITA Map Author Mode Actions

A variety of actions are available in the *DITA map framework* that can be added to the **DITA** menu, the **Author Custom Actions** toolbar, the contextual menu, and the *Content Completion Assistant*.

DITA Map Toolbar and Menu Actions

When a *DITA map* is opened in **Author** mode, the following default actions are available on the **DITA Map Author Custom Actions** toolbar (by default, they are also available in the **DITA** menu and in various submenus of the contextual menu):

Insert New Topic

Opens a ***New*** *DITA topic dialog box* that allows you to create a new topic and inserts a reference to it at the cursor position.

Insert Topic Reference

Opens the ***Insert Reference*** *dialog box* that allows you to insert and configure a reference to a topic at the cursor position.

Reuse Content

Opens the ***Reuse Content*** *dialog box* that allows you to insert and configure a content reference (`conref`), or a content key reference (`conkeyref`) at the cursor position.

Insert Topic Heading

Opens the ***Insert Reference*** *dialog box* that allows you to insert a topic heading at the cursor position.

Insert Topic Group

Opens the ***Insert Reference*** *dialog box* that allows you to insert a topic group at the cursor position.

Insert Relationship Table

Opens a dialog box that allows you to configure the relationship table to be inserted. The dialog box allows you to configure the number of rows and columns of the relationship table, if the header will be generated and if the title will be added.

Relationship Table Properties

Allows you to change the properties of rows in relationship tables.

Insert Relationship Row

Inserts a new table row with empty cells. The action is available when the cursor position is inside a table.

Insert Relationship Column

Inserts a new table column with empty cells after the current column. The action is available when the cursor position is inside a table.

Delete Relationship Column

Deletes the table column where the cursor is located.

Delete Relationship Row

Deletes the table row where the cursor is located.

Move Up

Moves the selected node up one position on its same level.

Move Down

Moves the selected node down one position on its same level.

Promote(Alt + LeftArrow)

Moves the selected node up one level to the level of its parent node.

Demote(Alt + RightArrow)

Moves the selected node down one level to the level of its child nodes.

DITA Menu Actions

In addition, the following default actions are available in the **DITA** menu when editing a *DITA map* in **Author** mode:

Refresh References

You can use this action to manually trigger a refresh and update of all referenced resources.

Tags display mode Submenu

Full Tags with Attributes

Displays full tag names with attributes for both *block* and *inline elements*.

Full Tags

Displays full tag names without attributes for both *block* and *inline elements*.

Block Tags

Displays full tag names for *block elements* and simple tags without names for *inline elements*.

Inline Tags

Displays full tag names for *inline elements*, while *block elements* are not displayed.

Partial Tags

Displays simple tags without names for *inline elements*, while *block elements* are not displayed.

No Tags

No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Configure Tags Display Mode

Use this option to go to the ***Author*** *preferences page* where you can configure the **Tags Display Mode** options.

Profiling/Conditional Text Submenu

Edit Profiling Attributes

Allows you to configure the *profiling attributes* and their values.

Show Profiling Colors and Styles

Select this option to turn on conditional styling.

Show Profiling Attributes

Select this option to turn on conditional text markers. They are displayed at the end of conditional text blocks, as a list of attribute name and their currently set values.

Show Excluded Content

When this option is selected, the content filtered by the currently applied condition set is grayed-out. To show only the content that matches the currently applied condition set, deselect this option.

List of all profiling condition sets that match the current document type

You can click a listed *condition set* to activate it.

Profiling Settings

Opens the *profiling options preferences page*, where you can manage profiling attributes and profiling conditions sets. You can also configure the profiling styles and colors options from the *colors/styles preferences page* and the *attributes rendering preferences page*.

ID Options

Opens the **ID Options** dialog box that allows you to configure options for generating IDs in **Author** mode. The dialog box includes the following:

ID Pattern

The pattern for the ID values that will be generated. This text field can be customized using constant strings or any of the Oxygen XML Editor *Editor Variables*.

Element name or class value to generate ID for

The elements for which ID values will be generated, specified using class attribute values. To customize the list, use the **Add**, **Edit**, or **Remove** buttons.

Auto generate IDs for elements

If selected, Oxygen XML Editor will automatically generate unique IDs for the elements listed in this dialog box when they are created in **Author** mode.

Remove IDs when copying content in the same document

When copying and pasting content in the same document, this option allows you to control whether or not pasted elements that are listed in this dialog box should retain their existing IDs. To retain the element IDs, deselect this option.

Note: This option does not have an effect on content that is *cut* and *pasted*.

Generate IDs

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements from the list in the **ID Options** dialog box that are found in the current selection.

Note: The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `id` attribute.

DITA Map Contextual Menu Actions

In addition to many of the ***DITA Map*** *toolbar actions* and the *general* ***Author*** *mode contextual menu actions*, the following *DITA map framework*-specific actions are also available in the contextual menu when editing in **Author** mode:

Edit Properties

Opens the **Edit Properties** dialog box that allows you to configure the properties of a selected node. For more details about this dialog box, see *Edit Properties Dialog Box* on page 1603.

Generate IDs

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements from the list in the **ID Options** dialog box that are found in the current selection.

Note: The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `id` attribute.

Search References

Finds the references to the `href` or `keys` attribute value of the topic/map reference element at the current cursor position, in all the topics from the current *DITA map* (opened in the ***DITA Maps Manager*** *view*). The current topic/map reference element must have an `href` or `keys` attribute defined to complete the search.

Show Key Definition

Available for elements that have a `conkeyref` or `keyref` attribute set (or elements with an ancestor element that has a `conkeyref` or `keyref` attribute). It computes the key name and opens the *DITA map* that contains the definition of the key with the element that defines that key selected.

DITA Map Drag/Drop Actions

Dragging a file from the ***Project*** *view* or ***DITA Maps Manager*** *view* and dropping it into a *DITA map* document that is edited in **Author** mode creates a link to the dragged file (a `topicref` element, `chapter`, `part`, etc.) at the drop location.

Open Topic from DITA Map in Author Mode

When a *DITA map* is opened in **Author** mode, you can click the icon to the left of a topic to open that particular topic in the editor.

Working with DITA Topics

DITA is a structured writing format. Structure can have several meanings, all of which are relevant to DITA. This section includes information about working with DITA topics and the structure.

Information Types

The structure of a piece of content refers to how the words and images are selected and organized to convey information. One approach to structured writing is to divide content into discrete blocks that contain various types of information, and then to combine those blocks to form publications. DITA is based on this approach, and encourages the author to write in discrete blocks called topics. DITA provides three base topic types (concept, task, and reference), a number of extended topic types, and the capability to create new topic types through specialization.

Text Structure

Every piece of text is made up of certain text structures, such as paragraphs, lists, and tables. DITA supports text structures through XML elements such as `p`, `ol`, and `simpletable`. *The DITA markup* specifies the text structures, but not how they will be published in various types of media. The formatting of text structures is determined by the output transformations and may be customized to meet the needs of various organizations and type of media.

Semantic Structure

Semantic structure is structure that shows the meaning of things. For example:

- A `task` element specifies that a block of content contains the description of a task.
- A `codeblock` element specifies that a block of text consists of programming code.
- A `uicontrol` element specifies that a word is the name of a control in a computer GUI.
- The `platform` profiling attribute specifies that a particular piece of content applies only to certain computing platforms.

Semantic structure is important in a structured writing system because it allows both authors and readers to find content, and it allows processing scripts to process various pieces of content differently, based on their role or meaning. This can be used to do things such as filtering content related to a specific product so that you can produce documentation on many products from the same source.

There can be many forms of semantics captured in a document set. DITA captures some of these in topics and some of them in maps. If you are using a CMS, it may capture additional semantics.

Document Semantics

Documents consist of elements that may be made up of the same basic text structures as the rest of the text, but have a special function within the structure of the document. For instance, both tables of contents and indexes are lists, but they play a special role in the document. Chapters and sections are just sequences of paragraphs and other text structures, yet they are meaningful in the structure of the document. In some cases, such as indexes and tables of contents, these structures can be generated from semantic information embedded in the source. For instance, a table of contents can be built by reading the titles of chapters and sections. DITA provides elements to describe common document semantics.

Subject Matter Semantics

In some cases, the semantics of the content relate directly to the subject matter that the content describes. For instance, DITA supports tags that allow you to mark a piece of text as the name of a window in a software application (`wintitle`), or to mark a piece of text as applying only to a particular product.

Audience Semantics

In some cases, the semantics of the content relate to the audience that it is addressed to. For instance, a topic might be addressed to a particular role, or to a person with a particular level of experience. DITA provides an `audience` element to capture audience metadata.

Creating Topic Structures

Oxygen XML Editor provides a number of tools to help you create topic structures:

- *Content Completion Assistant* - Shows you which elements can be created at the current position.
- ***Model** view* - Shows you the complete structure supported by the current element.
- ***Outline** view* - Shows you the current structure of your document.
- **DITA** toolbar - Helps you to easily insert many common structures.

Related Information:

Your First DITA Topic on page 9
DITA Topics Document Type (Framework) on page 794

Creating a New DITA Topic

The basic building block for DITA information is the DITA topic. DITA provides a variety of specialized topic types, the most common of which are:

- *Topic* - The base topic type from which all other topic types are specialized. Typically, it is used when a more specialized topic type is inappropriate.
- *Task* - For procedural information such as how to use a dialog box.
- *Concept* - For general, conceptual information such as a description of a product or feature.
- *Reference* - For reference information.

Oxygen XML Editor also supports numerous other specialized topic types that you will find templates for in the various folder in the ***New** DITA file dialog box*. They include DITA 1.3 specializations, Lightweight DITA templates, MathML composites, Markdown documents, and other DITA specialized topic and *DITA map* types such as *Glossentry*, *Troubleshooting*, *Bookmap*, and *Subject Scheme Map*.

To create a new DITA topic and add a reference to it in your *DITA map*, follow these steps:

1. In the ***DITA Maps Manager***, right-click the node in the current map where you want to add the new topic.
2. Select one of the following actions:
   - **Append Child** > **New** - Select this action to insert the new topic as a child of the selected node. This action opens a ***New** file dialog box* that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click **Create**.

- **Insert Before** > **New** - Select this action to insert the new topic as a sibling to the current node, before it. This action opens a ***New*** *file dialog box* that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click **Create**.
- **Insert After** > **New** - Select this action to insert the new topic as a sibling to the current node, after it. This action opens a ***New*** *file dialog box* that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click **Create**.
- **Duplicate** - Select this action to create a copy of the selected topic and insert it as a sibling. This action opens a dialog box that allows you to choose the file name and location for the newly created copy of the topic. After you have selected the name and path for your new topic, click **OK**.

Step Result: The new topic is now referenced (as a `topicref`) in the *DITA map* at the location where you inserted it and the new topic is opened in the editor.

3. Save the *DITA map*.

New DITA File Dialog Box

The **New** DITA file dialog box allows you to create a new DITA topic using various types of DITA file templates and provides some options that help you to configure the new topic.

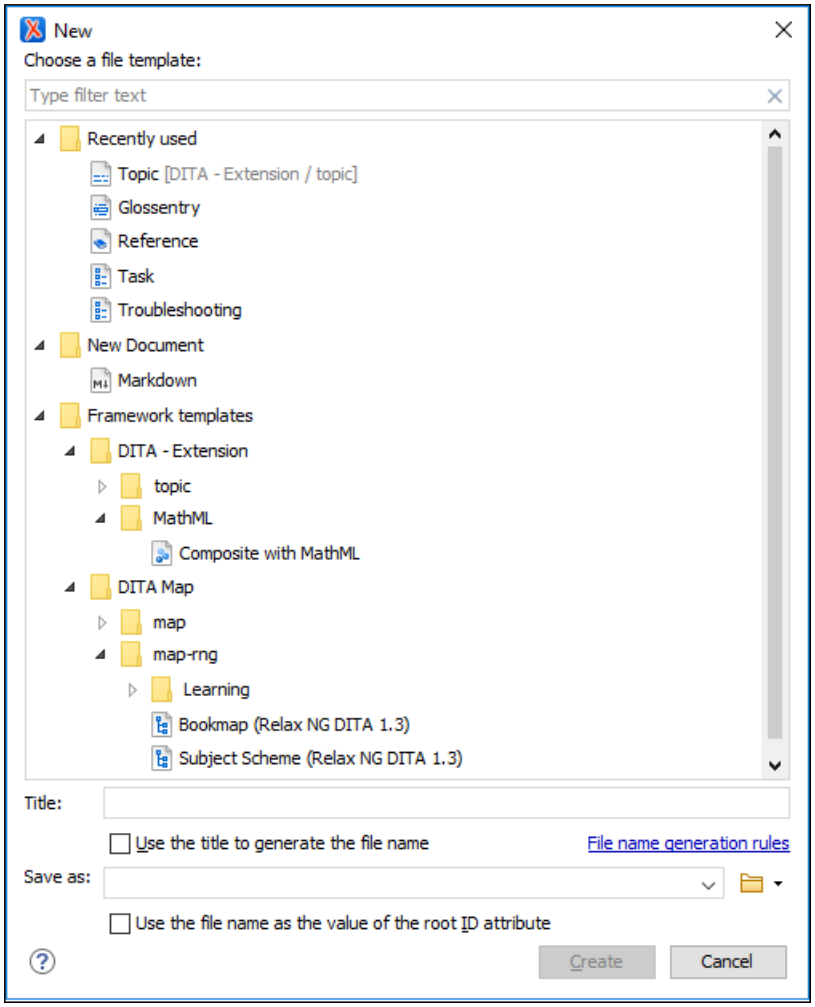


Figure 710: New DITA File Dialog Box

Note: The templates that appear in this dialog box include all templates that have an associated `.properties` file and the `type` property is set to `dita`, as well as templates that do not have an associated properties file or the `type` property is not defined. It will also include custom templates that you create using the procedures presented in *Creating New Document Templates* on page 256.

The **New** DITA file dialog box includes the following features and options:

Choose a file template

Use the template preview pane to select the appropriate type of DITA file you want to create.

Tip: You can use the text filter field at the top of the dialog box to search for a specific template.

Title

Depending on the selected file template, the value of the **Title** field is set in:

- the `title` element of a DITA topic file. The `title` element needs to be the first child of the root element.
- the `glossterm` element of a *Glossentry* file.

Use the title to generate the file name

Select this option to use the text entered in the **Title** field to automatically generate a file name. By default, the generated name will transform spaces into underscores (_), all illegal characters will be removed, and all upper case characters changed to lower case (the generated name can be seen in the **Save as** field). You can configure these rules by clicking the ***File name generation rules*** *link*.

Save as

Use this option to specify a file name and path for the new file. You can specify the path by using the text field, the history drop-down, or the browsing tools in the **Browse** drop-down menu.

Use the file name as the value of the root ID attribute

Select this option to use the file name in the **Save as** field (without the file extension) as the value of the root `id` attribute for the new topic.

Create

When you click this button, a reference (`topicref`) to the new topic is added to the current *DITA map* and the new topic is opened in the editor.

Related Information:

Your First DITA Topic on page 9
Adding Topics to a DITA Map on page 1592
Working with Markdown Documents in DITA on page 714
Fast Create Multiple DITA Topics on page 1620

Fast Create Multiple DITA Topics

The ***DITA Maps Manager*** includes a feature that allows you to quickly create multiple skeleton topics at once and you can specify their hierarchical structure within the *DITA map*. A common use-case for using this feature is when you need to insert a new chapter or section that will include multiple topics and you have the structure and titles planned out in advance.

To access this feature, right-click a node in the **DITA Maps Manager** where you want the new topics to be inserted and select **Fast Create Topics**. This opens the **Fast Create Topics** dialog box where you can configure the structure for the new topics.

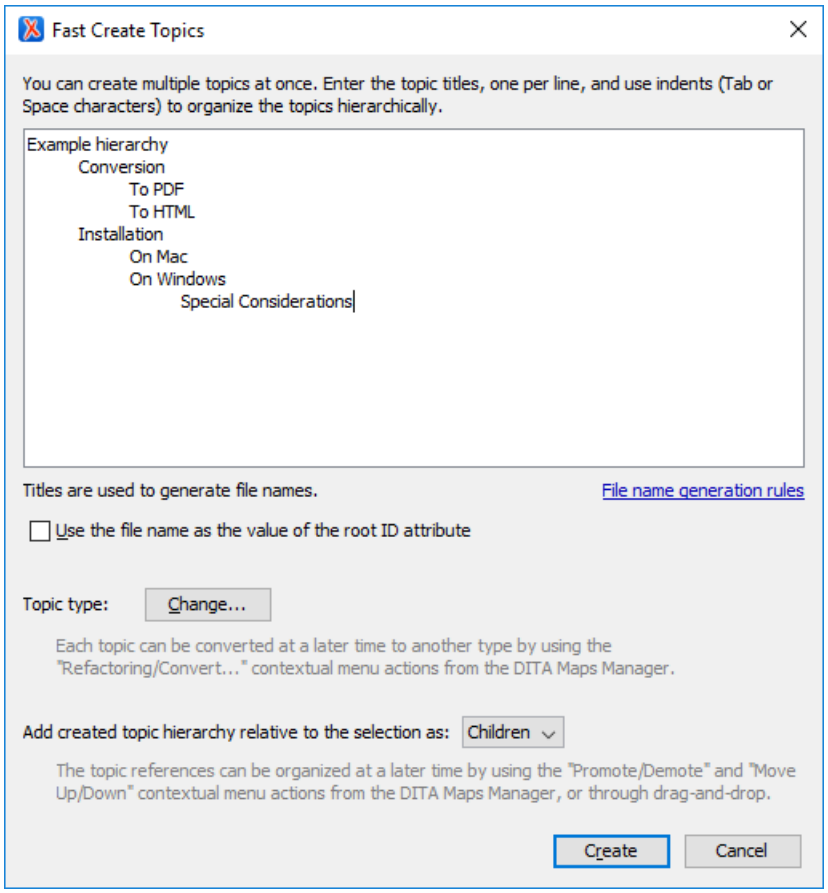


Figure 711: Fast Create Topics Dialog Box

The **Fast Create Topics** dialog box includes the following features and options:

Hierarchy Text Pane

Use this text area to enter the titles for your new topics, one per line, and specify the hierarchy by using indents (**Tab** or **Space**). Topic references will be created in the *DITA map* according to the hierarchy you enter in this section. Also, the titles that you enter in this text pane will not only be used for the topic titles but also to generate their file names and you can click the ***File name generation rules*** *link to configure the rules* for how those file names will be generated.

Use the file name as the value of the root ID attribute

Select this option to use the file name (without the file extension) as the value of the root `id` attribute for the new topics.

Topic type

All of the topics that will be created will have the same DITA topic type, which is detected from the most recently created topic. You can click the **Change** button to select a different type from a list of possible DITA templates.

Tip: You can convert any of these new files to a different DITA topic type at a later time by using another *feature that allows you to easily convert DITA documents to other types*.

Add created topic hierarchy relative to the selection as

By default, the hierarchy of topics will be added to the *DITA map* as **Children** to the node where the action was invoked. You can change this to **Siblings** if the selected node allows topics to be inserted as such.

Create

When you click **Create**, the specified hierarchy is added as topic references in the *DITA map*. The new documents are created as bare skeleton topics with only the topic title and possibly the root ID populated.

Tip: You can easily *change the order of the topics in the DITA map* at a later time,

Related Information:

Adding Topics to a DITA Map on page 1592
Converting DITA Topics to Another Type on page 1624

Editing DITA Topics

Oxygen XML Editor provides a number of features to help you edit DITA topics. A DITA topic is an XML document, thus all the editing features that Oxygen XML Editor provides for editing XML documents also apply to DITA topics. Oxygen XML Editor also provides extensive additional support specifically for DITA.

Opening a DITA Topic

There are several ways to open a DITA topic in the XML editor. Use any of the following methods to open a topic:

- Double-click the topic in the ***DITA Maps Manager*** (or right-click the topic and select **Open**).
- Double-click the file in the ***Project*** *view* (or right-click the file and select **Open**).
- If you have a *DITA map* opened in the XML editor, you can click the icon to the left of the topic.
- Drag a DITA file from your system browser and drop it in the XML editor.

Visual Editing in Author Mode

DITA is an *XML format*, although you do not have to write raw XML to create and edit DITA topics. Oxygen XML Editor provides a graphical view of your topics in ***Author*** *mode*. Your topics will likely open in **Author** mode by default, so this is the first view you will see when you open or edit a DITA topic. If your topic does not open in **Author** mode, just click **Author** at the bottom left of the editor window to switch to this mode.

Author mode presents a graphical view of the document you are editing, similar to the view you would see in a word processor. However, there are some differences, including:

- **Author** mode is not a **WYSIWYG** view. It does not show you exactly what your content will look like when printed or displayed on-screen. The appearance of your output is determined by the DITA publishing process, and your organization may have modified that process to change how the output is displayed. Oxygen XML Editor has no way of determining what your final output will look like or where line breaks or page breaks will fall. Treat **Author** mode as a friendly visual editing environment, not a faithful preview of your output.
- Your document is still an XML document. **Author** mode creates a visual representation of your document by applying a CSS stylesheet to the XML. You can see the XML at any time by switching to ***Text*** *mode*. You, or someone in your organization, can change how the **Author** view looks by changing the CSS stylesheet or providing an alternate stylesheet.
- Your aim in editing a DITA document is not to make it look right, but to create a complete and correct DITA XML document. **Author** mode keeps you informed of the correctness of your content by highlighting XML errors in the text and showing you the current status in a box at the top right of the editor window. Green means that your document is valid, yellow means valid with warnings, and red means invalid. Warnings and errors are displayed when you place the cursor on the error location.
- Your XML elements may have attributes set on them. Conventionally, attributes are used to contain metadata that is not displayed to the reader. By default, attributes are not displayed in the **Author** view (though there are some exceptions) and cannot be edited directly in the **Author** view (though in some cases the CSS that drives the display may use form controls to let you edit attributes directly). To edit the attributes of an element, place your cursor on the element and press **Alt+Enter** to bring up the attribute editor. Alternatively, you can use the ***Attributes*** *view* to edit attributes.

Tip: You can select **Hints** from the **Styles** drop-down menu (available on the **Author Styles** toolbar) to display tooltips throughout the DITA document that offers additional information to help you with the DITA structure. For more information, see the *Selecting and Combining Multiple CSS Styles* section.

Content Completion Assistance

Since it is a structured format, DITA only allows certain elements in certain places. The set of elements allowed differ from one DITA topic type to another (this is what makes one topic type different from another). To help you figure out which elements you can add in any given place and help you understand what they mean, Oxygen XML Editor has a number of *Content Completion Assistant* features.

- **The Enter key:** In **Author** mode, the Enter key does not create line breaks, it brings up the *Content Completion Assistant* to help you enter a new element. In XML, you do not use line breaks to separate paragraphs. You create paragraphs by creating paragraph elements (element p in DITA) and tools insert the line breaks in the output and on-screen.

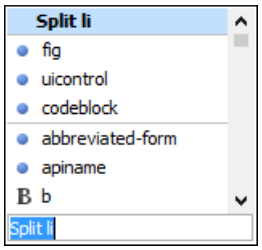


Figure 712: Content Completion Assistant

The *Content Completion Assistant* not only suggests new elements you can add. If you press **Enter** at the end of a *block element* (such as a paragraph) it suggests creating a new element of the same type. If you press **Enter** in the middle of a *block element*, it suggests splitting that element into two elements.

A useful consequence of this behavior is that you can create a new paragraph simply by hitting **Enter** twice (just as you might in a text editor).

As you highlight an element name, a basic description of the element is displayed. Select the desired element and press **Enter** to create it.

To wrap an element around an existing element or piece of text, simply select it and press **Enter** and use the *Content Completion Assistant* to choose the wrapper element.

- **The Model view:** You can see the entire model of the current element by opening the ***Model*** *view* (**Window** > **Show View** > **Model**, if the view is not already open). The **Model** view shows you what type of content the current element can contain, all the child elements it can contain, all its permitted attributes, and their types.

Tip: You can also select **Inline actions** from the **Styles** drop-down menu (available on the **Author Styles** toolbar) to display possible elements that are allowed to be inserted at various locations throughout the DITA document. For more information, see the *Selecting and Combining Multiple CSS Styles* section.

DITA Editing Actions

A variety of actions are available in the DITA *framework* to specifically assist you with editing DITA documents. These various actions are available in the contextual menu, the **DITA** menu, the **DITA (Author Custom Actions)** toolbar, or the *Content Completion Assistant*.

The **DITA** toolbar contains buttons for inserting a number of common DITA elements (elements that are found in most DITA topic types).

If the **DITA** toolbar is not displayed, right-click anywhere on the toolbar area, select **Configure Toolbars**, and select it from the displayed dialog box.

Note: The **DITA** toolbar contains a list of the most common elements and actions for DITA, such as inserting an image, creating a link, inserting a content reference, or creating a table. It does not contain a button for inserting every possible DITA element. For a complete list of elements that you can insert at the current location in your document, press **Enter** to open the *Content Completion Assistant*.

Whenever the current document in the editor is a DITA document, the **DITA** menu is displayed in the menu bar. It contains a large number of actions for inserting elements, creating content references and keys, editing DITA documents, and controlling the display. These actions are specific to DITA and supplement the general editing commands available for all document types. Many of these actions are also conveniently available in the contextual menu. In addition to the *DITA framework-specific actions*, the contextual menu also includes various *general* ***Author*** *mode contextual menu actions*.

Related Information:

Your First DITA Topic on page 9

DITA Author Mode Actions on page 795

Converting DITA Topics to Another Type

Oxygen XML Editor includes a features that allows you to convert an existing DITA document to a different topic type. For example, if you want to convert a DITA Task to a DITA Topic, or vice versa. There are several ways to access these refactoring actions and you can choose a scope for the operation and some filtering options.

DITA Conversion Refactoring Operations for *DITA*

The following conversion operations are available:

Convert to Concept

Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept).

Convert to Reference

Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference).

Convert to Task

Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task).

Convert to Topic

Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic).

Convert to Troubleshooting

Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting).

Methods for Accessing the DITA Conversion Refactoring Operations

To access the conversion operations, use one of the following methods:

Single Document Method

With the document opened in the editor, right-click anywhere in the main editing pane (or right-click the topic reference in the ***DITA Maps Manager***), go to the **Refactoring** submenu, and choose whichever operation is appropriate for your needs.

Multiple Documents At Once Method

Select **XML Refactoring** from the **Tools** menu (or from the **Refactoring** submenu when you right-click one or more documents in the ***Project*** *view*or the ***DITA Maps Manager*** *view*). Then select whichever operation is appropriate for your needs.

XML Refactoring Wizard Dialog Box

When you select any of the operations, Oxygen XML Editor proceeds to the **XML Refactoring Wizard**. If you used the ***Multiple Documents At Once Method***, the wizard page allows you to choose a scope for the operation and some filtering options:

- **Scope** - Select from a variety of options to define the scope for which resources will be affected by the operation. For example, you can choose to affect all resources in the **Project**, **All opened files**, **Current DITA map hierarchy**, or just the **Current file**.
- **Filters** section
  - **Include files** - Specifies files to be excluded from the operation. You can specify multiple files by separating them with commas and the patterns can include wildcards (such as ***** or **?**).
  - **Restrict to known XML file types only** - Excludes non-XML file types from the operation.
  - **Look inside archives** - If this option is selected, the scope of the operation will include files inside archives.

If you used the ***Single Document Method***, the scope will be the current file so the scope and filtering options are not displayed.

You can then use one of the following buttons to proceed with the operation:

Preview

You can use the **Preview** button to open a comparison panel where you can review all the changes that will be made by the refactoring operation before applying the changes.

Warning: It is always recommended to use the **Preview** button to make sure the operation is not going to do something unexpected and after you click the **Finish** button, any **Undo** action will only revert changes on the current document.

Finish

When you use the **Finish** button, behind the scenes Oxygen XML Editor maps the structure of the previous DITA document type to a structure that fits the new type. In some cases, especially when the previous structure was very complex, the conversion might result in an invalid structure and some manual adjustments might be required.

Related Information:

Editing DITA Topics on page 1622

Refactoring XML Documents on page 491

Adding Images in DITA Topics

There are several ways to add images to a DITA topic, depending on if you want to create a figure element (with a title and caption), just insert an image inline, or if you want to use multiple versions of a graphic depending on the situation. For instance, you might want to use a specific image for each different product version or output media.

Adding an Image Inline

Use the following procedure to add an image inline:

1. Place the cursor in the position you want the graphic to be inserted.
2. Select the **Insert Image** action. The **Insert Image** dialog box appears.

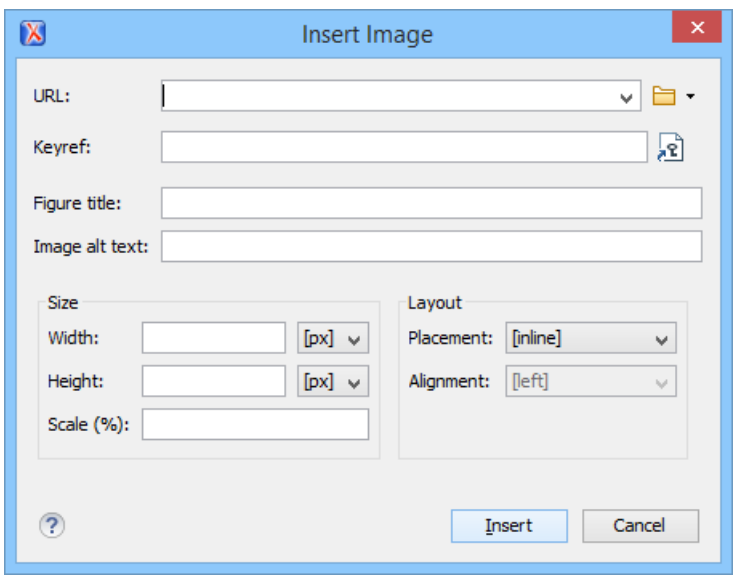


Figure 713: Insert Image Dialog Box

3. Configure the options in this dialog box and click **Insert**.

The **Insert Image** dialog box includes the following options for inserting images into a DITA document:

URL

Use this option to specify a URL for the image. It will insert the URL as the value of an `href` attribute inside an `image` element. You can type the URL of the image you want to insert or use the **Browse** drop-down menu to select an image using one of the following options:

- **Browse for local file** - Displays the **Open** dialog box to select a local file.
- **Browse for remote file** - Displays the ***Open URL*** *dialog box* to select a remote file.

- **Browse for archived file** - Opens the ***Archive Browser*** to select a file from an archive.
- **Browse Data Source Explorer** - Opens the ***Data Source Explorer*** to select a file from a connected data source.
- **Search for file** - Displays the **Find Resource** dialog box to search for a file.

Keyref

You can use the **Choose Key Reference** button to open the **Choose Key** dialog box that presents the list of keys available in the selected *root map*. Use this dialog box to insert the selected key as the value of a `keyref` attribute inside an `image` element. All keys that are presented in the dialog box are gathered from the *root map* of the current *DITA map*. Elements that have the `keyref` attribute set are displayed as links.

Note: If your defined keys are not listed in this dialog box, it is most likely trying to gather keys from the wrong *root map*. You can change the *root map* by using the **Change Root Map** link.

Figure title

Use this text box to insert a `title` and `image` element inside a `fig` element.

Alternate text

Use this text box to insert an `alt` element inside the `image` element.

Size

Use this section to configure the **Width** and **Height** of the image, or **Scale** the image. Specifying a value in these options inserts a `width`, `height`, and `scale` attribute, respectively.

Layout

Use the options in this section to insert `placement` and `align` attributes into the `image` element.

Adding an Image in a Figure Element

To add an image in a figure:

1. Add a `fig` element to your document at the appropriate place.
2. Add a `title` and/or `desc` element to the `fig` element, according to your needs.
3. *Add an `image` element* to the `fig` element.

Note: The `fig` element has a number of other child elements that may be appropriate to your content. See the *DITA documentation* for complete information about the `fig` element.

Note: The order in which the `image`, `title`, and `desc` content are presented in output is determined by the output transformation. If you want to change how they are output, you may have to modify the output transformation, rather than your source content.

Related Information:

Image Maps in DITA on page 415

Adding Video, Audio, and Embedded HTML Resources in DITA Topics

You can insert references to media resources (such as videos, audio clips, or embedded HTML frames) in your DITA topics. The media resources can be played directly in **Author** mode and in all HTML5-based outputs. There is a toolbar button ( ) that allows you to insert and configure a reference to the media resource. You can also drag media files from your system explorer or the ***Project*** *view* and drop them into your documents (or copy and paste them).

Table 52: Supported Media Types

Media	Description	Type	Supported Size Properties
mp3	Moving Picture Experts Group Layer-3 Audio	audio	Width
wav	Windows Wave	audio	Width

Media	Description	Type	Supported Size Properties
pcm	Pulse Code Modulation	audio	Width
m4a	Moving Picture Experts Group Layer-4 Audio	audio	Width
aif	Audio Interchange Format	audio	Width
mp4	Moving Picture Experts Group Layer-4 Video	video	Width & Height
flv	Flash Video	video	Width & Height
m4v	Itunes Video File	video	Width & Height
avi	Audio Video Interleaved	video	Width & Height
embedded video (such as YouTube or Vimeo)	Embedded Iframe Code	iframe	Width & Height

Adding a Media Resource

To insert a media resource in a DITA document, use the following procedure:

1. Place the cursor at the location where you want the media resource.
2. Select the **Insert Media Resource** action from the toolbar. The **Insert Media** dialog box appears.

   Note: You can also drag media files from your system explorer or the ***Project*** *view* and drop them into your documents (or copy and paste them). Note that this method will bypass the **Insert Media** dialog box, so if you need to adjust the size you will need to adjust the `width` or `height` attributes manually.

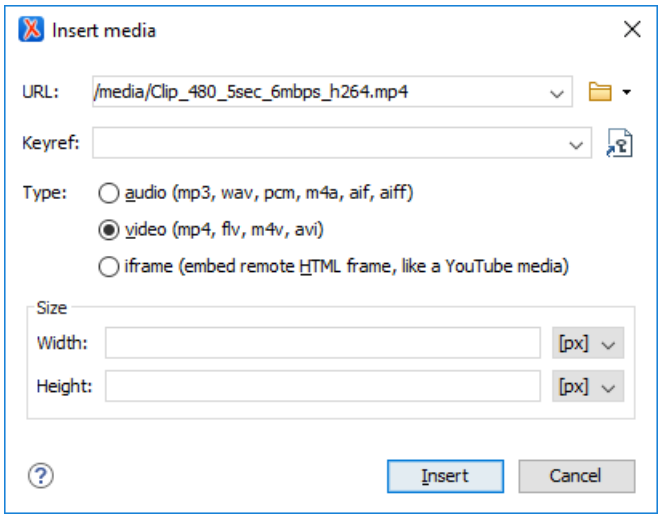


Figure 714: Insert Media Dialog Box

3. Configure the options in this dialog box and click **Insert**.

   The **Insert Media** dialog box includes the following options:

URL

Use this option to specify a URL for a media resource. It will insert the URL as the value of a `data` attribute of an `object` element. You can type the URL of the resource you want to insert or use the **Browse** drop-down menu to select it.

Keyref

You can use the **Choose Key Reference** button to open the **Choose Key** dialog box that presents the list of keys available in the selected *root map*. Use this dialog box to insert the selected key as the value of the `datakeyref` attribute of an `object` element. All keys that are presented in the dialog box are gathered from the *root map* of the current *DITA map*.

Tip: If your defined keys are not listed in this dialog box, it is most likely trying to gather keys from the wrong *root map*. You can change the *root map* by using the **Change Root Map** link.

Type

Oxygen XML Editor detects and automatically selects the media type based upon the specified resource in the *URL field*. You can manually change the type, but keep in mind that in the publishing stage the *`object` element is converted to an HTML5 element* based upon the type selected here. You can choose between: **audio**, **video**, or **iframe**.

Size

Use this section to configure the **Width** and **Height** of the frame for the media resource. Specifying a value in these options inserts a `width` and `height` attribute, respectively. For audio clips, only the **Width** can be adjusted.

Result in Author Mode: A reference to the specified video, audio, or embedded HTML frame is inserted in an `object` element and it is rendered in **Author** mode so that it can be played directly from there.

Result in Output: In the publishing stage, the `object` element is converted to an HTML5 element so that it can be rendered properly and played in all HTML5-based outputs.

- **Videos** - The `object` element is converted to an HTML5 `video` element.
- **Audio Clips** - The `object` element is converted to an HTML5 `audio` element.
- **Embedded HTML Frames** - The `object` element is converted to an HTML5 `iframe` element.

Tip: There is an even faster way of inserting an embedded video (such as a YouTube or Vimeo). If you copy the embed code from the source (for example, you can right-click on a YouTube video and select **Copy embed code**), you can then paste the contents of the clipboard in the ***URL** field* and the ***Type*** will automatically be set on **iframe**, while the ***Width** and **Height*** will be populated according to the detected size.

For more information, see the following video demonstration: *https://www.oxygenxml.com/demo/Media_Objects.html*.

Related Information:

Adding Images in DITA Topics on page 1625
Adding Video and Audio Objects in DITA WebHelp Output on page 968

Image Maps in DITA

Oxygen XML Editor includes support for **image maps** in DITA documents through the use of the `imagemap` element. This feature provides an easy way to create hyperlinks in various areas within an image without having to divide the image into separate image files. The visual **Author** editing mode includes an **Image Map Editor** that helps you to easily create and configure image maps.

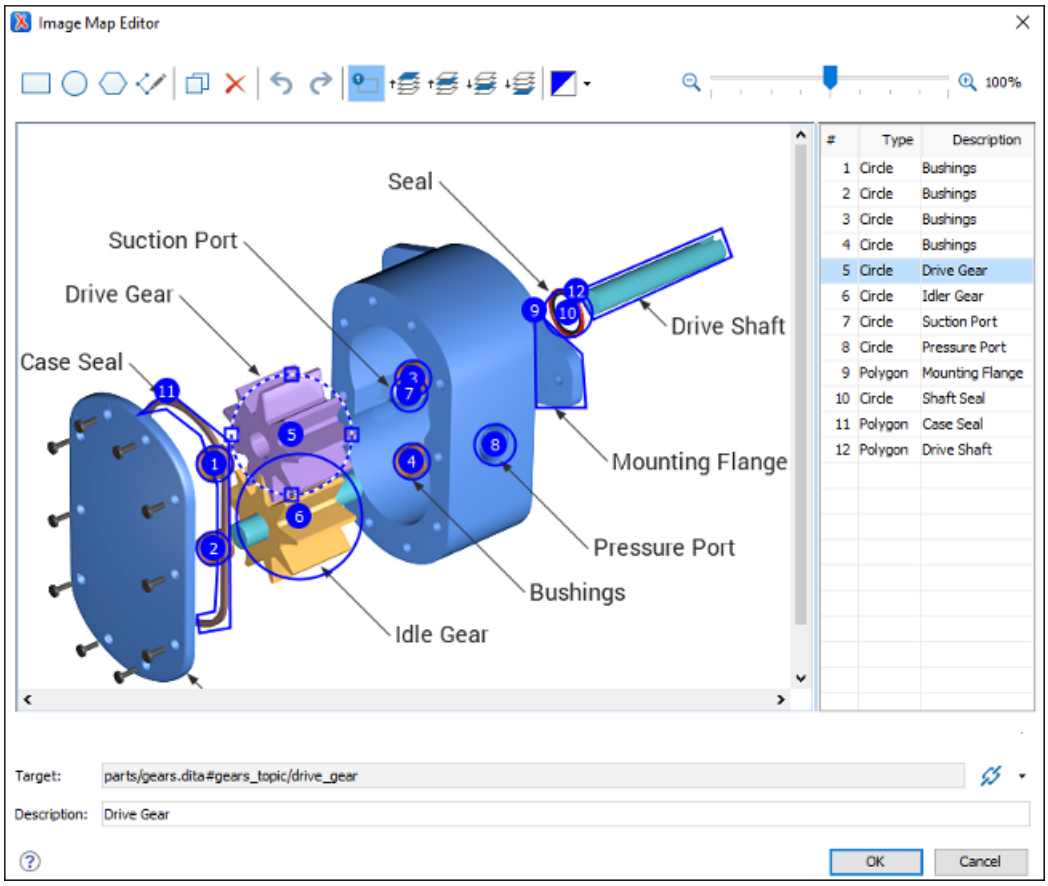


Figure 715: Image Map Editor in DITA

Image Map Editor Interface in DITA

The interface of the **Image Map Editor** consists of the following sections and actions:

Toolbar

New Rectangle

Use this button to draw a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

New Circle

Use this button to draw a circle over an area in the image. You can drag any of the four points to adjust the size of the circle.

New Polygon

Use this button to draw a polygon shape over an area in the image. This actions opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

New Free Form Shape

Use this button to draw a free form shape over an area in the image. After selecting this button, left-click anywhere in the image to place the first point of your shape. Then move the cursor to the location of the next desired point and left-click to place the next point, and so on. To complete the shape (area), click the first point again and a line will automatically be added from the last point that was added, or simply double-click the last point to automatically add the line from the last point back to the first.

Duplicate

Use this button to create a duplicate of the currently selected shape.

Delete

Use this button to delete the currently selected shape.

Undo

Use this button to undo the last action.

Redo

Use this button to redo the last action that was undone.

Show/Hide Numbers

Use this button to toggle between showing or hiding the numbers for the shapes.

Bring Shape to Front

Use this button to bring the currently selected shape forward to the top layer.

Bring Shape Forward

Use this button to bring the currently selected shape forward one layer.

Send Shape Backward

Use this button to send the currently selected shape back one layer.

Send Shape to Back

Use this button to send the currently selected shape back to the bottom layer.

Color Chooser

Use this drop-down menu to select a color scheme for the lines and numbers of the shapes.

Zoom Slider

Use this slider to zoom the image in or out in the main image pane.

Image Pane

This main image pane is where you work with shapes to add hyperlinks to multiple areas within an image. Use the mouse to move shapes around in the image pane. You can hold down the **Ctrl** key to select multiple shapes and then move them simultaneously. You can also drag the points of a selected shape to adjust its size and shape. It is easy to see which shape is selected in this image pane because the border of the selected shape changes from a solid line to a dotted one.

Contextual Menu Actions Available in the Image Pane

You can right-click the shapes, points, or anywhere in the Image Pane to invoke the contextual menu where the following actions are available:

Add Point

Adds a point to *Polygon* or *Free Form* shapes.

Remove Point

Removes the current point from *Polygon* or *Free Form* shapes.

Duplicate

Create a duplicate of the currently selected shape.

Delete

Delete the currently selected shape.

New Rectangle

Creates a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

New Circle

Creates a circle over an area in the image. You can drag any of the four points to adjust the size of the circle.

New Polygon

Creates a polygon shape over an area in the image. This actions opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

Undo

Use this action to undo the last action.

Redo

Use this action to redo the last action that was undone.

Shape Table

The table at the right of the *Image Pane* is a sequential list of all the areas (shapes) that have been added in the image. It shows their number, type, and description (if one has been added). If you select one of the entries in the table, the corresponding shape will be selected in the *Image Pane*.

Properties

Target

Allows you to choose the target resource that you want the selected area (shape) to be linked to. You can enter the path to the target in the text field but the easiest way to select a target is to use the **Link** drop-down menu to the right of the text field. You can choose between the following types of links: **Cross Reference**, **File Reference**, or **Web Link**. All three types will open a dialog box that allows you to define the target resource. This linking process is similar to the normal process of *inserting links in DITA* by using the identical **Link** drop-down menu from the main toolbar.

When you click **OK** to finalize your changes in the **Image Map Editor**, an `xref` element will be inserted with either an `href` attribute or a `keyref` attribute. Additional attributes may also be inserted and their values depend on the target and the type of link. For details about the three types of links and their dialog boxes, see *Inserting a Link in Oxygen XML Editor* on page 1675.

Description

You can enter an optional description for the selected area (shape) that will be displayed in the ***Image Map Details** section* in **Author** mode and as a tooltip message when the end user hovers over the hyperlink in the output.

How to Create an Image Map in DITA

To create an image map on an existing image in a DITA document, follow these steps:

1. Right-click the image and select **Image Map Editor**.

   Result: This action will apply an *image map* to the current image and open the **Image Map Editor** dialog box.
2. Add hyperlinks to the image by selecting one of the shape buttons (**New Rectangle**, **New Circle**, or **New Polygon**).
3. Move the shape to the desired area in the image and drag any of the points on the shape to adjust its size or form. You can use the *other buttons on the toolbar* to adjust its layer and color, or to perform other editing actions.
4. With the shape selected, use one of the *linking options* in the **Link** drop-down menu to select a target resource (or enter its path in the ***Target*** text field).
5. (Optional) Enter a ***Description*** for the selected area (shape).
6. If you want to add more hyperlinks to the image, select a shape button again and repeat the appropriate steps.
7. When you are finished creating hyperlinks, click **OK** to process your changes.

Result: The *image map* is applied on the image and the appropriate elements and attributes are automatically added. In **Author** mode, the image map is now rendered over the image. If the image includes an `alt` element, its value will be displayed under the image. The following two buttons will also now be available under the image in **Author** mode:

- **Image Map Editor** - Click this button to open the **Image Map Editor**.
- **Image Map Details** - Click this button to expand a section that displays the details of the image map and allows you to change the shape and coordinates of the hyperlinked areas. Keep in mind that if you change the shape in this section, you also need to add or remove coordinates to match the requirements of the new shape.

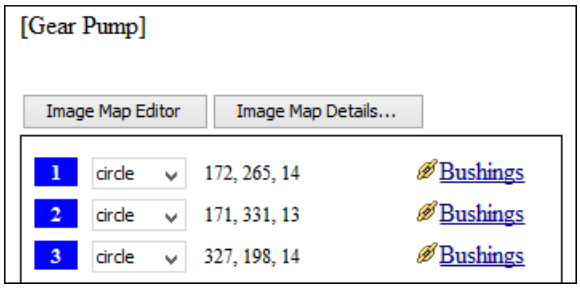


Figure 716: Image Map Details

How to Edit an Existing Image Map in DITA

To edit an existing image map, use any of the following methods:

- Simply double-click the image.
- Right-click the image and select **Image Map Editor**.
- Click the **Image Map Editor** button below the image.

All three methods open the **Image Map Editor** where you can make changes to the image map with a visual editor. You can also make changes to the XML structure of the image map in the **Text** editing mode.

You can also click the **Image Map Details** button below the image to expand a section that displays the details of the image map and allows you to change the shape and coordinates of the hyperlinked areas. Keep in mind that if you change the shape in this section, you also need to add or remove coordinates to match the requirements of the new shape.

Overlapping Areas

If shapes overlap one another in the **Image Map Editor**, the one on the top layer takes precedence. The number shown inside each shape represent its layer (if the numbers are not displayed, click the **Show/Hide Numbers** button on the ***Image Map Editor*** *toolbar*). To change the layer order for a shape, use the layer buttons on the ***Image Map Editor*** *toolbar* (, , ,).

If you insert a shape and all of its coordinates are completely inside another shape, the **Image Map Editor** will display a warning to let you know that the shape is entirely covered by a bigger shape. Keep in mind that if a shape is completely inside another shape, its hyperlink will only be accessible if its layer is on top of the bigger shape.

Warning: PDF output is limited to rectangular shaped image map objects. Therefore, if your image contains circles or polygons, those objects will be redrawn as rectangles in the PDF output. Keep in mind that this might affect overlaps in the output.

Related Information:

DITA 'imagemap' Element Specifications

Adding Images in DITA Topics on page 1625

Adding Tables in DITA Topics

You can use the **Insert Table** action on the toolbar or from the contextual menu to add a table in a DITA topic. By default, DITA supports four types of tables:

- *DITA Simple table model* - This is the most commonly used model for basic tables.
- *CALS table model (OASIS Exchange Table Model)* - This is used for more advanced functionality.
- *DITA Choice table model* - This is used within a `step` element in a DITA Task document to describe a series of optional choices that a user must make before proceeding.
- *DITA Properties table model* - This is used in DITA Reference documents to describe a property (for example, its type, value, and description).

If you are using a specialized DITA vocabulary, it may contain specialized versions of these table models.

Since DITA is a structured format, you can only insert a table in places in the structure of a topic where tables are allowed. The Oxygen XML Editor toolbar provides support for entering and editing tables. It also helps to indicate where you are allowed to insert a table or its components by disabling the appropriate buttons.

Inserting a Simple Table Model

To insert a **Simple** DITA table, select the **Insert Table** action on the toolbar or from the contextual menu (or the **Table** submenu from the **DITA** menu). The **Insert Table** dialog box appears. Select **Simple** for the table **Model**.

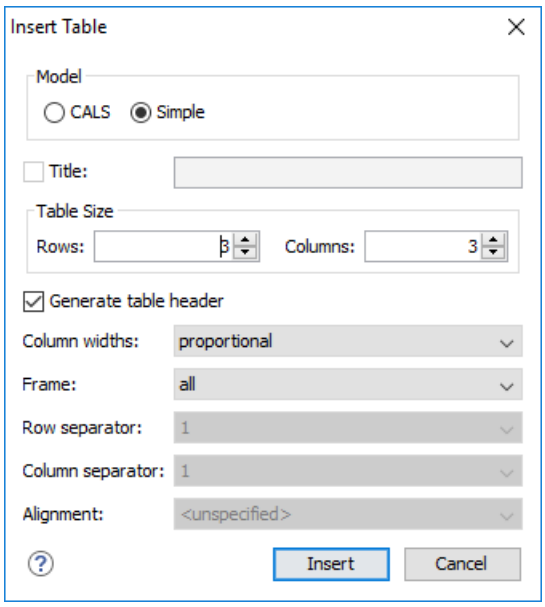


Figure 717: Insert Table Dialog Box - Simple Model

The dialog box allows you to configure the following options when you select the **Simple** table model:

Title

If this checkbox is selected, you can specify a title for your table in the adjacent text box.

Generate table header

If selected, an extra row will be inserted at the top of the table to be used as the table header.

Column widths

Allows you to specify the type of properties for column widths (`colwidth` attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a `relcolwidth` attribute with the values listed as the number of shares followed by an asterisk. The value of the shares are totaled and rendered as a percent. For example, `relcolwidth="1* 2* 3*"` causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in **Author** mode, the values of the `relcolwidth` attribute are automatically

changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the `relcolwidth` attribute is `1*`.

- **dynamic** - If you choose this option, the columns are created without a specified width. Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in **Author** mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.

Frame

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

Note: The options in the **Insert Table** dialog box for DITA documents are persistent, so changes made in one session will carry over to another.

When you click **Insert**, a simple table is inserted into your document at the current cursor position.

Inserting a CALS Table Model (OASIS Exchange Table)

To insert an OASIS Exchange Table (**CALS**), select the **Insert Table** action on the toolbar or from the contextual menu (or the **Table** submenu from the **DITA** menu). The **Insert Table** dialog box appears. Select **CALS** for the table **Model**. This model allows you to configure more properties than the *Simple* model.

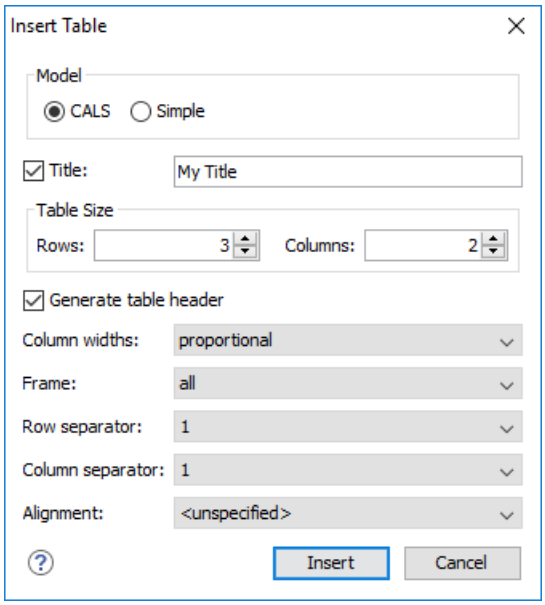


Figure 718: Insert Table Dialog Box - CALS Model

The dialog box allows you to configure the following options when you select the **CALS** table model:

Title

If this checkbox is selected, you can specify a title for your table in the adjacent text box.

Table Size

Allows you to choose the number of **Rows** and **Columns** for the table.

Generate table header

If selected, an extra row will be inserted at the top of the table to be used as the table header.

Column widths

Allows you to specify the type of properties for column widths (`colwidth` attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a `colwidth` attribute with the values listed as the number of shares followed by an asterisk. The value of the shares are totaled and rendered as a percent. For example, `colwidth="1* 2* 3*"` causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in **Author** mode, the values of the `colwidth` attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the `colwidth` attribute is `1*`.
- **dynamic** - If you choose this option, the columns are created without a specified width (`colwidth` attribute). Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in **Author** mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.
- **fixed** - The width is specified in fixed units. By default, the `pt` unit is inserted, but you can change the units in the **colspecs** (column specifications) section above the table or in **Text** mode. The following units are allowed: `pt` (points), `cm` (centimeters), `mm` (millimeters), `pi` (picas), `in` (inches).

Frame

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

Row separator

Specifies whether or not to include row separators (`rowsep` attribute). The allowed values are: `0` (no separator) and 1 (include separators).

Column separator

Specifies whether or not to include column separators (`colsep` attribute). The allowed values are: `0` (no separator) and 1 (include separators).

Alignment

Specifies the alignment of the text within the table (`align` attribute). The allowed values are:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width.

  Note: The `justify` value cannot be rendered in **Author** mode, so you will only see it in the output.
- **char** - Aligns text to the leftmost occurrence of the value specified on the `char` attribute for alignment.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular

attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

Note: The options in the **Insert Table** dialog box for DITA documents are persistent, so changes made in one session will carry over to another.

When you click **Insert**, a CALS table is inserted into your document at the current cursor position.

When you insert a CALS table, you see a link for setting the `colspecs` (column specifications) of your table. Click the link to open the controls that allow you to adjust various column properties. Although they appear as part of the ***Author** mode*, the `colspecs` link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

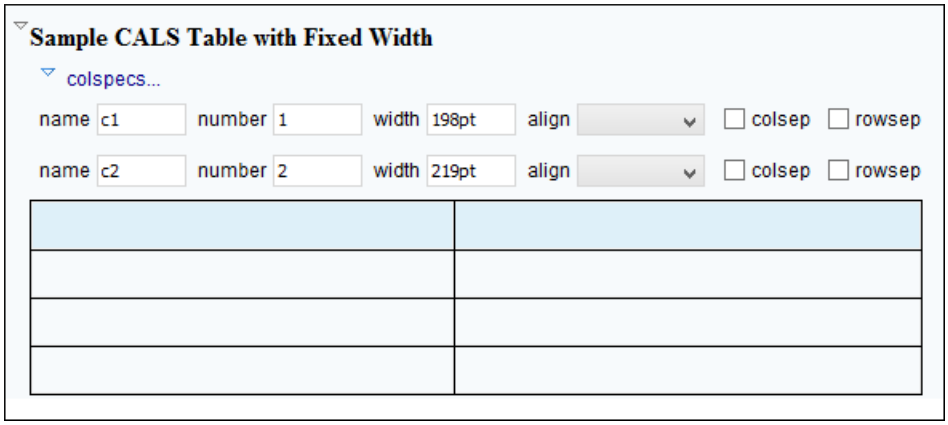


Figure 719: CALS Table in DITA

Inserting a Choice Table Model

To insert a **Choice** table within a `step` element in a DITA Task document, select the **Insert Table** action on the toolbar or in the **Insert** submenu from the contextual menu (or the **Table** submenu from the **DITA** menu), or select `choicetable` from the *Content Completion Assistant*. The **Insert Table** dialog box appears. Select **Simple** for the table **Model**.

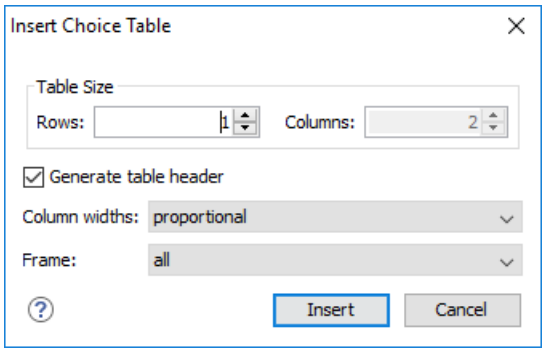


Figure 720: Insert Table Dialog Box - Choice Model

The dialog box allows you to configure the following options when you insert a *Choice* table model within a DITA Task:

Table Size

Allows you to choose the number of **Rows** and **Columns** for the table.

Generate table header

If selected, an extra row will be inserted at the top of the table to be used as the table header.

Column widths

Allows you to specify the type of properties for column widths (`colwidth` attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a `relcolwidth` attribute with the values listed as the number of shares followed by an asterisk. The value of the shares are totaled and rendered as a percent. For example, `relcolwidth="1* 2* 3*"` causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in **Author** mode, the values of the `relcolwidth` attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the `relcolwidth` attribute is `1*`.
- **dynamic** - If you choose this option, the columns are created without a specified width. Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in **Author** mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.

Frame

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

When you click **Insert**, a *Choice* table is inserted into your DITA Task document at the current cursor position (within a `step` element).

Inserting a Properties Table Model

To insert a **Properties** table within a `refbody` element in a DITA Reference document, select the **Insert Table** action on the toolbar or in the **Insert** submenu from the contextual menu (or the **Table** submenu from the **DITA** menu), or select `properties(wizard)` from the *Content Completion Assistant*. The **Insert Table** dialog box appears. Select **Properties** for the table **Model**.

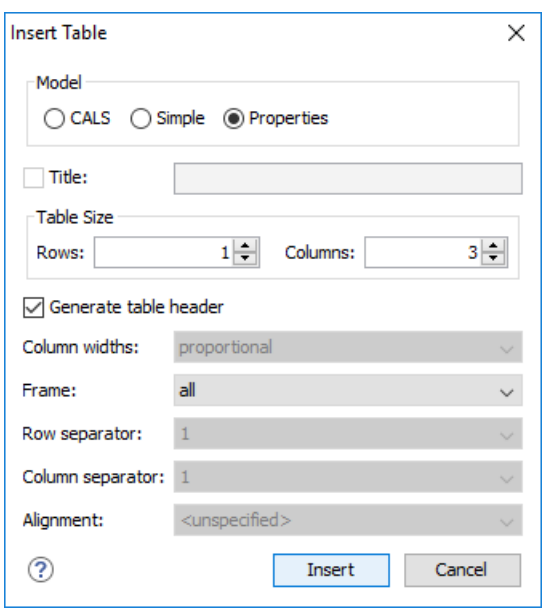


Figure 721: Insert Table Dialog Box - Properties Model

The dialog box allows you to configure the following options when you insert a *Properties* table model within a DITA Reference:

Table Size

Allows you to choose the number of **Rows** and **Columns** for the table.

Generate table header

If selected, an extra row will be inserted at the top of the table to be used as the table header.

Frame

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

When you click **Insert**, a *Properties* table is inserted into your DITA Reference document at the current cursor position (within a `refbody` element).

Editing an Existing Table

You can edit the structure of an existing table using the table buttons on the toolbar (or in the contextual menu) to add or remove cells, rows, or columns, and to set basic table properties. Additional attributes can be used to fine-tune the formatting of your tables by using the ***Attributes*** *view* (**Window** > **Show View** > **Attributes**). See the *DITA documentation* for a full explanation of these attributes.

You can also use the ***Table Properties (Ctrl + T (Command + T on OS X))*** action from the toolbar or contextual menu (or **DITA** menu) to *modify many of the properties of the table*.

Also, remember that underneath the visual representation, both table models are really just XML. If necessary, you can edit the XML directly by switching to ***Text*** *mode*.

Related Information:

Editing Tables in Author Mode on page 385

DITA Table Layouts

Depending on the context, DITA accepts the following table layouts:

- *CALS table model*
- *Simple table model*
- *Choice table model*
- *Properties table model*

CALS Table Model Layout

The **CALS** table model allows for more flexibility and table customization than other models. When choosing a **CALS** table model from the **Insert Table** dialog box, you have access to more configurable properties. The layout of a **CALS** table includes a **colspecs** section that allows you to easily configure some properties without opening the **Table Properties** dialog box. For example, you can change the value of column widths (`colwidth` attribute) or the text alignment (`align` attribute). Although they appear as part of the ***Author*** *mode*, the `colspecs` link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

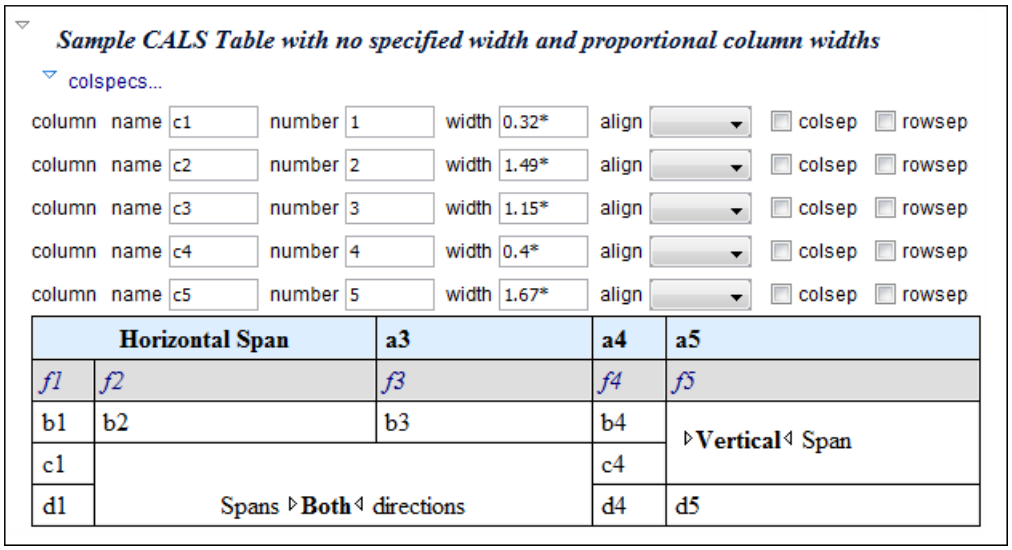


Figure 722: CALS Table in DITA

Simple Table Model Layout

When choosing a **Simple** table model from the **Insert Table** dialog box, you only have access to configure a few properties. For example, you can choose the number of rows and columns, specify values for frames, and choose from a few types of properties for the column width. The layout of this type of table is very simple, as the name suggests.

Header 1	Header 2
Column 1	Column 2

Figure 723: DITA Simple Table

Choice Table Model Layout

A **Choice** table model is used within a `step` element in a DITA Task document to describe a series of optional choices that a user must make before proceeding. The `choicetable` element is a useful device for documenting options within a single step of a task. You can insert *Choice* tables in DITA Task documents either by selecting `choicetable` from the *Content Completion Assistant* (within a `step` element) or by using the **Insert Table** action on the toolbar or from the contextual menu). The options and layout of a *Choice* table is similar to the *Simple* table model.

Option	Description
Opt A	
Opt B	
Opt C	

Figure 724: DITA Choice Table

Properties Table Model Layout

A **Properties** table model is used within a `refbody` element in a DITA Reference document to describe a property (for example, its type, value, and description). You can insert *Properties* tables in DITA Reference documents either by selecting `properties(wizard)` from the *Content Completion Assistant* (within a `refbody` element) or by using the **Insert Table** action on the toolbar (or from the contextual menu) and selecting **Properties** for the **Model**. The layout of a *Properties* table is very simple. It allows for a maximum of 3 columns (typically for property type, value, and description) and the only options available are for whether or not you want a header row and for specifying frames (borders).

Property Type	Property Value	Property Description
A	B	C

Figure 725: DITA Properties Table

Table Validation in DITA

Oxygen XML Editor reports table layout problems that are detected in manual or automatic validations. When you validate a *DITA map* with the **Validate and Check for Completeness** action, if the ***Report table layout problems*** *option is selected in the* ***DITA Map Completeness Check*** *dialog box*, table layout problems will be reported in the validation results. The types of errors that may be reported for DITA table layout problems include:

CALS Tables

- A row has fewer cells than the number of columns detected from the table `cols` attribute.
- A row has more cells than the number of columns detected from the table `cols` attribute.
- A cell has a vertical span greater than the available rows count.
- The number of `colspecs` is different than the number of columns detected from the table `cols` attribute.
- The number of columns detected from the table `cols` attribute is different than the number of columns detected in the table structure.
- The value of the `cols`, `rowsep`, or `colsep` attributes are not numeric.
- The `namest`, `nameend`, or `colname` attributes point to an incorrect column name.

Simple or Choice Tables

- A row has fewer cells than the number of table columns.

Editing Table Properties in DITA

To customize the look of a table in DITA, place the cursor anywhere in a table and invoke the **Table Properties (Ctrl + T (Command + T on OS X))** action from the toolbar or the **Table** submenu of the contextual menu (or **DITA** menu). This opens the **Table properties** dialog box.

The **Table properties** dialog box allows you to set specific properties to the table elements. The options that are available depend on the context and location within the table in which the action was invoked.

Note: Some properties allow the following special values, depending on the context and the current properties or values:

- **<not set>** - Use this value if you want to remove a property.
- **<preserve>** - If you select multiple elements that have the same property set to different values, you can choose this value to keep the values that are already set. In some cases it can also be used to keep the current non-standard value for a particular property.

Edit Table Properties for a CALS Table Model

For a **CALS** table model, the **Table properties** dialog box includes four tabs of options:

- **Table** tab - The options in this tab apply to the entire table.
- **Row** tab - The options in this tab apply to the current row or selection of multiple rows. A message at the bottom of the tab tells you how many rows will be affected.
- **Column** tab - The options in this tab apply to the current column or selection of multiple columns. A message at the bottom of the tab tells you how many columns will be affected.
- **Cell** tab - The options in this tab apply to the current cell or selection of multiple cells. A message at the bottom of the tab tells you how many cells will be affected.

The options in four tabs include a **Preview** pane that shows a representation of the modification.

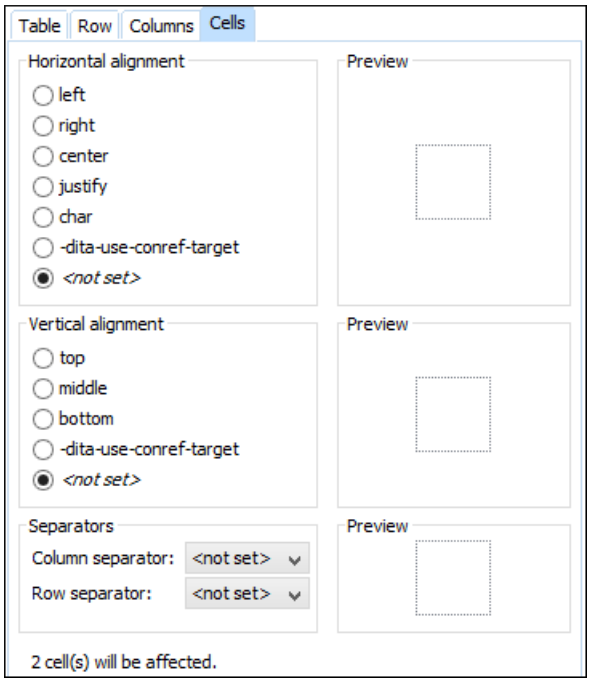


Figure 726: Table Properties Dialog Box with Cell Tab Selected (DITA CALS Table Model)

The options in the four tabs include the following:

Horizontal alignment (Available in the Table, Column, and Cell tabs)

Specifies the horizontal alignment of text within the current table/column/cell or selection of multiple columns/cells (`align` attribute). The allowed values are as follows:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width.

  Note: The `justify` value cannot be rendered in **Author** mode, so you will only see it in the output.
- **char** - Aligns text to the leftmost occurrence of the value specified on the `char` attribute for alignment.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

Vertical alignment (Available in the Row and Cell tabs)

Specifies the vertical alignment of text within the current row/cell or selection of multiple rows/cells (`valign` attribute). The allowed values are as follows:

- **top** - Aligns the text at the top of the cell.
- **middle** - Aligns the text in a vertically centered position.
- **bottom** - Aligns the text at the bottom of the cell.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

Column separator (Available in the Table, Column, and Cell tabs)

Specifies whether or not to include column separators (borders/grid lines) in the form of the `colsep` attribute. The allowed values are: `0` (no separator) and 1 (include separators).

Row separator (Available in all four tabs)

Specifies whether or not to include row separators (borders/grid lines) in the form of the `rowsep` attribute. The allowed values are: `0` (no separator) and 1 (include separators).

Frame (Available only in the Table tab)

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

Edit Table Properties for a Simple, Choice, or Properties Table Model

For a *Simple*, *Choice*, *Properties* table model, the **Table properties** dialog box only allows you to edit a few options.

Table tab

Frame

Allows you to specify a value for the `frame` attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a `conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `conref` target. For more information, see *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html*.

Row tab (not available for *Properties* tables)

Row type

Allows you change the row to a body or header type of row.

Related Information:

Adding Tables in DITA Topics on page 394
Editing Tables in Author Mode on page 385

Adding MathML Equations in DITA Topics

You can add MathML equations in a DITA document using one of the following methods:

- Embed MathML directly into a DITA topic. You can start with the **Framework templates / DITA / topic / Composite with MathML** document template, available from the **New** file action wizard.
- Reference an external MathML file as an image, using the **Insert Image** action that is available on the DITA toolbar (or from the **DITA** > **Insert** menu).

Note: MathML equations contained in DITA topics can only be published out-of-the-box in PDF using the **DITA PDF** transformation scenario. For other publishing formats, you must employ additional customizations for handling MathML content.

DITA Author Mode Actions

A variety of actions are available in the DITA *framework* that can be added to the **DITA** menu, the **Author Custom Actions** toolbar, the contextual menu, and the *Content Completion Assistant*.

DITA Toolbar Actions

The following default actions are readily available on the **DITA (Author Custom Actions)** toolbar when editing in **Author** mode (by default, most of them are also available in the **DITA** menu and in various submenus of the contextual menu):

Bold

Surrounds the selected text with a b tag. You can use this action on multiple non-contiguous selections.

Italic

Surrounds the selected text with an `i` tag. You can use this action on multiple non-contiguous selections.

Underline

Surrounds the selected text with a u tag. You can use this action on multiple non-contiguous selections.

Link Actions Drop-Down Menu

The following link actions are available from this menu:

Cross Reference

Opens the ***Cross Reference (xref)*** *dialog box* that allows you to insert a link to a target resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your *DITA map* structure. Once the target resource has been selected, you can also target specific elements within that resource. For more information, see *Linking in DITA Topics* on page 1674.

File Reference

Opens the ***File Reference*** *dialog box* that allows you to insert a link to a target file resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your *DITA map* structure. For more information, see *Linking in DITA Topics* on page 1674.

Web Link

Opens the ***Web Link*** *dialog box* that allows you to insert a link to a target web-related resource at the current location within a document. The target resource can be a URL or a key that is already defined in your *DITA map* structure. For more information, see *Linking in DITA Topics* on page 1674.

Related Link to Topic

Opens the ***Cross Reference (xref)*** *dialog box* that allows you to insert a link to a target resource in a related links section at the bottom of the current document. The target resource can be the location of a file or a key that is already defined in your *DITA map* structure. Once the target resource has been selected, you can also target specific elements within that resource. If a related links section does not already exist, this action creates one. For more information, see *Linking in DITA Topics* on page 1674.

Related Link to File

Opens the ***File Reference*** *dialog box* that allows you to insert a link to a target file resource in a related links section at the bottom of the current document. The target resource can be the location of a file or a key that is already defined in your *DITA map* structure. If a related links section does not already exist, this action creates one. For more information, see *Linking in DITA Topics* on page 1674.

Related Link to Web Page

Opens the ***Web Link*** *dialog box* that allows you to insert a link to a target web-related resource in a related links section at the bottom of the current document. The target resource can be a URL or a key that is already defined in your *DITA map* structure. If a related links section does not already exist, this action creates one. For more information, see *Linking in DITA Topics* on page 1674.

Insert Image

Opens the ***Insert Image*** *dialog box* that allows you to configure the properties of an image to be inserted into a DITA document at the cursor position.

Insert Media Resource

Opens the ***Insert Media*** *dialog box* that allows you to select and configure the properties of a media object to be inserted into a DITA document at the cursor position. The result will be that a reference to the specified video, audio, or embedded HTML frame is inserted in an `object` element and it is rendered in **Author** mode so that it can be played directly from there.

Insert Section Drop-Down Menu

The following insert actions are available from this menu:

Insert Section

Inserts a new `section` element in the document, depending on the current context.

Insert Concept

Inserts a new `concept` element, depending on the current context. Concepts provide background information that users must know before they can successfully work with a product or interface.

Insert Task

Inserts a new `task` element, depending on the current context. Tasks are the main building blocks for task-oriented user assistance. They generally provide step-by-step instructions that will help a user to perform a task.

Insert Topic

Inserts a new `topic` element, depending on the current context. Topics are the basic units of DITA content and are usually organized around a single subject.

Insert Reference

Inserts a new `reference` element, depending on the current context. A reference is a top-level container for a reference topic.

Insert Paragraph

Inserts a new paragraph at current cursor position.

Reuse Content

This action provides a mechanism for reusing content fragments. It opens the ***Reuse Content*** *dialog box* that allows you to insert several types of references to reusable content at the cursor position. The types of references that you can insert using this dialog box include *content references (`conref`)*, *content key references (`conkeyref`)*, or *key references to metadata (`keyref`)*.

Insert step or list item

Inserts a new list or step item in the current list type.

Insert Unordered List

Inserts an unordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an unordered list.

Insert Ordered List

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

Sort

Sorts cells or list items in a table.

Insert Table

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content — text + markup — that is rendered inside a *block element*) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

Note: If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables**.

Insert Row Below

Inserts a new table row with empty cells below the current row. This action is available when the cursor is positioned inside a table.

Delete Row(s)

Deletes the table row located at cursor position or multiple rows in a selection.

Insert Column After

Inserts a new table column with empty cells after the current column. This action is available when the cursor is positioned inside a table.

Delete Column(s)

Deletes the table column located at cursor position or multiple columns in a selection.

Table Properties

Opens the **Table properties** dialog box that allows you to configure properties of a table (such as frame borders).

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

DITA Menu Actions

In addition to the *DITA toolbar actions*, the following default actions are available in the **DITA** menu when editing in **Author** mode:

Reuse Content

This action provides a mechanism for reusing content fragments. It opens the ***Reuse Content*** *dialog box* that allows you to insert several types of references to reusable content at the cursor position. The types of references that you can insert using this dialog box include *content references (`conref`)*, *content key references (`conkeyref`)*, or *key references to metadata (`keyref`)*.

Push Current Element

Opens the ***Push current element*** *dialog box* that allows content from a source topic to be inserted into another topic without any special coding in the topic where the content will be re-used.

Edit Content Reference

This action is available for elements with a `conref` or `conkeyref` attribute. it opens the **Edit Content Reference** dialog box that allows you to edit the source location (or key) and source element of a content reference (or content key reference), and the reference details (`conref`/`conkeyref` and `conrefend` attributes). For more information, see *Reuse Content Dialog Box* on page 1659.

Replace Reference with content

Replaces the referenced fragment (`conref` or `conkeyref`) at the cursor position with its content. This action is useful if you want to make changes to the content in the currently edited document without changing the referenced fragment in its source location.

Remove Content Reference

Removes the content reference (`conref` or `conkeyref`) inside the element at the cursor position.

Create Reusable Component

Creates a reusable component from the selected fragment of text. For more information, see *Creating a Reusable Content Component* on page 1667.

Insert Reusable Component

Inserts a reusable component at cursor location. For more information, see *Inserting a Reusable Content Component* on page 1668.

Paste special submenu

This submenu includes the following special paste actions that are specific to the DITA *framework*:

Paste as content reference

Inserts a content reference (a DITA element with a `conref` attribute) to the DITA XML element from the clipboard. An entire DITA XML element with an ID attribute must be present in the clipboard when the action is invoked. The `conref` attribute will point to this ID value.

Paste as content key reference

Allows you to indirectly reference content using the `conkeyref` attribute. When the DITA content is processed, the key references are resolved using key definitions from *DITA maps*. To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.
2. In the **DITA Maps Manager** view, make sure that the ***Root map*** *combo box* points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the **DITA Maps Manager**, select **Edit Properties**, and enter a value in the **Keys** field.

Paste as link

Inserts a `link` element or an `xref` (depending on the location of the paste operation) that points to the DITA XML element from the clipboard. An entire DITA XML element with an ID attribute must be present in the clipboard when the action is invoked. The `href` attribute of `link/href` will point to this ID value.

Paste as link (keyref)

Inserts a link to the element that you want to reference. To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.
2. In the **DITA Maps Manager** view, make sure that the ***Root map*** *combo box* points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the **DITA Maps Manager**, select **Edit Properties**, and enter a value in the **Keys** field.

Table submenu

In addition to the table actions available on the toolbar, the following actions are available in this submenu:

Insert Row Above

Inserts a new table row with empty cells above the current row. This action is available when the cursor is positioned inside a table.

Insert Rows

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (**Above** or **Below** the current row).

Insert Column Before

Inserts a column before the current one.

Insert Columns

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (**Above** or **Below** the current column).

Insert Cell

Inserts a new empty cell depending on the current context. If the cursor is positioned between two cells, Oxygen XML Editor a new cell at cursor position. If the cursor is inside a cell, the new cell is created after the current cell.

Insert > Σ Insert Equation

Opens the **XML Fragment Editor** that allows you to insert and *edit MathML notations*.

ID Options

Opens the **ID Options** dialog box that allows you to configure options for generating IDs in **Author** mode. The dialog box includes the following:

ID Pattern

The pattern for the ID values that will be generated. This text field can be customized using constant strings or any of the Oxygen XML Editor *Editor Variables*.

Element name or class value to generate ID for

The elements for which ID values will be generated, specified using class attribute values. To customize the list, use the **Add**, **Edit**, or **Remove** buttons.

Auto generate IDs for elements

If selected, Oxygen XML Editor will automatically generate unique IDs for the elements listed in this dialog box when they are created in **Author** mode.

Remove IDs when copying content in the same document

When copying and pasting content in the same document, this option allows you to control whether or not pasted elements that are listed in this dialog box should retain their existing IDs. To retain the element IDs, deselect this option.

Note: This option does not have an effect on content that is *cut* and *pasted*.

Generate IDs

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements from the list in the **ID Options** dialog box that are found in the current selection.

Note: The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `id` attribute.

Browse DITA Style Guide

Opens the **DITA Style Guide Best Practices for Authors** in your browser and displays a topic that is relevant to the element at the cursor position. When editing DITA documents, this action is available in the contextual menu of the editing area (under the **About Element** sub-menu), in the **DITA** menu, and in some of the documentation tips that are displayed by the *Content Completion Assistant*.

Refresh References

You can use this action to manually trigger a refresh and update of all referenced resources.

Tags display mode Submenu

Full Tags with Attributes

Displays full tag names with attributes for both *block* and *inline elements*.

Full Tags

Displays full tag names without attributes for both *block* and *inline elements*.

Block Tags

Displays full tag names for *block elements* and simple tags without names for *inline elements*.

Inline Tags

Displays full tag names for *inline elements*, while *block elements* are not displayed.

Partial Tags

Displays simple tags without names for *inline elements*, while *block elements* are not displayed.

No Tags

No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Configure Tags Display Mode

Use this option to go to the ***Author*** *preferences page* where you can configure the **Tags Display Mode** options.

Profiling/Conditional Text Submenu

Edit Profiling Attributes

Allows you to configure the *profiling attributes* and their values.

Show Profiling Colors and Styles

Select this option to turn on conditional styling.

Show Profiling Attributes

Select this option to turn on conditional text markers. They are displayed at the end of conditional text blocks, as a list of attribute name and their currently set values.

Show Excluded Content

When this option is selected, the content filtered by the currently applied condition set is grayed-out. To show only the content that matches the currently applied condition set, deselect this option.

List of all profiling condition sets that match the current document type

You can click a listed *condition set* to activate it.

Profiling Settings

Opens the *profiling options preferences page*, where you can manage profiling attributes and profiling conditions sets. You can also configure the profiling styles and colors options from the *colors/styles preferences page* and the *attributes rendering preferences page*.

DITA Contextual Menu Actions

In addition to many of the *DITA toolbar actions* and the *general* ***Author*** *mode contextual menu actions*, the following DITA *framework*-specific actions are also available in the contextual menu when editing in **Author** mode:

Paste special submenu

This submenu includes the following special paste actions that are specific to the DITA *framework*:

Paste as content reference

Inserts a content reference (a DITA element with a `conref` attribute) to the DITA XML element from the clipboard. An entire DITA XML element with an ID attribute must be present in the clipboard when the action is invoked. The `conref` attribute will point to this ID value.

Paste as content key reference

Allows you to indirectly reference content using the `conkeyref` attribute. When the DITA content is processed, the key references are resolved using key definitions from *DITA maps*. To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.
2. In the **DITA Maps Manager** view, make sure that the ***Root map*** *combo box* points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the **DITA Maps Manager**, select **Edit Properties**, and enter a value in the **Keys** field.

Paste as link

Inserts a `link` element or an `xref` (depending on the location of the paste operation) that points to the DITA XML element from the clipboard. An entire DITA XML element with an ID attribute must be present in the clipboard when the action is invoked. The `href` attribute of `link/href` will point to this ID value.

Paste as link (keyref)

Inserts a link to the element that you want to reference. To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.
2. In the **DITA Maps Manager** view, make sure that the ***Root map*** *combo box* points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the **DITA Maps Manager**, select **Edit Properties**, and enter a value in the **Keys** field.

Image Map Editor

This action is available in the contextual menu when it is invoked on an image. This action applies an *image map* to the current image (if one does not already exist) and opens the **Image Map Editor** dialog box. This feature allows you to create hyperlinks in specific areas of an image that will link to various destinations.

Table Actions

The following table editing actions are available in the contextual menu when it is invoked on a tabl

e:

Insert Rows

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (**Above** or **Below** the current row).

Delete Row(s)

Deletes the table row located at cursor position or multiple rows in a selection.

Insert Columns

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (**Above** or **Below** the current column).

Delete Column(s)

Deletes the table column located at cursor position or multiple columns in a selection.

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

Sort

Sorts cells or list items in a table.

Table Properties

Opens the **Table properties** dialog box that allows you to configure properties of a table (such as frame borders).

Other Actions submenu

This submenu give you access to all the usual contextual menu actions.

Insert > Σ Insert Equation

Opens the **XML Fragment Editor** that allows you to insert and *edit MathML notations*.

Generate IDs

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements from the list in the **ID Options** dialog box that are found in the current selection.

Note: The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `id` attribute.

Reuse submenu

This submenu includes the following actions in regards to reusing content in DITA:

Push Current Element

Opens the ***Push current element*** *dialog box* that allows content from a source topic to be inserted into another topic without any special coding in the topic where the content will be re-used.

Edit Content Reference

This action is available for elements with a `conref` or `conkeyref` attribute. it opens the **Edit Content Reference** dialog box that allows you to edit the source location (or key) and source element of a content reference (or content key reference), and the reference details (`conref/conkeyref` and `conrefend` attributes). For more information, see *Reuse Content Dialog Box* on page 1659.

Replace Reference with content

Replaces the referenced fragment (`conref` or `conkeyref`) at the cursor position with its content. This action is useful if you want to make changes to the content in the currently edited document without changing the referenced fragment in its source location.

Remove Content Reference

Removes the content reference (`conref` or `conkeyref`) inside the element at the cursor position.

Create Reusable Component

Creates a reusable component from the selected fragment of text. For more information, see *Creating a Reusable Content Component* on page 1667.

Insert Reusable Component

Inserts a reusable component at cursor location. For more information, see *Inserting a Reusable Content Component* on page 1668.

Search References (Ctrl + Shift + G)

Finds the references to the `id` attribute value for the element at the current cursor position, in all the topics contained in the current *DITA map* (opened in the ***DITA Maps Manager*** *view*). If no references are found for the current element, a dialog box will be displayed that offers you the option of searching for references to its ancestor elements.

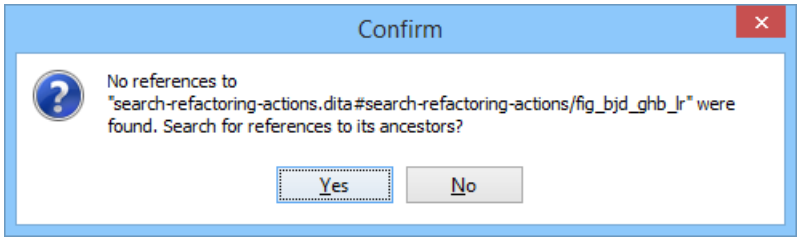


Figure 727: Search References to Ancestors Dialog Box

Show Key Definition

Available for elements that have a `conkeyref` or `keyref` attribute set (or elements with an ancestor element that has a `conkeyref` or `keyref` attribute). It computes the key name and opens the *DITA map* that contains the definition of the key with the element that defines that key selected.

About Element submenu

This submenu includes the following actions:

Style Guide

Opens the **DITA Style Guide Best Practices for Authors** in your browser and displays a topic that is relevant to the element at the cursor position. When editing DITA documents, this action is available in the contextual menu of the editing area (under the **About Element** sub-menu), in the **DITA** menu, and in some of the documentation tips that are displayed by the *Content Completion Assistant*.

Browse reference manual

Opens a reference to the documentation of the XML element closest to the cursor position in a web browser.

Show Definition

Moves the cursor to the definition of the current element.

DITA Drag/Drop (or Copy/Paste) Actions

Dragging a file from the ***Project*** *view* or ***DITA Maps Manager*** *view* and dropping it into a DITA document that is edited in **Author** mode, creates a link to the dragged file (the `xref` DITA element with the `href` attribute) at the drop location. Copy and paste actions work the same.

You can also drag images or media files from your system explorer or the ***Project*** *view* and drop them into a DITA document (or copy and paste). This will insert the appropriate element at the drop or paste location (for example, dropping/pasting an image will insert the `image` DITA element with the `href` attribute).

Working with Markdown Documents in DITA

Oxygen XML Editor includes some unique features that allow you to easily integrate Markdown documents in a DITA project. This is especially helpful for teams that have contributors who are familiar with the Markdown syntax, but they want their output to be generated from DITA projects. The integration between the Markdown editor and DITA includes actions to export or convert Markdown documents to DITA topics and the **DITA** tab in the *Preview* pane provides a visualization of how the topic will look after conversion.

Export Markdown as a DITA Topic

The Markdown editor includes an option to quickly convert the current Markdown document into a DITA topic. The **Export as DITA Topic** action is available in the contextual menu of the left-side text editor and the right-side *Preview* pane when the **DITA** tab selected.

The conversion creates a new XML file that is defined as a DITA topic and opens it in the **Text** editing mode. You can then work with the document as you would with any other DITA topic, although you may need to manually correct some issues where the parser could not properly map Markdown syntax to DITA markup.

Working with Markdown Documents in the DITA Maps Manager

Oxygen XML Editor has some specialized features that allow you to integrate Markdown documents directly into your DITA project using the ***DITA Maps Manager***. The following features are available for Markdown documents in the **DITA Maps Manager** view:

- **Insert Reference to Markdown Document** - You can use the **New**, **Reference**, and **Reference to the currently edited file** actions from the **Append Child**, **Insert Before**, or **Insert After** submenu when invoking the contextual menu in the **DITA Maps Manager** to insert a reference to a Markdown document at the selected location in the map. Markdown documents will be inserted as a topic reference (`topicref` element) with the `format` attribute set to `markdown`.
- **Validate Markdown Documents in DITA Maps** - When you use the **Validate and Check for Completeness** action from the **DITA Maps Manager** toolbar to check the integrity of the structure of a *DITA map*, Markdown documents that are referenced in the *DITA map* will be converted to DITA topics in the background and validated the same as any other DITA topic.
- **Transforming DITA Maps with Markdown Documents** - When transforming *DITA maps* that have Markdown documents referenced, the transformation will convert the Markdown documents to normal DITA output without you needing to manually convert the Markdown documents to DITA topics.
- **Manually Convert Markdown Documents to DITA Topics** - If you need to use DITA semantics that are not possible in Markdown syntax (such as content references, related links, and other DITA-specific syntax), you can manually convert the Markdown document into a DITA topic. To do so, right-click the Markdown document in the **DITA Maps Manager** and select **Refactoring** > **Convert Markdown to DITA Topic**. This will open a dialog

box that allows you to configure options for converting the document to an XML file that is defined as a DITA topic.

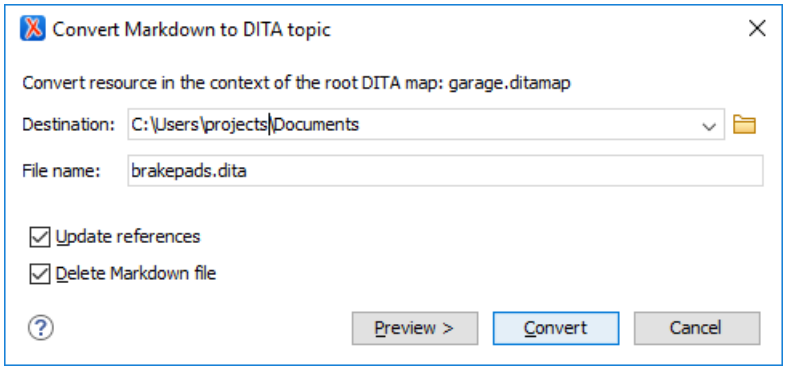


Figure 728: Convert Markdown to DITA Topic Dialog Box

This dialog box includes the following options:

Destination

The destination path for the new DITA topic.

File Name

Presents the current name and allows you to change it.

Update references

Select this option to update all references of the file in the *DITA map* and in the files referenced from the *DITA map*.

Delete Markdown file

If selected, the Markdown version of the file is deleted when the document is converted into a DITA file. If deselected (default value), when the document is converted into a DITA file, the original Markdown file is also preserved in its current location.

Preview

Select this button to display a preview of the changes Oxygen XML Editor is about to make.

Convert

Select this button to perform the conversion.

Tip: Oxygen XML Editor comes with a sample ditamap project for converting Markdown to DITA. Go to the ***Project*** *view*, open the `sample.xpr` project, and navigate to the `dita/markdown-dita` folder.

Related Information:

Markdown Editor on page 707
Actions Available in the Markdown Editor on page 709
Markdown Editor Syntax Rules and Specifications on page 715
Automatic Validation in Markdown Documents on page 714

Working with Keys

DITA uses keys to insert content that may have different values in various circumstances. Keys provide a way to reference something indirectly. This can make it easier to manage and to reuse content in a number of ways.

You can think of keys as like renting a post office box. Instead of the mail going directly from the sender to your house, it now goes to the post office box. You then go to the post office box and bring the mail back to your house. If you move to a new house, your mail still gets to you because it comes to the same post office box. You do not have to send change of address cards to all the people who send you mail. Your mailbox address is the key that makes sure your mail always reaches you, even if you move.

Similarly, if you use keys in your content to reference other content, you do not have to update the source content to change the value of the key or what it points to. You just change the *definition of the key*.

Using Keys for Values

You can use keys to represent values that may vary depending on the type of output. For instance, you may have several products that share a common feature. When you want to describe that feature, you need a way to insert the name of the product, even though that name is different depending on which product the feature description is being used for. For more information, see *Working with Variable Text in DITA* on page 1668.

Assigning Keys to Topics

You can assign a key to a topic and use that key to reference that topic for various purposes, such as reuse or linking. As always, keys are defined in maps, so the key definition is done using the keys attribute of the `topicref` element:

```
<topicref href="quick-heat.dita" keys="feature.quick-heat"/>
```

The easiest way to assign keys to a topic (and insert the `topicref` element in its *DITA map*) is to use the ***Keys** tab in the **Edit Properties** dialog box*. In the ***DITA Maps Manager***, invoke the contextual menu on the topic for which you want to assign a key and select **Edit Properties**. Go to the **Keys** tab and enter the name of the key in the **Define keys** field.

Once a key is assigned to a topic, you can use it to reference that topic for various purposes:

- You can *create a link* to it using `<xref keyref="feature.quick-heat"/>`. This allows you to change the target of the link by changing the topic that is pointed to by the key (for example, by profiling).
- You can use it *in a map to create a reference to a topic* by key: `<topicref keyref="feature.quick-heat"`. This allows you to change which topic is inserted in the map by the build, by changing the topic that is pointed to by the key.
- You can use it to *insert a content reference*. In this case, the content reference uses the key to locate the topic to pull content from. It uses a `conkeyref` attribute: `<procedure conkeyref="feature.quick-heat/preheat-procedure"/>`. In this example, `feature.quick-heat` is the key, and `preheat-procedure` is the ID of a procedure within the topic for that key. Using this mechanism, you could have multiple versions of the preheat procedure in various topics and control which one is inserted by changing the topic that is pointed to by the key.

Assigning Keys to Graphics

You can assign a key to an image (using a map to point to the image file) and *insert the image using the key*.

Related Information:

Defining Keys in DITA Maps on page 1602
Creating a DITA Content Key Reference on page 1657
Reuse Content Dialog Box on page 1659
DITA Reusable Components View on page 1671

Reusing DITA Content

Reusing content is one of the key features of DITA. DITA provides several methods for reusing content. Oxygen XML Editor provides support for each of these methods.

Reusing Topics in DITA Maps

A DITA topic does not belong to any one publication. You add a DITA topic to a publication by referencing it in a map. You can reference the same topic in multiple maps.

Reusing Content with References and Keys

DITA allows you to reuse content by referencing it in another topic. DITA provides two mechanisms for including content by reference: `conref` and `conkeyref`. A `conref` creates a direct reference to a specific element of another topic. A `conkeyref` creates a reference to a key, which then points to a specific element in another topic. The advantage of using a `conkeyref` is that you can change the element that is included by changing the

key reference. For example, since keys are defined in maps, if you include a topic in multiple maps, you can use a different key reference in each map.

Oxygen XML Editor provides support for both `conref` and `conkeyref` mechanisms.

While the `conref` and `conkeyref` mechanisms can be used to reference any content element, it is considered best practice to only *conref* or *conkeyref* content that is specifically set and managed as reusable content. This practice helps reduce expensive errors, such as an author accidentally deleting the source element that other topics are including by `conref`. Oxygen XML Editor can help you create a reusable component from your current content.

Reusing Content with Reusable Components

DITA allows you to select content in a topic, create a reusable component from it and reference that component in other locations. Each reusable component is created as a separate file. Anytime the content needs to be edited, you only need to update it in the component file and all the locations in your topics that reference it will also be updated. This can help you to maintain continuity and accuracy throughout your documents.

Reusing Content with Variables

DITA allows you to replace the content of certain elements with a value that is pointed to by a key. This mechanism effectively means that you can create variables in your content, which you can then create multiple outputs by changing the value that the key points to. This is done by profiling the definition of the key value, or by substituting another map with a different key value.

Reusing Content with DITA 1.3 Concepts

DITA 1.3 allows you to use some advanced concepts to expand content reuse possibilities even further. *Key Scopes* (or scoped keys) allow you to reuse topics with variable content depending on the particular context and it maximizes reuse possibilities for keys. *Branch Filtering* allows you to reuse the same content that is profiled in multiple ways within the same publication, each time using a different filter.

Related Information:

Working with Keys on page 1652

Reusing DITA Topics in Multiple Maps

You can reuse an entire DITA topic simply by referencing it in multiple maps (or *multiple locations within the same map*) using one of the following procedures:

Reuse Topics Using the DITA Maps Manager

1. Make sure the *DITA map* is opened in the ***DITA Maps Manager***.
2. Add a reference to an existing topic by using one of the following methods (depending on your particular situation):
   a. If the topic already exists in this *DITA map*, do one of the following:
      - Simply drag the topic and press **Ctrl** (or **Alt** on OS X) at the new location within the map (or use the **Copy** and **Paste** contextual menu actions).
      - If the topic is the currently opened document in the main editor, determine the new location in the map (in the ***DITA Maps Manager***), right-click a parent or sibling topic, and select **Append Child** > **Reference to the currently edited file** or **Insert After** > **Reference to the currently edited file**.
   b. If the topic already exists in another *DITA map*, do one of the following:
      - Open the other map in the ***DITA Maps Manager***, right-click the topic, select **Copy**, switch back to the original *DITA map* in the **DITA Maps Manager**, determine the new location in the map, right-click a parent or sibling topic, and use one of the **Paste** contextual menu actions (**Paste**, **Paste Before**, or **Paste After**).

- If the topic is the currently opened document in the main editor, determine the new location in the map (in the ***DITA Maps Manager***), right-click a parent or sibling topic, and select **Append Child** > **Reference to the currently edited file** or **Insert After** > **Reference to the currently edited file**.

c. If the topic exists in the project, but has not yet been added to a *DITA map*, do one of the following:

- Right-click the topic in the ***Project*** *view* (or the file system), select **Copy**, switch to the ***DITA Maps Manager*** view, determine the new location in the map, right-click a parent or sibling topic, and use one of the **Paste** contextual menu actions (**Paste**, **Paste Before**, or **Paste After**).
- If the topic is the currently opened document in the main editor, determine the new location in the map (in the ***DITA Maps Manager***), right-click a parent or sibling topic, and select **Append Child** > **Reference to the currently edited file** or **Insert After** > **Reference to the currently edited file**.

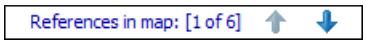
3. If your topic uses a *key reference*, set up the appropriate *key definition in your map*.
4. If you want to define relationships between topics, other than those defined in the topics themselves, you can *add a relationship table to your map*.
5. When you have finished adding topics, check that your map is complete and that all topic links and keys resolve correctly. To do this *validation, click the* ***Validate and Check for Completeness*** *action* on the toolbar in the **DITA Maps Manager**.

Reuse Topics Using Author Mode Editor

1. *Open the DITA map* in the **Author** mode editor.
2. Add a reference to an existing topic by dragging it from the ***Project*** *view* (or the file system) and dropping it in the desired location in the *DITA map* opened in **Author** mode. You can also accomplish the same thing by using the **Copy** and **Paste** contextual menu actions.
3. If your topic uses a *key reference*, set up the appropriate *key definition in your map*.
4. If you want to define relationships between topics, other than those defined in the topics themselves, you can *add a relationship table to your map*.
5. When you have finished adding topics, check that your map is complete and that all topic links and keys resolve correctly. To do this *validation, click the* ***Validate and Check for Completeness*** *action* on the toolbar in the **DITA Maps Manager**.

Displaying Multiple References to the Same Topics

Whenever multiple references to the same topic are detected in the context of the current map in the ***DITA Maps Manager***, an indicator will appear in the top-right corner of the **Author** mode editor that shows the number of times the current topic is referenced in the *DITA map*. It also includes navigation arrows that allow you to jump to the next or previous reference in the **DITA Maps Manager**.



Working with Content References

The DITA `conref` feature (short for *content reference*) lets you insert a piece of source content by referencing it from its source. When you need to update that content, you only need to do it in one place.

There are several strategies for managing content references:

- *Reusable components* - With this strategy, you create a new file for each piece of content that you want to reuse and you insert references from the content of the reusable component files. For example, suppose that you have a disclaimer that needs to be included in certain sections of your documentation. You can create a reusable component that contains your disclaimer and reuse it as often as you need to. If the disclaimer ever needed to be updated, you only have to edit it in one file.
- *Single-source content references* - You may prefer to keep many pieces of reusable content in one file. For example, you might want to create a single file that contains all the actions that are available in various menus or toolbars for your software application. Then, wherever you need to describe or display an action in your documentation, you can reuse content from that single file by inserting content references. This strategy requires more setup than reusable components, but might make it easier to centrally managing the reused content and it allows for more flexibility in the XML structure of the reusable content.

- *Arbitrary content references* - Although it is not recommended, you can create content references amongst topics without storing the reusable content in components or a single file. This strategy might make it difficult to manage content that is reused and to maintain continuity and accuracy, since you may not have any indication that content you are editing is reused elsewhere.

Oxygen XML Editor creates a reference to the external content by adding a `conref` attribute to an element in the local document. The `conref` attribute defines a link to the referenced content, made up of a path to the file and the topic ID within the file. The path may also reference a specific element ID within the topic. Referenced content is not physically copied to the referencing file. However, by default Oxygen XML Editor displays it in **Author** mode as if it is there in the referencing file. If you do not want referenced content displayed, *open the **Preferences** dialog box **(Options** > **Preferences)***, go to **Editor** > **Edit modes** > **Author**, and deselect the ***Display referenced content** option*.

Note: A reference also displays *tracked changes* and comments that are included in the source fragment. To edit these comments (or accept/reject changes) right-click the comment or tracked change and select **Edit Reference**.

Tip: To search for references made through a direct content references, use the **Search References** action from the contextual menu.

Related Information:

Working with Reusable Components on page 1667
Working with Keys on page 1652
Working with the Conref Push Mechanism on page 1665

Creating a DITA Content Reference

DITA Content Reference

A DITA content reference, or `conref`, is one of the main *content reuse features of DITA*. It is a mechanism for re-using the same content in multiple topics (or even in multiple locations within the same topic).

For a `conref` to be created, the source content must have an `id` attribute that the `conref` can reference. Therefore, creating a `conref` requires that you add an `id` to the content to be reused before inserting a `conref` into the topic that reuses the referenced content.

Assigning an *ID* to the Referenced Content

To add an `id` to a DITA element in a topic, place the cursor on the element and select **Edit Attributes** from the contextual menu (or simply press **Alt+Enter**) to open the *in-place attribute editor*. Enter `id` as the **Name** of the attribute and a value of your choice in the **Value** field. You can also use *the **Attributes** view* to enter a value in the **id** attribute.

Note: The element may already have an `id`, since in some cases Oxygen XML Editor automatically generates an *ID* value when the `id` attribute is created.

Creating a Content Reference

To create a content reference (`conref`), follow these steps:

1. Make sure the element you want to reference has an *ID assigned to it*.
2. In ***Author** mode*, place the cursor at the location where you want the reused content to be inserted.
3. Select the **Reuse Content** action on the main toolbar (or from the **DITA** menu or **Reuse** submenu of the contextual menu). The ***Reuse Content** dialog box* is displayed.
4. In the **Location** field of the **Reuse Content** dialog box, select the topic that contains the element you want to reference. The elements that you can reference are presented in a table.
5. Select the **Target ID** of the element (or elements) from which you want to insert the content, and verify the content in the **Preview** pane. The `id` value of the element that you select is automatically added to the **Reference to (conref)** field.
6. Make any other selections you need in the ***Reuse Content** dialog box*. If you select multiple elements, the **Expand to (conrefend)** field is automatically filled with the `id` value of the last element in your selection.

7. Click **Insert** or **Insert and close** to create the content reference.

Using Copy/Paste Actions to Create a Content Reference

Oxygen XML Editor also includes support for creating content references with simple copy/paste actions. The copied content must be an entire DITA XML element with an ID attribute. Also, the location in the document where you paste the element must be valid, although as long as the ***Smart paste and drag and drop*** *option* is selected in the **Schema Aware** preferences page, if you try to paste it in an invalid location, Oxygen XML Editor will attempt to place it in a valid location, and may prompt you with one or more choices for where to place it.

To create a content reference (`conref`) using copy/paste actions, follow these steps:

1. Copy an entire DITA element that has an ID attribute assigned to it.
2. Place the cursor at a location where the copied element will be valid.
3. Select the **Paste as Content Reference** action from the **Paste Special** submenu from the contextual menu.

Other Ways to Reuse Content

An alternate way to reuse content is to use the Oxygen XML Editor ***Create Reusable Component*** and ***Insert Reusable Component*** actions (available in the **DITA** menu and the **Reuse** submenu of the contextual menu). They handle the details of creating an *ID* and *conref* and creates reusable component files, separate from your normal content files. This can help you manage your reusable content more effectively.

You can also *insert reusable content using content key references*. This may also make reusable content easier to manage, depending on your particular situation and needs.

Related Information:

Reuse Content Dialog Box on page 1659
Working with Reusable Components on page 1667
Editing DITA Content References on page 1658

Creating a DITA Content Key Reference

DITA Content Key Reference

A DITA content key reference, or `conkeyref`, is a mechanism for inserting a piece of content from one topic into another. It is a version of the *DITA content reference mechanism* that uses *keys* to locate the content to reuse rather than direct references to topics that contain reused content.

As with a *conref*, a *conkeyref* requires that the element to be reused has an `id` attribute. It also requires the topic that contains the reusable content to be assigned a *key* in a map. As with all uses of keys, you can substitute multiple maps or *use profiling* to create multiple definitions of keys in a single map. This allows the same `conkeyref` to pull in content from various sources, depending on how your build is configured. This can make it easier to create and manage sophisticated content reuse scenarios.

Creating a Content Key Reference

To create a content key reference (`conkeyref`), follow these steps:

1. Make sure the topic that contains the reusable content is assigned a key in the *DITA map* and the element you want to reference has an *ID* assigned to it.
2. In ***Author*** *mode*, place the cursor at the location where you want the reused content to be inserted.
3. Select **Reuse Content** on the main toolbar (or from the **DITA** menu or **Reuse** submenu of the contextual menu). The ***Reuse Content*** *dialog box* is displayed.
4. Select the **Key** radio button for the content source and use the **Choose Key Reference** button to select the key for the topic that contains the reusable content (you can also select one from the drop-down list in the **Key** field). The elements that you can reference from the source are presented in the table in the middle of the **Reuse Content** dialog box.
5. Select the **Target ID** of the element (or elements) that you want to insert, and verify the content in the **Preview** pane. The `id` value of the element that you select is automatically added to the **Reference to (conkeyref)** field.

6. Make any other selections you need in the ***Reuse Content*** *dialog box*. If you select multiple elements, the **Expand to (conrefend)** field is automatically filled with the `id` value of the last element in your selection.
7. Click **Insert** or **Insert and close** to create the content reference.

Note: If you are using ***Text*** *mode*, when you insert a `conkeyref` attribute, after you enter the first quote (`conkeyref="`), the **Content Completion Assistant** will list all the defined keys that you can select from. Also, after you select the key, the **Content Completion Assistant** will then list the element IDs from the referenced topic, allowing you to insert an anchor. Note that this only works for local files.

Using Copy/Paste Actions to Create a Content Key Reference

Oxygen XML Editor also includes support for creating content key references with simple copy/paste actions. When the DITA content is processed, the key references are resolved using key definitions from *DITA maps*. The copied content must be an entire DITA XML element with an ID attribute and the topic that contains the reusable content must have a key assigned in a *DITA map*. Also, the location in the document where you paste the element must be valid, although as long as the ***Smart paste and drag and drop*** *option* is selected in the **Schema Aware** preferences page, if you try to paste it in an invalid location, Oxygen XML Editor will attempt to place it in a valid location, and may prompt you with one or more choices for where to place it.

To create a content key reference (`conkeyref`) using copy/paste actions, follow these steps:

1. In the ***DITA Maps Manager*** *view*, make sure that the ***Root map*** *combo box* points to the correct map that stores the keys.
2. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the ***DITA Maps Manager***, select **Edit Properties**, and enter a value in the **Keys** field.
3. In a topic with an assigned key, copy an entire DITA element that has an ID attribute assigned to it.
4. Place the cursor at a location where the copied element will be valid.
5. Select the **Paste as Content Key Reference** action from the **Paste Special** submenu from the contextual menu.

Related Information:

Reuse Content Dialog Box on page 1659
Working with Reusable Components on page 1667
Editing DITA Content References on page 1658

Editing DITA Content References

Oxygen XML Editor also includes some actions that allows you to quickly edit existing content references. When the element that contains a content reference (`conref` or `conkeyref`) is selected, the following actions are available in the **DITA** menu and the **Reuse** submenu of the contextual menu:

Edit Content Reference

This action is available for elements with a `conref` or `conkeyref` attribute. it opens the **Edit Content Reference** dialog box that allows you to edit the source location (or key) and source element of a content reference (or content key reference), and the reference details (`conref/conkeyref` and `conrefend` attributes). For more information, see *Reuse Content Dialog Box* on page 1659.

Replace Reference with content

Replaces the referenced fragment (`conref` or `conkeyref`) at the cursor position with its content. This action is useful if you want to make changes to the content in the currently edited document without changing the referenced fragment in its source location.

Remove Content Reference

Removes the content reference (`conref` or `conkeyref`) inside the element at the cursor position.

Converting Conrefs to Conkeyrefs

Oxygen XML Editor includes a DITA refactoring operation called **Convert conrefs to conkeyrefs** that will find all *content references* (that reference content outside the current document) and convert them to *content key references*. You can also use it to quickly convert all *content references* in the current document or multiple documents at once.

To access the **Convert conrefs to conkeyrefs** operation, use one of the following methods:

Single Document Method

With the document opened in the editor, right-click anywhere in the main editing pane (or right-click the topic reference in the ***DITA Maps Manager***), go to the **Refactoring** submenu, and choose **Convert conrefs to conkeyrefs**.

Multiple Documents At Once Method

Select **XML Refactoring** from the **Tools** menu (or from the **Refactoring** submenu when you right-click a document in the ***Project*** *view*or the ***DITA Maps Manager*** *view*). Then select **Convert conrefs to conkeyrefs** from the **DITA** section and click **Next**.

Either method will proceed to the **XML Refactoring Wizard**. If you used the ***Multiple Documents At Once Method***, the wizard page allows you to choose a scope for the operation and some filtering options:

- **Scope** - Select from a variety of options to define the scope for which resources will be affected by the operation. For example, you can choose to affect all resources in the **Project**, **All opened files**, **Current DITA map hierarchy**, or just the **Current file**.
- **Filters** section
  - **Include files** - Specifies files to be excluded from the operation. You can specify multiple files by separating them with commas and the patterns can include wildcards (such as ***** or **?**).
  - **Restrict to known XML file types only** - Excludes non-XML file types from the operation.
  - **Look inside archives** - If this option is selected, the scope of the operation will include files inside archives.

If you used the ***Single Document Method***, the scope will be the current file so the scope and filtering options are not displayed.

You can then use one of the following buttons to proceed with the operation:

Preview

You can use the **Preview** button to open a comparison panel where you can review all the changes that will be made by the refactoring operation before applying the changes.

Warning: It is always recommended to use the **Preview** button to make sure the operation is not going to do something unexpected and after you click the **Finish** button, any **Undo** action will only revert changes on the current document.

Finish

When you use the **Finish** button, the operation will be processed and all *content references* will be converted to *content key references* (either all *content references* in the current document or all *content references* in all of the documents specified in the scope). The file name for each converted document is used as the value for its new key. However, the operation does NOT automatically add the key to the *DITA Map*, so you still need to manually *define each key in your DITA map*.

Related Information:

Creating a DITA Content Reference on page 1656
Creating a DITA Content Key Reference on page 1657
Defining Keys in DITA Maps on page 1602

Reuse Content Dialog Box

The **Reuse Content** dialog box provides a mechanism for reusing content fragments. DITA `conref`, `conkeyref`, and `keyref` attributes can be used to insert references to reusable content. The `conref` attribute stores a reference to another element and is processed to replace the referencing element with the referenced element. The `conkeyref` attribute uses *keys* to locate the content to reuse rather than direct references to the topic that contains the reusable content. The `keyref` attribute also uses *keys* and can be used to indirectly reference metadata that may have different values in various circumstances.

Note: For a `conref` or `conkeyref`, to reference the content inside a DITA element, the source element must have an `id` attribute assigned to it. The element containing the content reference acts as a placeholder for the referenced element. For more details about DITA `conref` and `conkeyref` attributes, go to *https://www.oxygenxml.com/dita/1.3/specs/archSpec/base/conref.html*.

Note: For the purposes of using a `keyref`, keys are defined at map level and referenced afterwards. For more information about the DITA `keyref` attribute, go to *https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/thekeyrefattribute.html*.

Oxygen XML Editor *displays the referenced content* of a DITA content reference if it can resolve it to a valid resource. If you use URIs instead of local paths in your XML documents and your DITA OT transformation needs an *XML Catalog* to map the URIs to local paths, you need to *add the catalog to Oxygen XML Editor*. If the URIs can be resolved, the referenced content is displayed in **Author** mode and in the transformation output.

In **Author** mode, a references to reusable content (`conref`, `conkeyref`, or `keyref`) can easily be inserted at the cursor position by using the **Reuse Content** dialog box. It can be opened with any of the following methods:

- Go to **DITA** > **Reuse Content**.
- Click the **Reuse Content** action on the main toolbar.
- In the contextual menu of the editing area, go to **Reuse** > **Reuse Content**.

Your selection at the top of the dialog box for choosing the content source determines whether Oxygen XML Editor will insert a `conref`, `conkeyref`, or `keyref`.

If you select **Location** for the content source, a *content reference* (`conref`) will be inserted. If you select **Key** for the content source, keys will be used to insert a *content key reference* (`conkeyref`) or a *key reference* (`keyref`).

Content Reference (`conref`) Options Using the Reuse Content Dialog Box

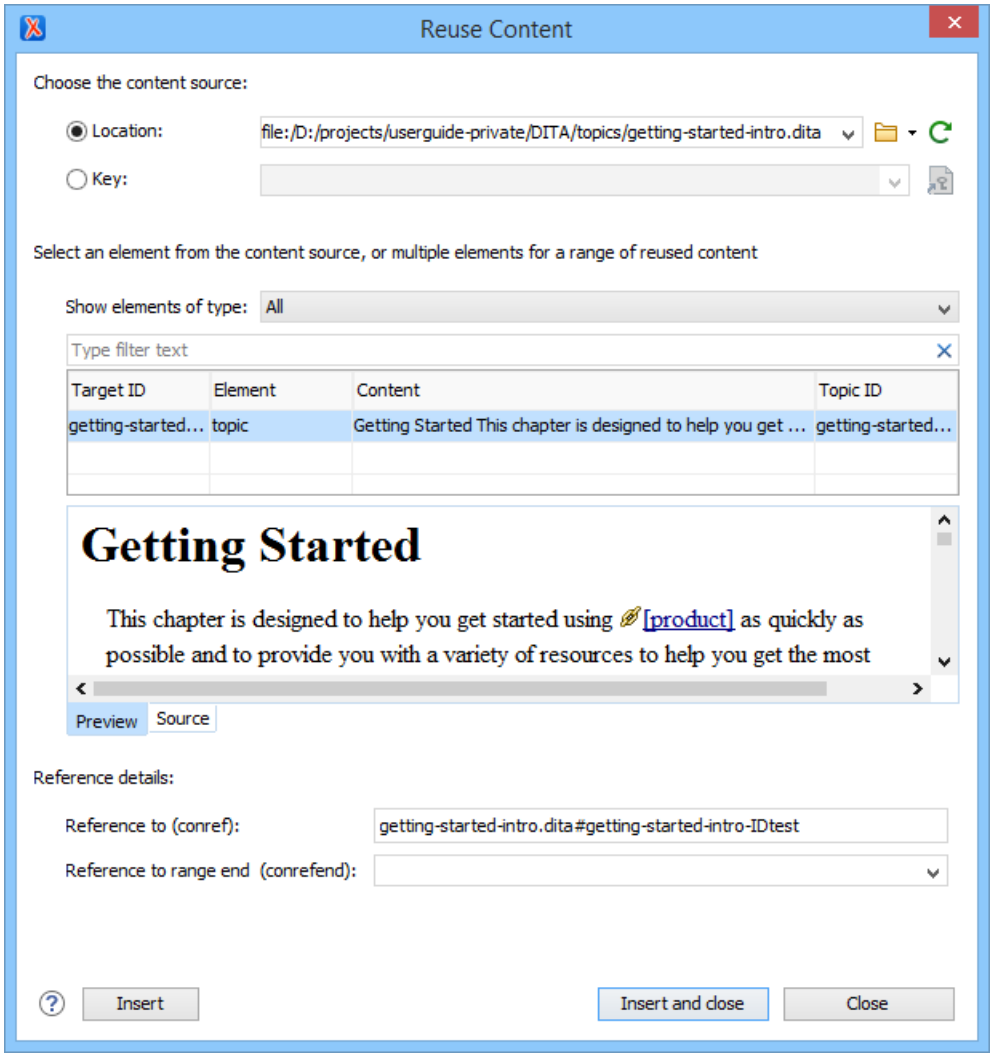


Figure 729: Reuse Content Dialog Box (with the Default Insert Content Reference Options Displayed)

Choose the content source Section

When **Location** is selected for the content source, a *content reference* (`conref`) will be inserted. Here you can specify the path of the topic that contains the content you want to reference.

The dialog box offers the following options:

Select an element from the content source Section

Show elements of type

You can use this drop-down list to select specific types of elements to be displayed in the subsequent table. This can help you narrow down the list of possible source elements that you can select.

Text Filter Field

You can also use the text filter field to narrow down the list of possible source elements to be displayed in the subsequent table.

Element Table

Presents all the element IDs defined in the source topic. Use this table to select the **Target ID** of the element that you want to reference. You can select multiple contiguous elements to reference a block of content.

Preview Pane

Displays the content that will be references. If you select multiple elements in the element table, the content from all the selected elements is displayed.

Source Pane

Displays the source code of the element to be referenced.

Reference details Section

Reference to (conref)

Oxygen XML Editor automatically fills this text field with the value of the `conref` attribute to be inserted. However, you can edit this value if need be.

Reference to range end (conrefend)

If you select multiple elements (of the same type) in the element table, Oxygen XML Editor automatically fills this text field with the `id` value of the last element in your selection. This value will be inserted as a `conrefend` attribute, defining the end of the `conref` range.

Content Key Reference (`conkeyref`) Options Using the Reuse Content Dialog Box

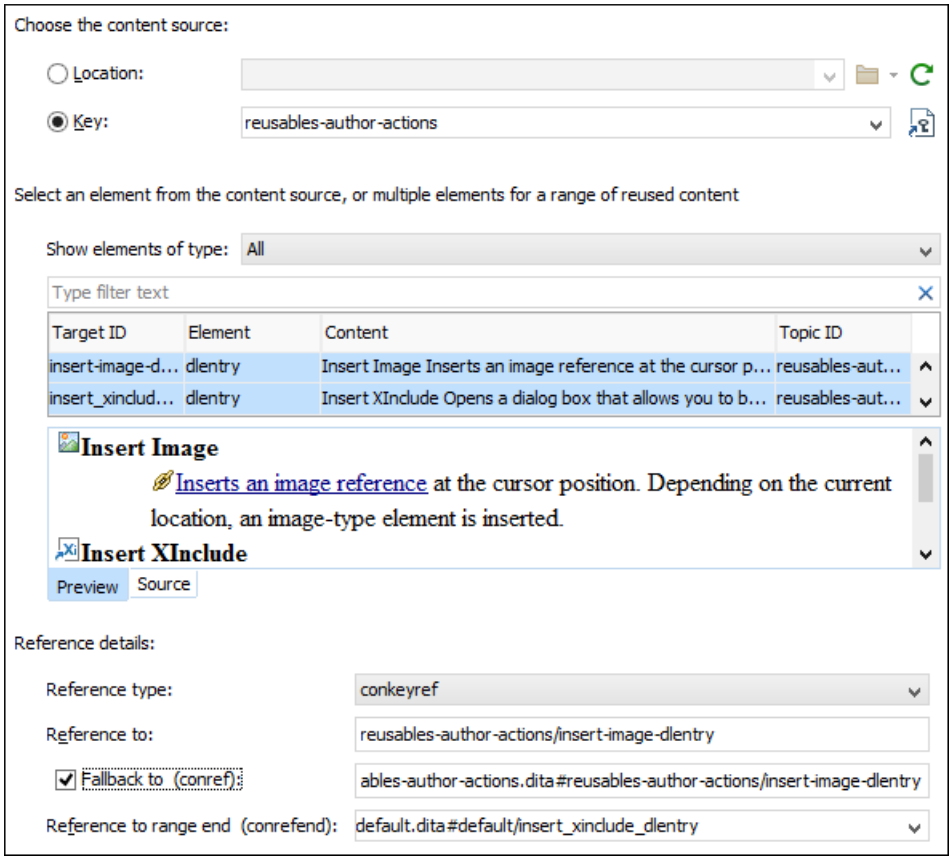


Figure 730: Insert Content Key Reference Options

Choose the content source Section

When **Key** is selected for the content source, you can use keys to reference content. You can use the **Choose Key Reference** button to open the **Choose Key** dialog box that allows you to select one from a list of all the keys that are gathered from the *root map* (you can also select one from the drop-down list in the **Key** field).

Note: If the current *DITA map* is not selected as the *root map*, no keys will be listed.

Tip: You can also use the ***DITA Reusable Components*** *view* for similar purposes.

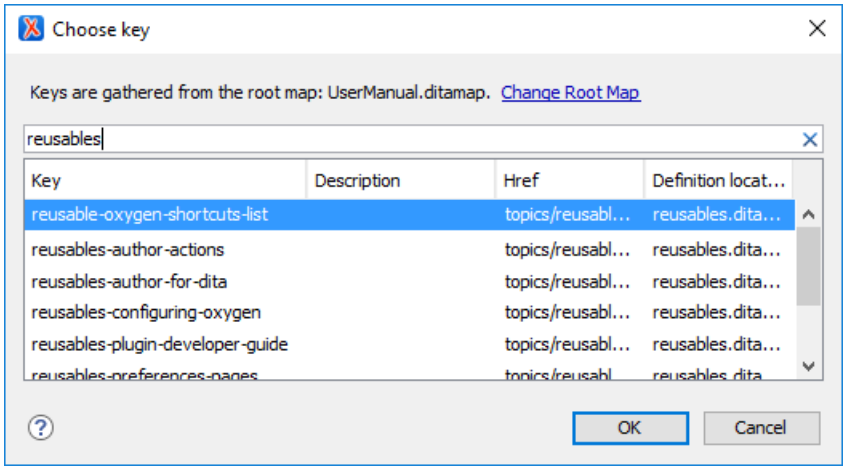


Figure 731: Choose Key Dialog Box

The **Choose Key** dialog box includes the following:

- **Change Root Map** - Opens a small dialog box that allows you to *select a root map*.

- **Search Filter** - You can enter text in the filter field at the top of the dialog box to filter the list and search for a specific keys.
- **Sortable Columns** - The dialog box includes the following columns that can be sorted by clicking on the heading:
  - **Key** - The name of the key.
  - **Description** - The description of the key that is obtained from the definition of the key (for example, the description for a *keydef* would be found in the `keyword` element inside *topicmeta*, or the description for a *topicref* would be found in a `navitle` element). For more information about where the data in this column is collected, see *this note*.
  - **Href** - The value of the *href* where the key points to.
  - **Definition Location** - The name of the *DITA map* where the key is defined.
- **Group by Definition Location** - A contextual menu action that can be used to group (and sort) all the keys based upon the value in the **Definition Location** column.

To insert a *content key reference* (`conkeyref`), select the key that contains the content you want to reference. Notice that the file path is shown in the **Href** column. Keys that do not have a value in the **Href** column are for referencing metadata with a `keyref` attribute. Therefore, to insert a `conkeyref`, you need to select a key that does have a value (file path) in the **Href** column.

After you select a key, click **OK** to return to the **Reuse Content** dialog box.

When a key that is defined as a *content key reference* has been selected, the **Reuse Content** dialog box offers the following additional options for inserting a `conkeyref`:

Select an element from the content source Section

Show elements of type

You can use this drop-down list to select specific types of elements to be displayed in the subsequent table. This can help you narrow down the list of possible source elements that you can select.

Text Filter Field

You can also use the text filter field to narrow down the list of possible source elements to be displayed in the subsequent table.

Element Table

Presents all the element IDs defined in the source topic. Use this table to select the **Target ID** of the element that you want to reference. You can select multiple contiguous elements to reference a block of content.

Preview Pane

Displays the content that will be references. If you select multiple elements in the element table, the content from all the selected elements is displayed.

Source Pane

Displays the source code of the element to be referenced.

Reference details Section

Reference type

The type of reference that will be inserted. If you selected a key that references a DITA resource, you will notice that **conkeyref** value is automatically selected.

Reference to

Oxygen XML Editor automatically fills this text field with the value of the `conkeyref` attribute to be inserted. However, you can edit this value if need be.

Fallback to (conref)

You can select this option to define a `conref` attribute to be used as a fallback to determine the content reference relationship if the specified `conkeyref` cannot be resolved.

Reference to range end (conrefend)

If you select multiple elements (of the same type) in the element table, Oxygen XML Editor automatically fills this text field with the `id` value of the last element in your selection. This value will be inserted as a `conrefend` attribute, defining the end of the `conkeyref` range.

Key Reference to Metadata (`keyref`) Options Using the Reuse Content Dialog Box

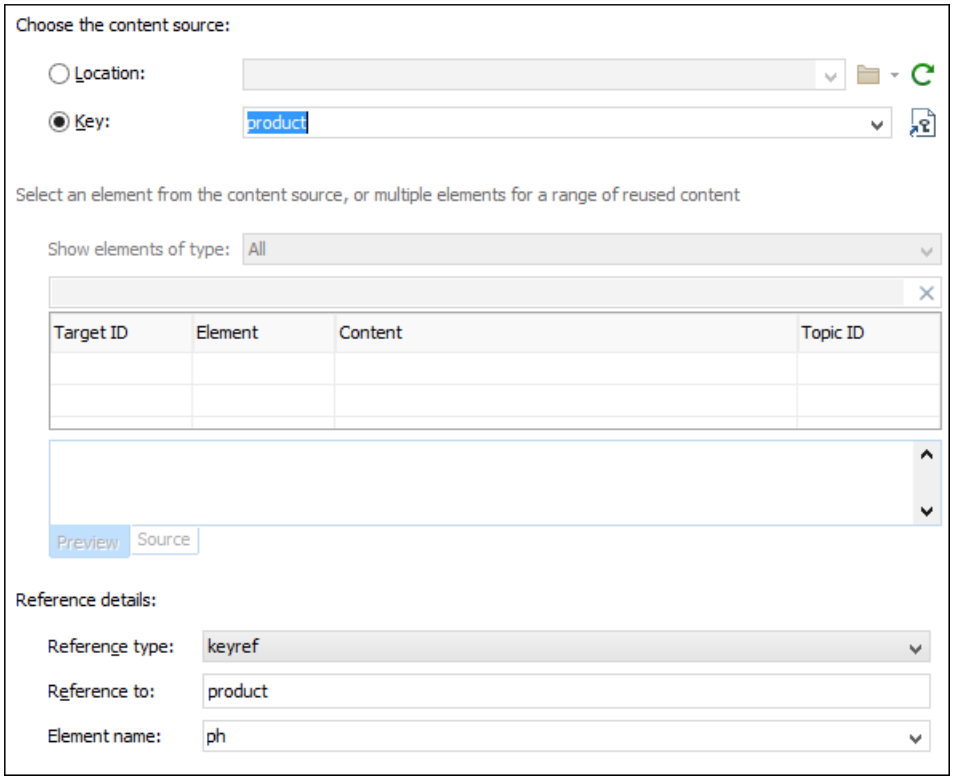


Figure 732: Insert Key Reference Options

Choose the content source Section

When **Key** is selected for the content source, you can use keys to reference content. You can use the **Choose Key Reference** button to open the **Choose Key** dialog box that allows you to select one from a list of all the keys that are gathered from the *root map* (you can also select one from the drop-down list in the **Key** field).

Note: If the current *DITA map* is not selected as the *root map*, no keys will be listed.

Tip: You can also use the ***DITA Reusable Components*** *view* for similar purposes.

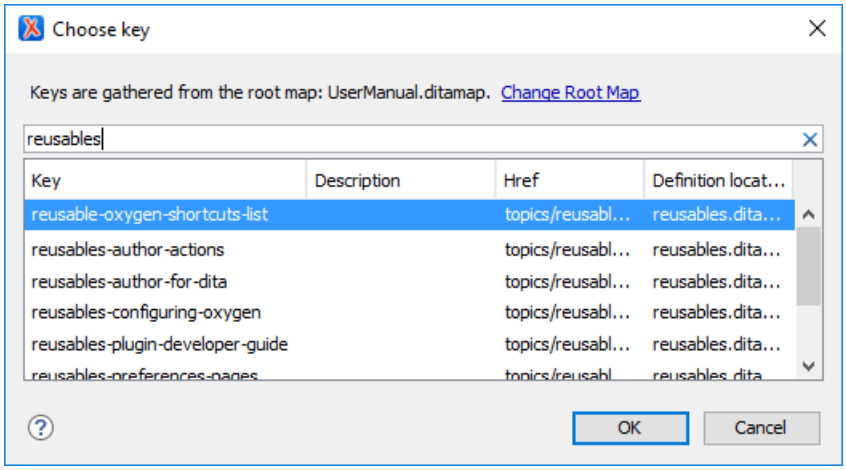


Figure 733: Choose Key Dialog Box

The **Choose Key** dialog box includes the following:

- **Change Root Map** - Opens a small dialog box that allows you to *select a root map*.
- **Search Filter** - You can enter text in the filter field at the top of the dialog box to filter the list and search for a specific keys.
- **Sortable Columns** - The dialog box includes the following columns that can be sorted by clicking on the heading:
  - **Key** - The name of the key.
  - **Description** - The description of the key that is obtained from the definition of the key (for example, the description for a *keydef* would be found in the `keyword` element inside *topicmeta*, or the description for a *topicref* would be found in a `navitle` element). For more information about where the data in this column is collected, see *this note*.
  - **Href** - The value of the *href* where the key points to.
  - **Definition Location** - The name of the *DITA map* where the key is defined.
- **Group by Definition Location** - A contextual menu action that can be used to group (and sort) all the keys based upon the value in the **Definition Location** column.

To insert a *key reference* to metadata (`keyref`), select the key you want to reference. Keys that do not have a value in the **Href** column are for referencing metadata with a `keyref` attribute. Therefore, to insert a `keyref`, you need to select a key that does not have a value (file path) in the **Href** column.

After you select a key, click **OK** to return to the **Reuse Content** dialog box.

When a key that references metadata has been selected, the **Reuse Content** dialog box offers the following additional options for inserting a `keyref`:

Select an element from the content source Section

This section is not used when referencing metadata.

Reference details Section

Reference type

The type of reference that will be inserted. If you selected a key that does not reference a DITA resource, you will notice that **keyref** value is automatically selected.

Reference to

Oxygen XML Editor automatically fills this text field with the value of the `keyref` attribute to be inserted.

Element name

Oxygen XML Editor automatically selects the element that is most commonly used for the selected type of key reference, but you can use the drop-down list to choose another element to use for the reference.

Finalizing Your Content Reference Configuration

Once you click **Insert** or **Insert and close**, the configured content reference is inserted into your document.

Tip: You can easily insert multiple content references by keeping the **Reuse Content** dialog box opened, using the **Insert** button.

Working with the Conref Push Mechanism

Content Reference Push Mechanism

The usual method of using content references pulls element content from a source element and inserts it in the current topic. DITA 1.2 introduced an alternative method of content referencing, allowing element content to be pushed, or injected, from a source topic to another topic without any special coding in the topic where the content will be re-used. This technique is known as a content reference *push* mechanism (*conref push*).

The *conref push* mechanism requires elements in the target topic (the topic where the content is to be pushed) to have ID elements, as the push mechanism inserts elements *before* or *after* a named element, or *replaces* the named element. Assuming the source topic is included in the *DITA map*, the *conref push* will be processed during publishing stage for the *DITA map*.

Example of a Conref Push Scenario

An example of a scenario in which a *conref push* would be useful is where a car manufacturer produces driver manuals that are distributed to various regions with their own specific regulations and certain sections need to be customized by the local car dealers before publishing. The local dealer could use a *conref push* technique to insert specific content without modifying the manufacturer-supplied content.

Push Current Element Action

Oxygen XML Editor includes an action that allows you to easily reference content with a *conref push* mechanism. The **Push Current Element** action is available in the **DITA** menu and in the **Reuse** subfolder of the contextual menu when editing in **Author** mode. Selecting this action opens the **Push current element** dialog box that allows you to select a target resource and element, and where to insert the current element content.

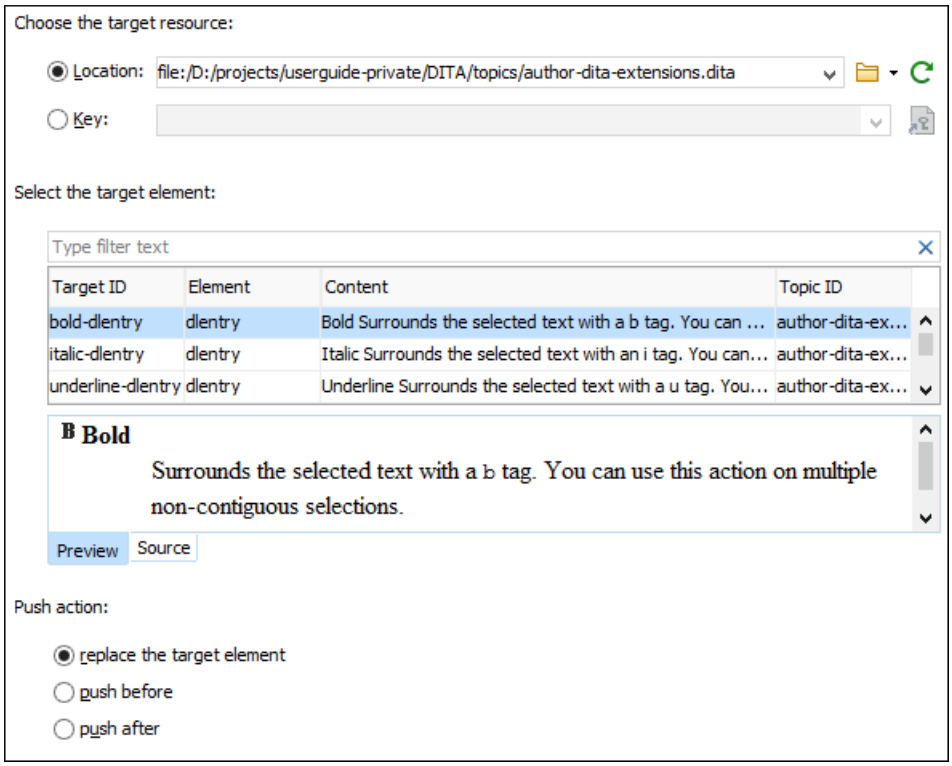


Figure 734: Push Current Element Dialog Box

This dialog box allows you to configure the following options for the *conref push* action:

Choose the target resource

Allows you to select a **Location** URL or a **Key** for the target resource and the table in the next section of the dialog box will be populated using the information from the specified resource.

Select the target element

The table in this section contains the available elements (identified by their ID) that belong to the same class as the current element on which the action was invoked.

Push action

Allows you to choose one of the following options for where you want to insert the current element content:

replace the target element

The target element will be replaced with the current element content.

On the technical side, the value of the `conaction` attribute in the current element will be set to `pushreplace` and the `conref` or `conkeyref` attribute will be set to the specified reference.

push before

The current element content will be inserted before the specified target element in the target resource.

On the technical side, the value of the `conaction` attribute in the current element will be set to `pushbefore`. Another element with the same name and class as the target element will be inserted in the document after the current element. The new element will have the `conaction` attribute set to `mark` and the `conref` or `conkeyref` attribute will be set to the specified reference.

push after

The current element content will be inserted after the specified target element in the target resource.

On the technical side, the value of the `conaction` attribute in the current element will be set to `pushafter`. Another element with the same name and class as the target element will be inserted in the document before the current element. The new element will have the `conaction` attribute set to `mark` and the `conref` or `conkeyref` attribute will be set to the specified reference.

You can also use the **Preview** panel to view the content that will be pushed and the **Source** panel to see the XML code for the content to be pushed. After you click **OK**, the *conref push* mechanism is inserted in the current document. The changes in the target resource will be processed when you transform the *DITA map*.

Related Information:

For more technical details about the conref push mechanism, refer to "The Conref Push Technique" section of The DITA Style Guide Best Practices for Authors.

Working with Reusable Components

In DITA, the content of almost any element can be made reusable simply by adding an `id` attribute to the element. The DITA content reference mechanism can reuse any element with an *ID*. However, it is not considered best practice to arbitrarily reuse pieces of text from random topics due to the difficulties this creates in trying to manage it. It also creates the possibility of authors deleting or changing content that is reused in other topics without being aware that the content is reused.

To prevent these types of problems, you can create reusable components to manage a separate set of topics that contain topics designed specifically for reuse. Then, all of your reusable content can be referenced from the reusable components and if the content needs to be updated you only need to edit it in one place.

Oxygen XML Editor allows you to select content in a topic, create a reusable component from it and reference that component in other locations by using the **Create Reusable Component** and **Insert Reusable Component** actions.

Related Information:

DITA Reusable Components View on page 1671

Creating a Reusable Content Component

Oxygen XML Editor makes it easy to create reusable content components from existing topic content.

Note: To ensure that the topic file that contains the reusable component is a valid container for just the reusable content component, without having to include the other elements required by a standard topic type, Oxygen XML Editor creates a specialized topic type on the fly. This specialization is designed to make sure that the content is compatible with the topic type from which it is created.

Follow these steps to create a reusable component:

1. In **Author** mode, select the content you want to make into a reusable component.
2. Select the **Create Reusable Component** action that is available in the **DITA** menu or the **Reuse** submenu of the contextual menu.
   The **Create Reusable Component** dialog box is displayed.
3. Use the **Reuse Content** drop-down list to select the scope of the content to be made reusable. It allows you to select how much of the current content you want to make reusable. The choices presented include the element at the current cursor position and its ancestor elements.
4. Add a description. This becomes the title of the topic that contains the reusable component, but is not part of the reusable content. It is just to help you identify the reusable content and will not become part of your output.
5. If you want to replace the extracted content with a reference to the new component you should leave the **Replace selection with content reference** option selected. This is recommended, since the point of reuse is to maintain only one copy of the content.

6. Select a file name and location to save the topic containing the reusable component and click **Save**. It is considered best practice to save or store reusable components in an area set aside for that purpose.
   If the **Replace selection with content reference** option was selected, Oxygen XML Editor replaces the current content with a `conref` attribute. The content of the content reference will be displayed in your current topic with a gray background, but it will no longer be editable since it is stored in a separate file. To edit the source of the reusable component, click the **Edit Content** icon at the beginning of the inserted content.

You now have a reusable component that you can include in other topics by using the **Insert Reusable Component** action that is available in the **DITA** menu or the **Reuse** submenu of the contextual menu. You can also reference this new reusable component by using a *content reference* or *content key reference*.

Inserting a Reusable Content Component

Oxygen XML Editor includes an **Insert Reusable Content** action that allows you to easily insert a reusable content component that you *created using the **Create Reusable Component** action*.

CAUTION: This action is only designed to insert reusable components created using the Oxygen XML Editor **Create Reusable Component** action. It assumes certain things about the structure of the reusable content file that may not be true of reusable content created by other methods and it may not provide the expected results if used with content that does not have the same structure.

The **Insert Reusable Content** action creates a DITA `conref` to insert the content, and creates a parent element for the `conref` attribute based on the type of the reusable element in the reusable component file. This action ensures that the correct element is used to create the `conref`. However, that element must still be inserted at a point in the current topic where that element type is permitted.

To insert a reusable component that was created using the **Create Reusable Component** action, follow these steps:

1. Place the cursor at the insertion point where you want the reusable component to be inserted.
2. Select the **Insert Reusable Component** action that is available in the **DITA** menu or the **Reuse** submenu of the contextual menu.
   The **Insert Reusable Component** dialog box is displayed.
3. Locate the reusable content file in which you want to insert its content.
4. If you select **Content reference** in the **Insert as** drop-down list, the action will add a `conref` attribute to the DITA element at the current location. If you select **Copy** in the drop-down list, the content of the reusable component file will simply be pasted at the current location (assuming the content is valid at the current location).
5. Click **Insert** to perform the action.

Working with Variable Text in DITA

You may often find that you want a certain piece of text in a topic to have a different value in various circumstances. For example, if you are reusing a topic about a feature that is shared between several products, you might want to make the name of the product variable so that the correct product name is used in the manual for each product.

For example, you might have a sentence like this:

```
The quick-heat feature allows [product-name] to come up to temperature quickly.
```

You need a way to substitute the correct product name for each product.

One way to do this would be to use conditional profiling, as in the following example:

```
<p>The quick-heat feature allows
    <ph product="basic">Basic Widget</ph>
    <ph product="pro">Pro Widget</ph>
to come up to temperature quickly.</p>
```

Creating Variable Text Using Keys

In DITA, you can create variable text using *keys*.

One way to do this would be to provide conditional values using the `product` profiling attribute.

However, this approach means that you are repeating the product names over and over again everywhere the product name is mentioned. This is time consuming for authors and will create a maintenance problem if the product names change.

Creating Variable Content Using a Key Reference

The alternative is to use a key reference, as in the following example:

```
<p>The quick-heat feature allows <ph keyref="product"/>
   to come up to temperature quickly.</p>
```

The definition of the key reference determines the name of the product:

```
<keydef keys="product" product="basic">
    <topicmeta>
      <keywords>
        <keyword>Basic Widget</keyword>
      </keywords>
    </topicmeta>
  </keydef>
  <keydef keys="product" product="pro">
    <topicmeta>
      <keywords>
        <keyword>Pro Widget</keyword>
      </keywords>
    </topicmeta>
  </keydef>
```

When the content is published, the current value of the key `product` will be inserted.

Inserting a Keyref

To insert a key reference into a document in Oxygen XML Editor **Author** mode, use one of the following methods:

- **DITA Reusable Components View Method**

  Use the ***DITA Reusable Components*** *view* to insert a *variable reference* to the *defined key*.
- **Code Template Method**

  Add the source code pattern of the *defined key* to a *code template* so that it appears in the list of proposals in the *Content Completion Assistant*.
- **Edit Attributes Method**

  Manually insert the `keyref` attribute using the attributes editor as follows:

  1. Press **Enter** and select any DITA element that supports the `keyref` attribute.
  2. Select **Edit Attributes** from the contextual menu (or simply press **Alt+Enter**) to bring up the *attributes editor*.
  3. In the **Name** field, select `keyref`.
  4. In the **Value** field, select or enter the name of the *defined key*.
- **Text Mode Method**

  If you are using ***Text*** *mode*, when you insert a `keyref` attribute, after you enter the first quote (`keyref="`), the **Content Completion Assistant** will list all the defined keys that you can select from. Note that this only works for local files.

Related Information:

DITA Reusable Components View on page 1671
Defining Keys in DITA Maps on page 1602

Working with DITA 1.3 Key Scopes

DITA 1.3 includes the possibility of using a concept called *Key Scopes* (or scoped keys). It allows you to reuse a topic in multiple places within the same *DITA map*, but with slightly different content in each instance.

Key Scopes Use-Case

Suppose that you develop a software product and you have a topic in your user guide that explains how to install your product on a Windows operating system. Suppose that the steps are exactly the same for installing it on Linux and the only difference is the name of the operating system. Therefore, it would be helpful if you could reuse the exact same content in two different topics, but with the name of the operating system different in each instance. In DITA 1.2, this is not possible since keys can only be resolved to a single value. However, with the DITA 1.3 *Key Scopes* mechanism, you can define multiple values for the same key depending on the context.

How to Use Key Scopes in Oxygen XML Editor

To use DITA 1.3 key scopes in Oxygen XML Editor, follow these steps:

1. *Define the keys* to be used in multiple places within your *DITA map*.
2. For each particular topic that contains the keys, define the key scopes:
   a. Right-click the topic in the ***DITA Maps Manager*** and select **Edit properties**.
   b. In the ***Keys*** *tab*, enter a value (or multiple values) in the **Key scopes** field.

      Tip: If you do not see the **Key scopes** field, make sure the support for DITA 1.3 (DITA-OT 2.x) is enabled in the *DITA preferences page*.
   c. Click **OK** to save your changes.
3. Save the *DITA map*.

   Result: In the ***DITA Maps Manager***, you can now see the key scopes in brackets and when you open each topic reference.

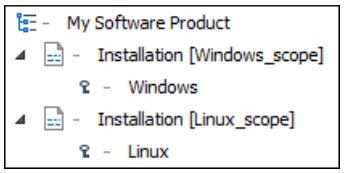


Figure 735: Key Scopes in DITA Maps Manager

The content will also be expanded in **Author** mode according to the context of the key scope you defined for that particular topic. Also, when you transform the *DITA map*, the scoped keys will be reflected in the published content.

Resources

- You can find a more detailed example and download samples for reuse possibilities based on key scopes in the *DITA 1.3 Key Scopes - Next Generation of Reuse* blog post.
- You can also watch our *DITA 1.3 video tutorial* to see how key scopes can be used in Oxygen XML Editor.

Related Information:

Working with DITA 1.3 Branch Filtering on page 1670

Oxygen XML Blog: DITA 1.3 Key Scopes - Next Generation of Reuse

Oxygen Video Tutorial: DITA 1.3 (Key Scopes, Branch Filtering)

Working with DITA 1.3 Branch Filtering

DITA 1.3 allows you to use a mechanism called *Branch Filtering* that enables you to set filtering conditions for specific branches of a *DITA map*. This makes it possible for multiple conditional profiles to be applied within a single publication, each time with a different filter.

Branch Filtering Use-Case

Suppose that you sell two models of a mobile phone and you need to create a brochure for each model. You want both brochures to have the same structure and most of the content is the same for both brochures. The only differences are in the values for certain details (for example, the model name, size dimensions, battery life, etc.) Therefore, it would be helpful if you could use the same topic and reference it twice in the same map, with

each reference using different filtering conditions. In DITA 1.2, this is not possible since you can only apply one DITAVAL filter to a map. However, with the DITA 1.3 *Branch Filtering* mechanism, you can reuse content multiple times within the same map, each time using different filters.

How to Use Branch Filtering in Oxygen XML Editor

To use DITA 1.3 branch filtering in Oxygen XML Editor, follow these steps:

1. The support for DITA 1.3 must be enabled in the *DITA preferences page*.
2. Assuming you have already *defined your profiling attributes*, *create a DITAVAL filter file*.
3. Insert a reference to the DITAVAL filter file in the *DITA map*:
   a. Right-click the *DITA map* reference in the ***DITA Maps Manager*** and select **Append Child** > **DITAVAL Reference**.

      Tip: If you do not see the **DITAVAL Reference** option, make sure the support for DITA 1.3 (DITA-OT 2.x) is enabled in the *DITA preferences page*.
   b. Select the DITAVAL file.
   c. Click **Insert and Close**.
4. Save the *DITA map*.

   Result: You can now see the **ditaval** files referenced in the ***DITA Maps Manager*** and when you transform the *DITA map*, filtered content will be reflected in the published output.

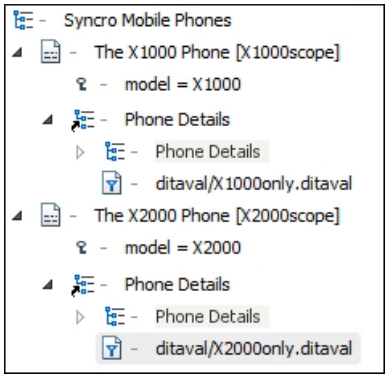


Figure 736: Branch Filtering in DITA Maps Manager

Resources

- You can find a more detailed example and download samples for reuse possibilities based on key scopes in the *DITA 1.3 Branch Filtering - Next Generation of Reuse* blog post.
- You can also watch our *DITA 1.3 video tutorial* to see how branch filtering can be used in Oxygen XML Editor.

Related Information:

Working with DITA 1.3 Key Scopes on page 1669

Oxygen XML Blog: DITA 1.3 Branch Filtering - Next Generation of Reuse

Oxygen Video Tutorial: DITA 1.3 (Key Scopes, Branch Filtering)

DITA Reusable Components View

The **DITA Reusable Components** view is helpful if you use a large amount of keys in your DITA project. It collects all of the keys that are defined in the *root map* and presents them in a dynamic table where you can easily locate and insert references to them as cross reference links or variables. It includes a search filter field to help you find particular keys, it has several sorting options, and some contextual menu actions. It also supports drag and drop actions and double-clicking a key is the fastest way to insert a key reference.

If the view is not displayed, it can be opened by selecting it from the **Window** > **Show View** menu. It will appear in the bottom-right section of the editor. It re-collects the keys anytime the *root map is changed* or you switch the editor focus to a different file.

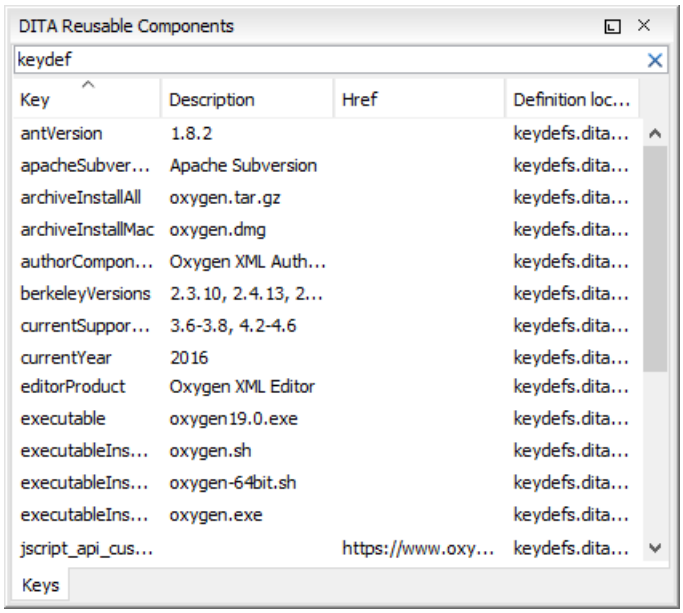


Figure 737: DITA Reusable Components View

The **DITA Reusable Components** view includes the following features and options:

Double-Click Mechanism

You can double-click any key listed in the view to insert a key reference at the current cursor position or surrounding the current selection. If the key points to an `href` value, it will be inserted as a *cross reference link (`xref`)*. If the key does not have an associated `href`, it will be inserted as a *variable reference (ph)*.

Drag and Drop Mechanism

You can drag a key from the view and drop it in the main editor to insert a key reference at the current cursor position. If the key points to an `href` value, it will be inserted as a *cross reference link (`xref`)*. If the key does not have an associated `href`, it will be inserted as a *variable reference (ph)*.

Search Filter

You can enter text in the filter field at the top of the view to filter the list and search for a specific keys.

Sortable Columns

The view includes the following columns that can be sorted by clicking on the heading:

- **Key** - The name of the key.
- **Description** - The description of the key that is obtained from the definition of the key (for example, the description for a *keydef* would be found in the `keyword` element inside *topicmeta*, or the description for a *topicref* would be found in a `navitle` element). For more information about where the data in this column is collected, see *this note*.
- **Href** - The value of the *href* where the key points to.
- **Definition Location** - The name of the *DITA map* where the key is defined.

Note: The **Description** column collects data from the definition of the key, either from the `navtitle` element or, if missing, from the `keyword` element. The following example shows two key definitions that will be collected in the keys table. Their corresponding information from the **Description** column will display `oxygen.sh` and `oxygen.tar.gz` respectively.

```
<keydef keys="executableInstallLinux">
  <topicmeta>
    <keywords>
      <keyword>oxygen.sh</keyword>
    </keywords>
  </topicmeta>
</keydef>

<keydef keys="archiveInstallAll">
  <topicmeta>
    <navtitle>
      oxygen.tar.gz
    </navtitle>
  </topicmeta>
```

```
</keydef>
```

Contextual Menu Actions

Insert as Link

Inserts a *cross reference link (`xref`)* to the selected key at the current cursor position or surrounding the current selection.

Insert as Variable

Inserts a *variable reference (ph)* to the selected key at the current cursor position or surrounding the current selection.

Insert as Keyref

Presents a submenu with all the elements that can be inserted at the current cursor position. Selecting an element will insert that element at the current cursor position or surrounding the current selection with a `keyref` attribute and its value set to the selected key.

Show Definition

Opens the *DITA map* where the key is defined.

Group by Definition Location

A toggle action that can be used to group (and sort) all the keys based upon the value in the **Definition Location** column.

Related Information:

Working with Reusable Components on page 1667
Linking in DITA Topics on page 1674
Working with Variable Text in DITA on page 1668
Working with Keys on page 1652

Linking in DITA

DITA provides support for various types of linking between topics, some of which is automated, while others are specified by the author. Oxygen XML Editor provides support for all forms of linking in DITA.

Linking Between Parent, Child, and Sibling Topics

A *DITA map* creates a hierarchical relationship between topics. That relationship map expresses a narrative flow from one topic to another, or it may be used as a classification system to help the reader find topics based on their classification, without creating a narrative flow. Since there may be various types of relationships between topics in a hierarchy, you may want to create links between topics in a variety of ways. For instance, if your topics are supposed to be organized into a narrative flow, you may want to have links to the next and previous topics in that flow. If your topics are part of a hierarchical classification, you may want links from parent to child topics, and vice versa, but not to the next and previous topics.

Parent, child, and sibling links are created automatically by the DITA output transformations (and may differ between various output formats). The kinds of links that are created are determined by the DITA *`collection-type` attribute*.

In-Line Linking in the Content of a Topic

DITA supports linking within the text of a topic using the `xref` element. The destination of the link can be expressed directly using the `href` attribute or indirectly using the `keyref` attribute. If you use the `keyref` attribute, you link to a key rather than directly to a topic. That key is then assigned to a topic in a map that includes that topic. This means that you can change the destination that a key points to by editing the key definition in the map or by substituting another map in the build.

Linking Between Related Topics

In addition to the relationships between topics that expressed by their place in the hierarchy of a map, a topic may be related to other topics in various ways. For instance, a task topic may be related to a concept topic that gives

the background of the task, or to a reference topic that provides data needed to complete the task. Task topics may also be related to other tasks in a related area, or concepts to related concepts.

Typically, they are grouped in a list at the end of the topic, although this depends on the behavior of the output transformation. DITA provides two mechanisms for expressing relationships between topics at the topic level: the **Related Links** section of a topic and relationship tables in maps.

Managing Links

Links can break for a variety of reasons. The topic that a link points to may be renamed or removed. A topic may be used in a map that does not include a linked topic. A topic or a key may not exist in a map when a particular profile is applied. The ***DITA Maps Manager*** provides a way to *validate all the links in the documents that are included in the map*. This can include validating all the profiling conditions that are applied.

Hierarchical Linking in DITA Maps

To create hierarchical linking between the topics in a *DITA map*, you set the appropriate value of the `collection-type` attribute on the map. See the *DITA documentation* for the meaning of each of the values of the `collection-type` attribute.

Note: Publishing scripts determine when and how to create hierarchical links. The `collection-type` attribute does not force a particular style of linking. Instead, it declares what the nature of the relationship is between the topics. The publishing scripts use that information to determine how to link topics. Scripts for different types of media might make the determination depending on what is appropriate for the particular type of media. You can provide additional instructions to the scripts using the `linking` attribute.

To add the `collection-type` to an item in a map:

1. Right-click the topic and choose **Edit Properties**. The **Edit Properties** dialog box is displayed.
2. In the **Attributes** tab, select the appropriate value from the **Collection type** drop-down list.
3. You can use the **Other attributes** table to add a value to the `linking` attribute.

Linking in DITA Topics

Direct Links

You can create inline links in the content of a DITA topic using the `xref` element. The destination of the link can be expressed directly by using the `href` attribute and the target can be another topic or a specific element within the other topic, another location within the same topic, a file, or a web link. You can also create direct *related links* to topics, files, or websites in a DITA topic using the `related-links` element.

Indirect Links Using Keys

The destination of the link can also be expressed indirectly by using *keys* to create either inline links or *related links* (with the `keyref` attribute). By using keys, you avoid creating a direct dependency between topics. This makes links easier to manage and can make it easier to reuse topics in various publications. It can also be helpful in verifying the completeness of a publication, by ensuring that a publication map provides a key definition for every key reference used in the content.

Links based on keys require two pieces:

- Key Definition - Assigns a key to a topic so that other topics can link to it. For more information, see *Defining Keys in DITA Maps* on page 1602.
- Key Reference - Created in an `xref` element and specifies the key to link to.

The key reference points to a key definition, and the key definition points to a topic. Key definitions are created in maps, as an element on the `topicref` element that points to a topic. This allows you to assign a particular key to one topic in one map and to another topic in another map. When a topic that links to that key is used in each of these maps, the links work correctly in both maps.

Inserting a Link in Oxygen XML Editor

To insert a link in ***Author*** *mode*, use the actions available in the **Link** drop-down menu from the toolbar (or the **Link** submenu in the contextual menu or **DITA** menu). You can choose between the following types of inline links:

Cross Reference

Opens the ***Cross Reference (xref)*** *dialog box* that allows you to insert a cross reference link to a target resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your DITA map structure. Once the target resource has been selected, you can also target specific elements within that resource. Depending on the context where it is invoked, the action inserts one of the following two elements:

- `xref` - Used to link to other topics or another location within the same topic and points to the target using the `href` or `keyref` attribute.
- `fragref` - A logical reference to a fragment element within a syntax diagram and points to the target using the `href` or `keyref` attribute.

File Reference

Opens a dialog box that allows you to insert a link to a target file resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your *DITA map* structure. It inserts an `xref` element with either an `href` attribute or a `keyref` attribute. If you select **Location** for the target, the link is expressed in an `href` attribute. If you select **Key** for the target, keys will be used to express the link in a `keyref` attribute. You can select a key from the drop-down list or click the **Choose Key Reference** button to use the ***Choose Key*** *dialog box*.

Web Link

Opens a dialog box that allows you to insert a link to a target web-related resource at the current location within a document. The target resource can be a URL or a key that is already defined in your *DITA map* structure. It inserts an `xref` element with either an `href` attribute or a `keyref` attribute. If you select **URL** for the target resource, the link is expressed in an `href` attribute. If you select **Key** for the target, keys will be used to express the link in a `keyref` attribute. You can select a key from the drop-down list or click the **Choose Key Reference** button to use the ***Choose Key*** *dialog box*.

Related Link to Topic

Opens the ***Cross Reference (xref)*** *dialog box* that allows you to insert a link to a target resource in a related links section that is typically at the bottom of your topic (although this depends on the behavior of the output transformation). The target resource can be the location of a file or a key that is already defined in your *DITA map* structure. Once the target resource has been selected, you can also target specific elements within that resource. If a related links section does not already exist, this action creates one. Specifically, it inserts a `link` element inside a *`related-links` element*.

Related Link to File

Opens a dialog box that allows you to insert a link to a target file resource in a related links section that is typically at the bottom of your topic (although this depends on the behavior of the output transformation). The target resource can be the location of a file or a key that is already defined in your *DITA map* structure. If a related links section does not already exist, this action creates one. Specifically, it inserts a `link` element inside a *`related-links` element*. If you select **Location** for the target, the link is expressed in an `href` attribute. If you select **Key** for the target, keys will be used to express the link in a `keyref` attribute. You can select a key from the drop-down list or click the **Choose Key Reference** button to use the ***Choose Key*** *dialog box*.

Related Link to Web Page

Opens the **Web Link** dialog box that allows you to insert a link to a target web-related resource in a related links section that is typically at the bottom of your topic (although this depends on the behavior of the output transformation). The target resource can be a URL or a key that is already defined in your *DITA map* structure. If a related links section does not already exist, this action creates one. Specifically, it inserts a `link` element inside a *`related-links` element*. If you select **URL** for the target resource, the link is expressed in an `href` attribute. If you select **Key** for the target, keys will be used to express the link in a `keyref` attribute. You can

select a key from the drop-down list or click the **Choose Key Reference** button to use the ***Choose Key*** *dialog box*.

Cross Reference (xref) Dialog Box

The **Cross Reference (xref)** dialog box is displayed when you insert a **Cross Reference** or **Related Link to Topic** (from the **Link** drop-down menu). It allows you to insert a link to a target resource at the current location within a document (for a **Cross Reference** link) or in a related links section (for a **Related Link to Topic**). The target resource can be the location of a file or a key that is already defined in your *DITA map* structure. Once the target resource has been selected, you can also target specific elements within that resource.

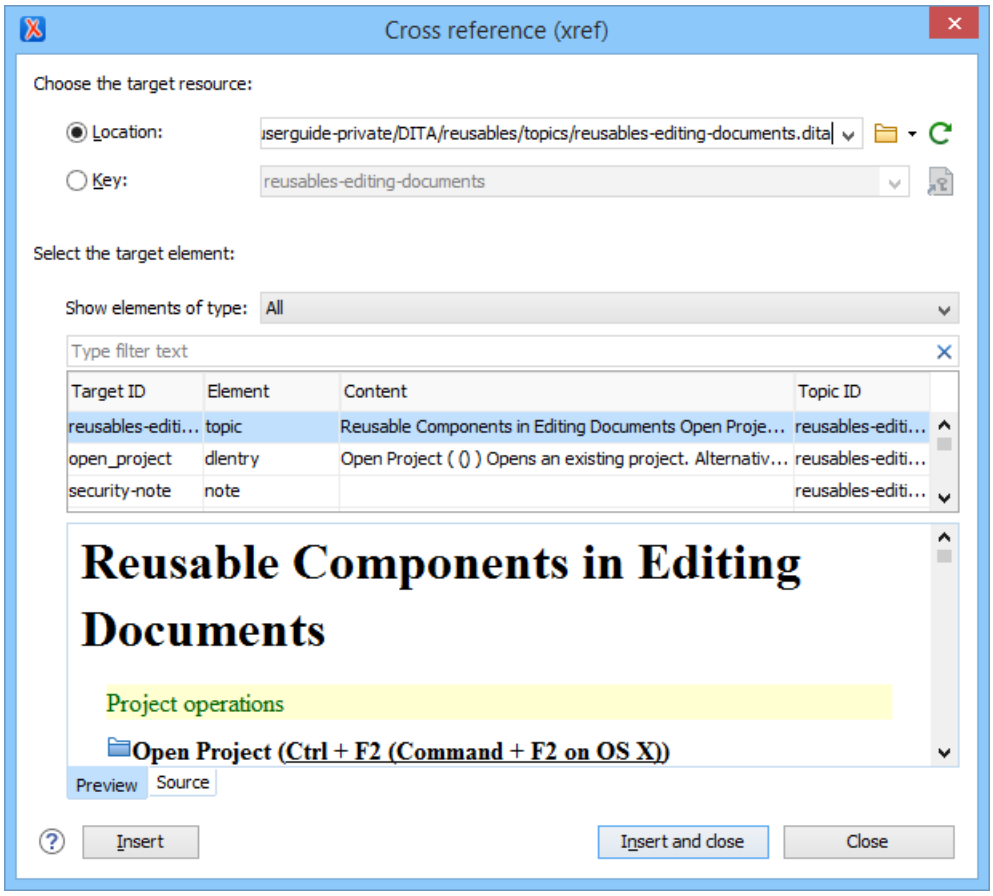


Figure 738: Cross Reference (xref) Dialog Box

This dialog box includes the following sections and fields:

Choose the Target Resource Section

Location

If you select **Location** for the target, the link is expressed in an `href` attribute.

Key

If you select **Key** for the target, keys will be used to express the link in a `keyref` attribute. You can use the **Choose Key Reference** button to open the **Choose Key** dialog box that allows you to select one from a list of all the keys that are gathered from the *root map* (you can also select one from the drop-down list in the **Key** field).

Tip: You can also use the ***DITA Reusable Components*** *view* for similar purposes.

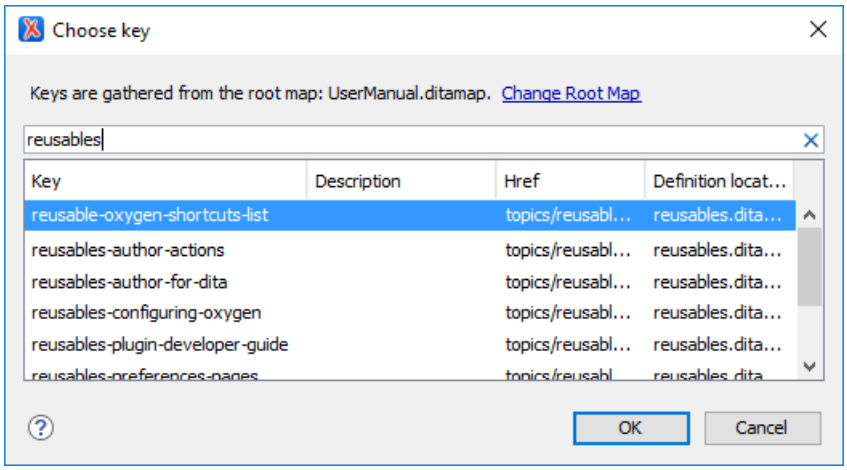


Figure 739: Choose Key Dialog Box

The **Choose Key** dialog box includes the following:

- **Change Root Map** - Opens a small dialog box that allows you to *select a root map*.
- **Search Filter** - You can enter text in the filter field at the top of the dialog box to filter the list and search for a specific keys.
- **Sortable Columns** - The dialog box includes the following columns that can be sorted by clicking on the heading:
  - **Key** - The name of the key.
  - **Description** - The description of the key that is obtained from the definition of the key (for example, the description for a *keydef* would be found in the `keyword` element inside *topicmeta*, or the description for a *topicref* would be found in a `navitle` element). For more information about where the data in this column is collected, see *this note*.
  - **Href** - The value of the *href* where the key points to.
  - **Definition Location** - The name of the *DITA map* where the key is defined.
- **Group by Definition Location** - A contextual menu action that can be used to group (and sort) all the keys based upon the value in the **Definition Location** column.

Select the Target Element Section

This section can be used to target a specific element inside the target resource.

Show elements of type

You can use this drop-down list to select specific types of elements to be displayed in the subsequent table. This can help you narrow down the list of possible source elements that you can select.

Text Filter Field

You can also use the text filter field to narrow down the list of possible source elements to be displayed in the subsequent table.

Element Table

Presents all the element IDs defined in the source topic. Use this table to select the **Target ID** of the element that you want to reference.

Preview Pane

Displays the content that will be references.

Source Pane

Displays the XML source code of the element to be referenced.

Once you click **Insert** or **Insert and close**, the configured cross reference is inserted into your document.

Tip: You can easily insert multiple cross references by keeping the dialog box opened, using the **Insert** button.

Using Copy/Paste Actions to Insert a Cross Reference

Oxygen XML Editor also includes support for inserting cross references with simple copy/paste actions. The copied content must be an entire DITA XML element with an ID attribute or a `topicref`. Also, the location in the document where you paste the link must be valid, although as long as the ***Smart paste and drag and drop*** *option* is selected in the **Schema Aware** preferences page, if you try to paste it in an invalid location, Oxygen XML Editor will attempt to place it in a valid location, and may prompt you with one or more choices for where to place it. You can paste the link as a cross reference with the link expressed in an `href` attribute or in a `keyref` attribute.

To insert a cross reference link (expressed in an `href` attribute) using copy/paste actions, follow these steps:

1. Copy an entire DITA element that has an ID attribute assigned to it or a `topicref`.
2. Place the cursor at a location, where you want to insert the link.
3. Select the **Paste as Link** action from the **Paste Special** submenu from the contextual menu.

To insert a cross reference link (expressed in a `keyref` attribute) using copy/paste actions, follow these steps:

1. In the ***DITA Maps Manager*** *view*, make sure that the ***Root map*** *combo box* points to the correct map that stores the keys.
2. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the ***DITA Maps Manager***, select **Edit Properties**, and enter a value in the **Keys** field.
3. Copy an entire DITA element that has an ID attribute assigned to it from a topic with an assigned key, or a `topicref` from a *DITA map*.
4. Place the cursor at a location, where you want to insert the link.
5. Select the **Paste as Link (keyref)** action from the **Paste Special** submenu from the contextual menu.

Related Information:

Defining Keys in DITA Maps on page 1602
DITA Reusable Components View on page 1671

Linking with Relationship Tables in DITA

A relationship table is used to express relationships between topics outside of the topics themselves. The DITA publishing scripts can then create links between related topics when the content is published.

The reason for using a relationship table is to help make topics easier to reuse. If a topic links directly to another topic, this creates a dependency between the topics. If one topic is reused in a publication where the other is not used, the link is broken. By defining relationships between topics in a relationship table, you avoid creating this dependency.

To create an appropriate set of links between topics in multiple publications, you can create a separate relationship table for each publication. If you are creating multiple publications by applying profiling conditions to a single map, you can also profile your relationship table.

To create a relationship table, follow these steps :

1. If the map is currently open in the ***DITA Maps Manager***, double-click the map icon ( ) to open the map in **Author** mode. If it opens in **Text** mode, click **Author** at the bottom left to switch to **Author** mode.
2. Move the insertion point inside the *map* root element (usually `map`, but it might be `bookmap`, or another specialization of the `map` element). The easiest way to do this is to click below the title of the map in the editor and then press the up arrow once. Confirm that you are inside the *map* root element by checking the breadcrumbs at the top left of the editor window. You should only see the name of the *map* root element.
3. Select the **Insert Relationship Table** action on the toolbar or from the **Relationship Table** submenu of the contextual menu.
   The **Insert Relationship Table** dialog box is displayed.
4. Set the number of rows, the number of columns, a table title (optional), and select whether or not you want a table header. Click **Insert**.
5. Enter the type of the topics in the header of each column.

The header of the table (the `relheader` element) already contains a `relcolspec` element for each table column. You should set the value of the attribute `type` of each `relcolspec` element to a value such as *concept*, *task*, or *reference*. When you click in the header cell of a column (that is a `relcolspec` element), you can see all the attributes of that `relcolspec` element, including the `type` attribute in the ***Attributes*** *view*. You can edit the attribute type in this view.

6. To insert a topic reference in a cell, place the cursor in a table cell and select ***Insert Reference*** from the contextual menu or the **DITA Map** toolbar.
7. To add a new row to the table or remove an existing row use **Insert Relationship Row**/**Delete Relationship Row** from the contextual menu or the **DITA Map** toolbar.
8. To add a new column to the table or remove an existing column, use **Insert Relationship Column**/**Delete Relationship Column** contextual menu or the **DITA Map** toolbar. If you double-click the relationship table (or select it and press **Enter**, or choose **Open** from the contextual menu) the *DITA map* is opened in the editor with the cursor positioned inside the corresponding relationship table.
9. To add topic references to your relationship table, drag and drop topics from the ***DITA Maps Manager*** or the ***Project*** view into the appropriate cell in the relationship table.

   See the *DITA documentation* for a full explanation of the relationship table format and its options. Note that you can change all the selections that you make here later by using the actions on the toolbar (or in the **Relationship Table** submenu of the contextual menu) or by editing the underlying XML in **Text** mode.
10. Save the *DITA map*.
    Relationship tables are also displayed in the ***DITA Maps Manager*** *view*, along with the other elements in its *DITA map*.

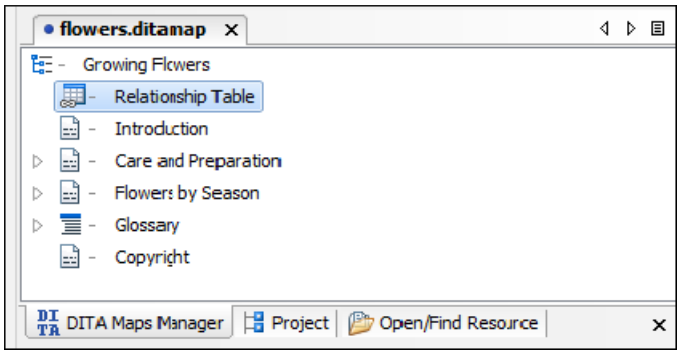


Figure 740: Relationship Table

You can open the DITA map to edit the relationship table by doing one of the following:

- Double-click the appropriate relationship table in the ***DITA Maps Manager***.
- Select the relationship table in the ***DITA Maps Manager*** and press **Enter**.
- Select **Open** from the contextual menu of the relationship table in the ***DITA Maps Manager***.

Publishing DITA Output

As a structured writing format, DITA produces structured content (content that is annotated with specific structural and semantic information rather than with formatting information). To create a publication, your *DITA map* and its associated topics must be processed by a transformation script. That script is responsible for how the structural and semantic information in the DITA files is converted into formatting information for display.

This means that you can display the same DITA content in multiple ways, types of media, or publications. It also means that you cannot control every aspect of the presentation of your content in your DITA files. The only way to change the formatting is to change the transformation routines that create it.

Therefore, to create output from your DITA content you have to run a transformation on your content. Transformations for various types of output are provided by the DITA Open Toolkit. Oxygen XML Editor provides a mechanism called transformation scenarios to help you configure and run transformations.

Note: Oxygen XML Editor does not create any output formats itself. Oxygen XML Editor runs externally defined transformations that produce output, and displays the result in the appropriate application, but the output itself is produced by the external transformation, not by Oxygen XML Editor.

Running a Transformation Scenario

To select and run a transformation scenario on your map, follow these steps:

1. Click the **Configure Transformation Scenario(s)** button on the ***DITA Maps Manager*** *toolbar*. The ***Configure Transformation Scenario(s)*** *dialog box* appears. This dialog box lists all the transformation scenarios that have been configured in your project. Oxygen XML Editor provides a default set of transformation scenarios, but the people in charge of your DITA system may have provided others that are specifically configured for your needs.
2. Select the transformation scenario you want to run and click **Apply Associated**. The transformation scenario runs in the background. You can continue to work in Oxygen XML Editor while the transformation is running. If there are errors or warnings, Oxygen XML Editor displays them when the transformation is complete. If the transformation is successful, Oxygen XML Editor opens the output in the appropriate application.
3. To rerun the same scenario again, click the **Apply Transformation Scenario(s)** button.

Transforming DITA Content

Oxygen XML Editor uses the DITA Open Toolkit (DITA-OT) to transform *DITA maps* and topics into an output format. For this purpose, both the DITA Open Toolkit and Ant come bundled in Oxygen XML Editor.

For more information about the DITA Open Toolkit, see *DITA Open Toolkit Documentation*.

This section includes information about how to create a DITA OT transformation and how to customize DITA transformations.

Related Information:

Transforming Documents on page 854

Creating or Editing a DITA OT Transformation

Creating a DITA OT Transformation Scenario

To create a **DITA OT Transformation** scenario, use one of the following methods:

- Use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **DITA OT Transformation**.
- Use the **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on OS X))** action from the toolbar or the **Document** > **Transformation** menu. Then click the **New** button and select **DITA OT Transformation**.

  Note: If a scenario is already associated with the edited document, selecting **Apply Transformation Scenario(s)** runs the associated scenario automatically. You can check to see if transformation scenarios are associated with the edited document by hovering your cursor over the **Apply Transformation Scenario** button.
- Go to **Window** > **Show View** and select **Transformation Scenarios** to display this view. Click the **New** button and select **DITA OT Transformation**.

All three methods open the **DITA Transformation Type** dialog box that presents the list of possible outputs.

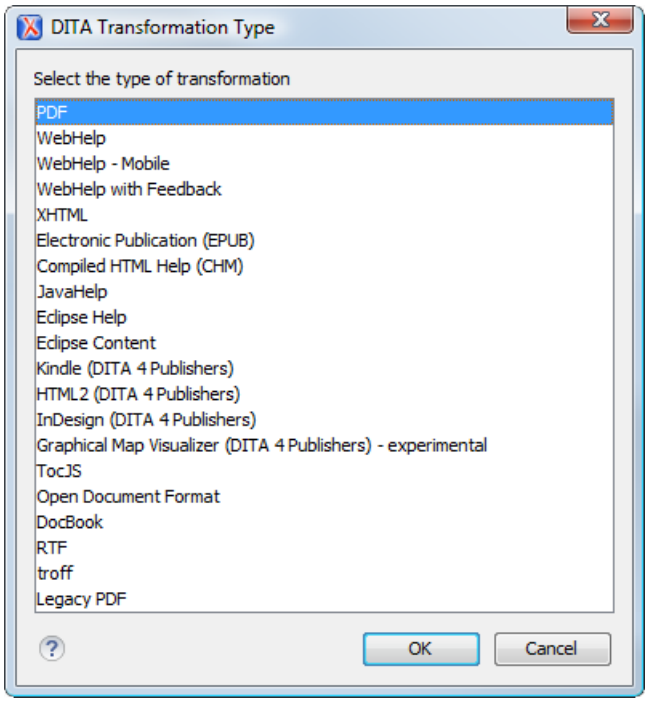


Figure 741: DITA Transformation Type Dialog Box

Select the desired type of output and click **OK**. This opens the **New Scenario** dialog box.

The upper part of the dialog box allows you to specify the **Name** of the transformation scenario and the following **Storage** options:

- ***Project Options*** - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- ***Global Options*** - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation. Some of these tabs are only available for certain output types (for example, a **Skins** tab is only available for **WebHelp Classic** and **WebHelp Classic with Feedback** output types, a **Templates** tab is available only for **WebHelp Responsive** and **WebHelp Responsive with Feedback**, and a **FO Processor** tab is available for PDF output).

Editing a DITA OT Transformation Scenario

Editing a transformation scenario is useful if you need to configure some of its parameters.

To configure an existing transformation scenario, follow these steps:

1. Select the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on OS X))** action from the toolbar or the **Document** > **Transformation** menu.

   Step Result: The ***Configure Transformation Scenario(s)*** *dialog box* is opened.
2. Select the particular transformation scenario and click the **Edit** button at the bottom of the dialog box or from the contextual menu.

   Note: Since transformation scenarios that are associated with predefined *frameworks* are read-only, these scenarios will prompt you to use the ***Duplicate*** *button and then edit the duplicated scenario*.

Result: This will open an **Edit scenario** configuration dialog box that contains several tabs that allow you to configure the options that control the transformation. Some of these tabs are only available for certain output types (for example, a **Skins** tab is only available for **WebHelp Classic** and **WebHelp Classic with Feedback** output

types, a **Templates** tab is available only for **WebHelp Responsive** and **WebHelp Responsive with Feedback**, and a **FO Processor** tab is available for PDF output).

Related Information:

Creating a DITA OT Customization Plugin on page 1712
Installing a Plugin in the DITA Open Toolkit on page 1714
DITA Open Toolkit Documentation

Skins Tab (DITA OT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Skins** tab is available for DITA OT transformations with **WebHelp Classic** or **WebHelp Classic with Feedback** output types and it provides a set of predefined skins that you can use as a base for your WebHelp system output.

A *skin* is a collection of CSS properties that can alter the look of the output by changing colors, font types, borders, margins, and paddings. This allows you to rapidly adapt the look and feel of your output.

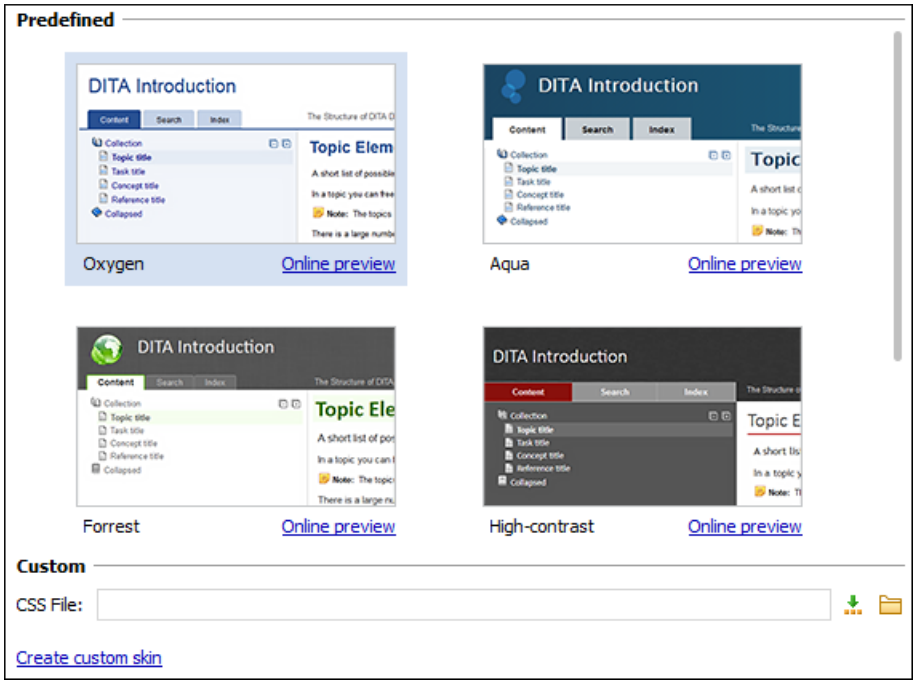


Figure 742: Skins Tab

The **Skins** tab includes the following sections:

Predefined Skins

This sections presents the predefined skins that are included in Oxygen XML Editor. The predefined skins cover a wide range of chromatic themes, ranging from a very light one to a high-contrast variant. To see how the *skin* looks when applied on a sample documentation project that is stored on the Oxygen XML Editor website, press the **Online preview** link.

Custom Skins

You can use this section to customize the look of the output.

CSS File

You can set this field to point to a custom CSS stylesheet or customized skin. A custom CSS file will overwrite a skin selection.

Note: The output can also be styled by setting the `args.css` parameter in the **Parameters tab**. The properties taken from the stylesheet referenced in this parameter take precedence over the properties declared in the skin set in the **Skins tab**.

Create custom skin

Use this link to open the ***WebHelp Skin Builder*** tool.

Templates Tab (DITA OT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Templates** tab is available for DITA OT transformations with **WebHelp Responsive** or **WebHelp Responsive with Feedback** output types and it provides a set of predefined *skins* that you can use as a base for the layout of your WebHelp system output.

A *skin* is a collection of CSS properties that can alter the look of the output by changing colors, font types, borders, margins, and paddings. This allows you to rapidly adapt the look and feel of your output. You can choose predefined skins in a *tile* style of layout or a *tree* style of layout, and you can also *add your own customized skins*.

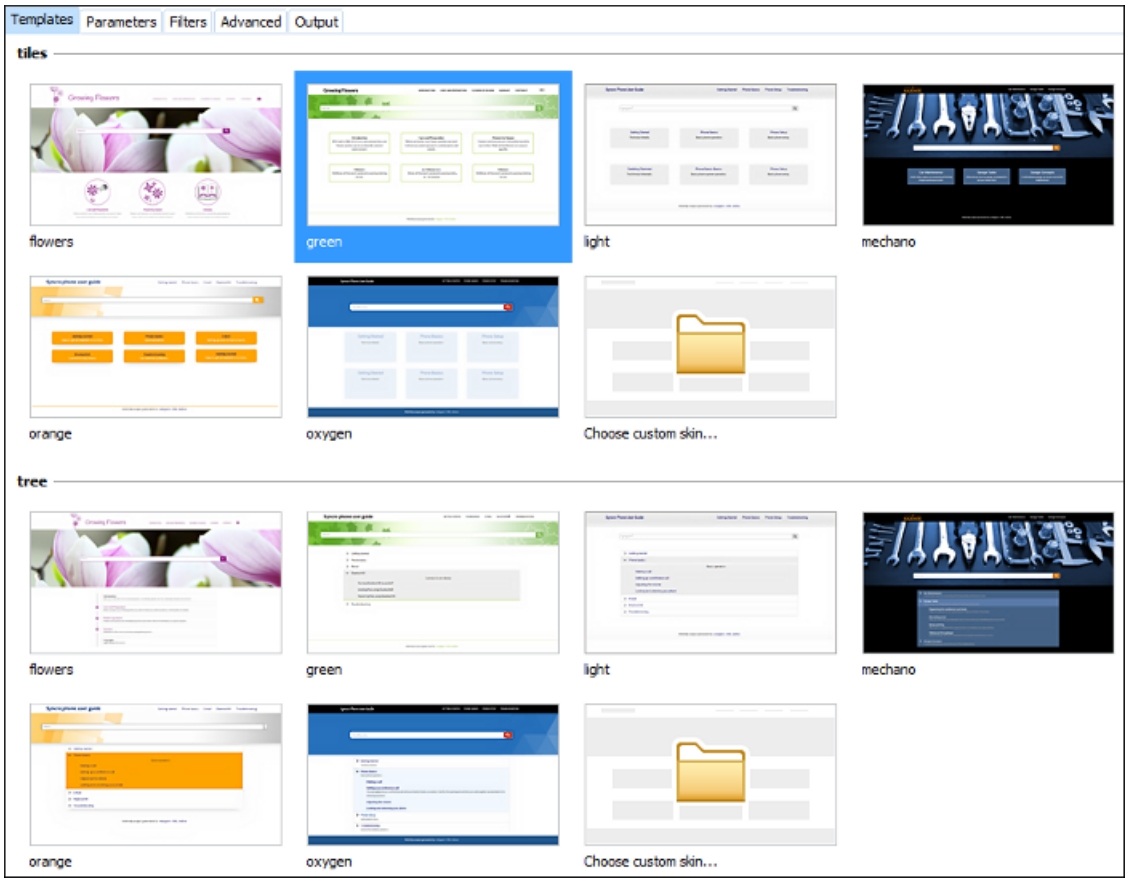


Figure 743: Templates Tab

The **Templates** tab comes by default with the following predefined collections of skins:

Tiles

This sections presents the predefined skins that are arranged in a *tiles* style of layout. These predefined skins include a variety of themes, ranging from a very light one to a high-contrast variant, and various styles. If you select **Choose custom skin**, you can select a custom CSS stylesheet to be used as your template.

Tree

This sections presents the predefined skins that are arranged in a *tree* style of layout. These predefined skins include a variety of themes, ranging from a very light one to a high-contrast variant, and various styles. If you select **Choose custom skin**, you can select a custom CSS stylesheet to be used as your template.

When you add a new collection of skins, this tab will list them.

FO Processor Tab (DITA OT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **FO Processor** tab is available for DITA OT transformations with a **PDF** output type.

This tab allows you to select an FO Processor to be used for the transformation.

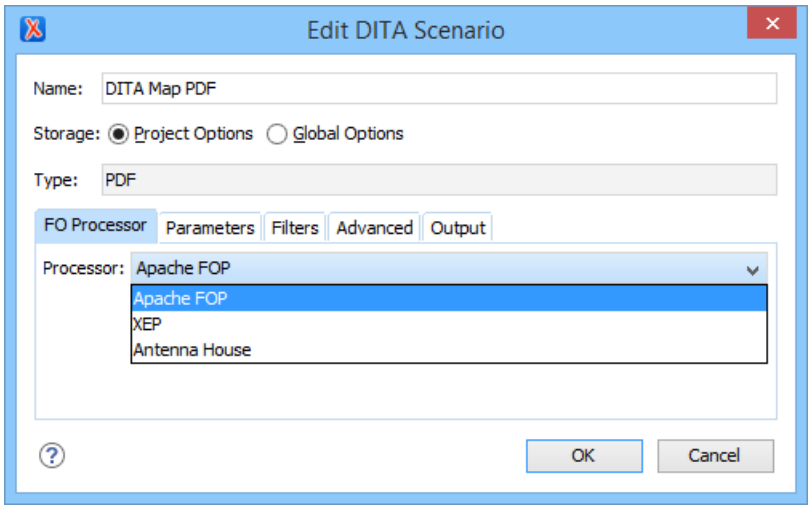


Figure 744: FO Processor Configuration Tab

You can choose one of the following processors:

Apache FOP

The default processor that comes bundled with Oxygen XML Editor.

XEP

The *RenderX* XEP processor. If XEP is already installed, Oxygen XML Editor displays the detected installation path under the drop-down menu. XEP is considered installed if it was detected in one of the following sources:

- XEP was configured as an external FO Processor in the ***FO Processors** option page*.
- The system property *com.oxygenxml.xep.location* was set to point to the XEP executable file for the platform (for example: `xep.bat` on Windows).
- XEP was installed in the *`DITA-OT-DIR`*`/plugins/org.dita.pdf2/lib` directory of the Oxygen XML Editor installation directory.

Antenna House

The *Antenna House* (AH Formatter) processor. If Antenna House is already installed, Oxygen XML Editordisplays the detected installation path under the drop-down menu. Antenna House is considered installed if it was detected in one of the following sources:

- Environment variable set by Antenna House installation (the newest installation version will be used).
- Antenna House was added as an external FO Processor in the Oxygen XML Editor preferences pages.

To further customize the PDF output obtained from the Antenna House processor, follow these steps:

1. **Edit** the transformation scenario.
2. Open the ***Parameters** tab*.
3. Add the `env.AXF_OPT` parameter and point to the Antenna House configuration file.

Related Information:

FO Processors Preferences on page 129
XSL-FO Processors on page 936

Parameters Tab (DITA OT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Parameters** tab allows you to configure the parameters sent to the DITA-OT build file.

The table in this tab displays all the parameters that the DITA-OT documentation specifies as available for each chosen type of transformation (for example, XHTML or PDF), along with their description and current values. You can find more information about each parameter in the *DITA OT Documentation*. You can also add, edit, and remove parameters, and you can use the text box to filter or search for a specific term in the entire parameters collection. Note that edited parameters are displayed with their name in bold.

Depending on the type of a parameter, its value can be one of the following:

- A simple text field for simple parameter values.
- A combo box with some predefined values.
- A file chooser and an *editor variable* selector to simplify setting a file path as the value of a parameter.

Note: To input parameter values at runtime, use the *ask editor variable* in the **Value** column.

Below the table, the following actions are available for managing parameters:

New

Opens the **Add Parameter** dialog box that allows you to add a new parameter to the list. You can specify the **Value** of the parameter by using the ***Insert Editor Variables*** button or the **Browse** button.

Unset

Resets the selected parameter to its default value. Available only for edited parameters with set values.

Edit

Opens the **Edit Parameter** dialog box that allows you to change the value of the selected parameter or its description.

Delete

Removes the selected parameter from the list. It is available only for new parameters that have been added to the list.

Related Information:

DITA Open Toolkit Documentation

Filters Tab (DITA Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Filters** tab allows you to add filters to remove certain content elements from the generated output.

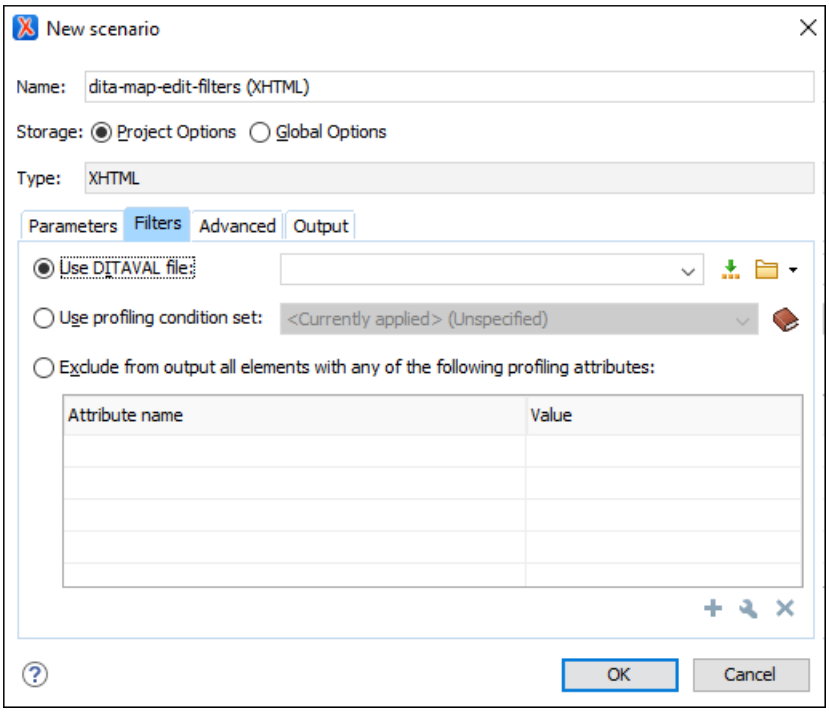


Figure 745: Edit Filters Tab

You can choose one of the following options to define filters:

Use DITAVAL file

If you already have a *DITAVAL* file associated with the *DITA map*, you can specify the file to be used when filtering content. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list. You can find out more about constructing a *DITAVAL* file in the *DITA Documentation*.

Attention: If a filter file is specified in the `args.filter` parameter (in *the **Parameters** tab*), that file takes precedence over a *DITAVAL* file specified here.

Use profiling condition set

Sets the *profiling condition set* that will be applied to your transformation.

Exclude from output all elements with any of the following attributes

By using the **New**, **Edit**, or **Delete** buttons at the bottom of the pane, you can configure a list of attributes (name and value) to exclude all elements that contain any of these attributes from the output.

Advanced Tab (DITA OT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Advanced** tab allows you to specify advanced options for the transformation scenario.

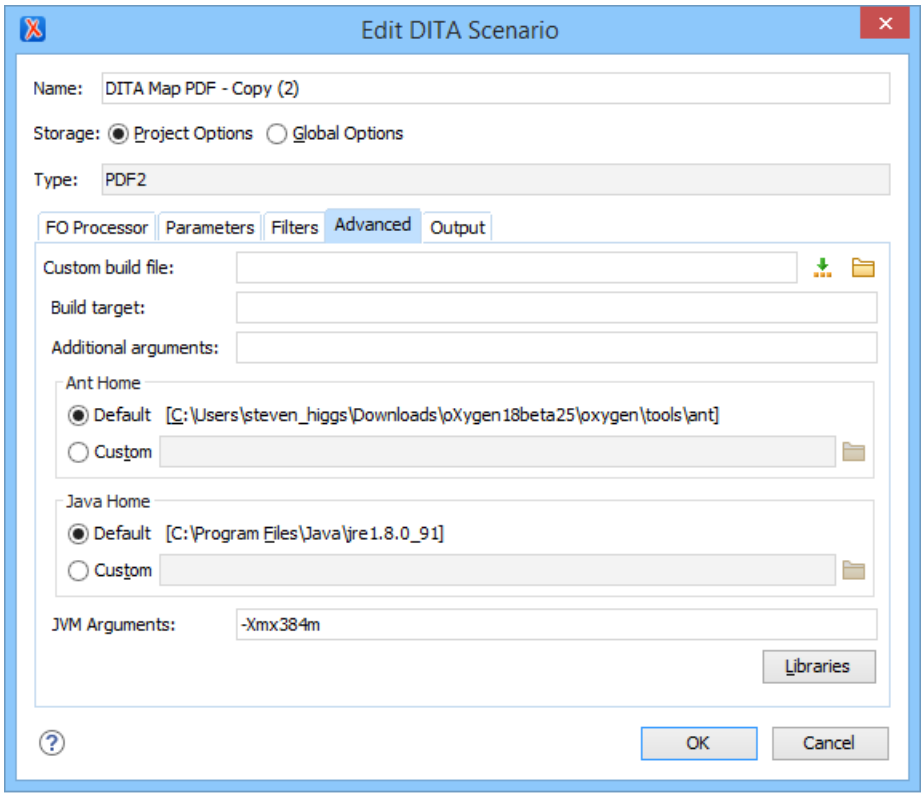


Figure 746: Advanced Settings Tab

You can specify the following parameters:

Custom build file

If you use a custom DITA-OT build file, you can specify the path to the customized build file. If empty, the `build.xml` file from the *dita.dir* parameter that is configured in the ***Parameters*** *tab* is used. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Build target

Optionally, you can specify a build target for the build file. If no target is specified, the default `init` target is used.

Additional arguments

You can specify additional command line arguments to be passed to the transformation (such as -verbose).

Ant Home

You can choose between the default or custom Ant installation to run the transformation. The default path can be configured in the *Ant preferences page*.

Java Home

You can choose between the default or custom Java installation to run the transformation. The default path is the Java installation that is used by Oxygen XML Editor.

Note: It may be possible that the used Java version is incompatible with the DITA Open Toolkit engine. For example, DITA OT 1.8 and older requires Java 1.6 or later, while DITA OT 2.0 and newer requires Java 1.7 or newer. Thus, if you encounter related errors running the transformation, consider installing a Java VM that is supported by the DITA OT publishing engine and using it in the **Java Home** text field.

JVM Arguments

This parameter allows you to set specific parameters for the Java Virtual Machine used by Ant. For example, if it is set to -Xmx384m, the transformation process is allowed to use 384 megabytes of memory. When performing a large transformation, you may want to increase the memory allocated to the Java Virtual Machine. This will help avoid *Out of Memory* error messages (**OutOfMemoryError**).

Libraries

By default, Oxygen XML Editor adds libraries (as high priority) that are not transformation-dependent and also patches for certain DITA Open Toolkit bugs. You can use this button to specify additional libraries (*JAR* files or additional class paths) to be used by the Ant transformer.

Tip: You can specify the path to the additional libraries using wildcards (for example, `${oxygenHome}/lib/*.jar`).

Output Tab (DITA OT Transformations)

When you *create a new transformation scenario* or *edit an existing one*, a configuration dialog box allows you to customize the transformation with various options in several tabs.

The **Output** tab allows you to configure options that are related to the location where the output is generated.

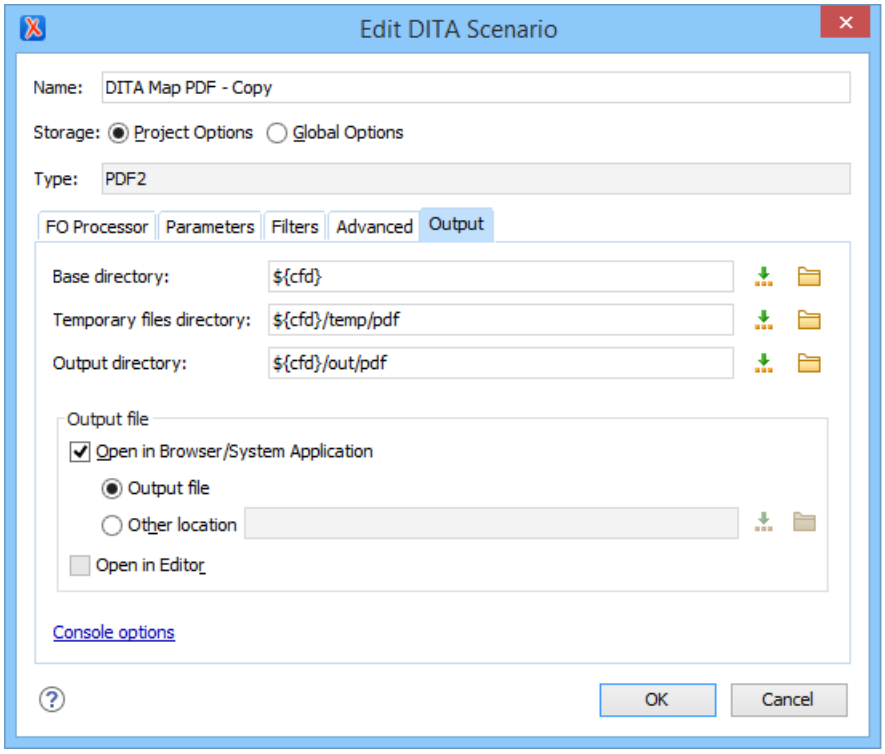


Figure 747: Output Settings Tab

You can specify the following parameters:

Base directory

All the relative paths that appear as values in parameters are considered relative to the base directory. The default value is the directory where the transformed map is located. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Temporary files directory

This directory is used to store pre-processed temporary files until the final output is obtained. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Output directory

The folder where the content of the final output is stored. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Note: If the *DITA map* or topic is opened from a remote location or a ZIP file, the parameters must specify absolute paths.

Open in Browser/System Application

If selected, Oxygen XML Editor automatically opens the result of the transformation in a system application associated with the file type of the result (for example, in Windows *PDF* files are often opened in *Acrobat Reader*).

Note: To set the web browser that is used for displaying HTML/XHTML pages, go to **Options** > **Preferences** > **Global**, and set it in the **Default Internet browser** field.

- **Output file** - When **Open in Browser/System Application** is selected, you can use this button to automatically open the default output file at the end of the transformation.
- **Other location** - When **Open in Browser/System Application** is selected, you can use this option to open the file specified in this field at the end of the transformation. You can specify the path by using the text field, the ***Insert Editor Variables*** button, or the **Browse** button.

Open in editor

When this is option is selected, at the end of the transformation, the default output file is opened in a new editor panel with the appropriate built-in editor type (for example, if the result is an XML file it is opened in the built-in XML editor, or if it is an XSL-FO file it is opened with the built-in FO editor).

At the bottom of the pane there is a link to the *Console options* preferences page that contains options to control the display of the console output received from the publishing engine.

Customizing DITA Transformations

Oxygen XML Editor includes a bundled copy of the DITA-OT as an Oxygen XML Editor *framework*. That *framework* includes a number of transformation scenarios for common output formats. This section includes topics about customizing DITA transformations, such as using a custom build file, customizing PDF output, and using DITA OT *plugins* to customize your needs.

Customizing Output Transformations

You can customize the appearance of any of the output types by customizing the output transformations. There are several ways to do this:

- Most transformations are configurable by passing parameters to the transformation script. Oxygen XML Editor allows you to *set parameters* on a transformation scenario and you can *save and share them with others*. You can also use the *${ask} editor variable* in the **Parameters** tab to instruct Oxygen XML Editor to prompt you for a particular parameter whenever a transformation scenario is run. You can set up multiple transformation scenarios for a given output type, allowing you to maintain several customized transformation scenarios for multiple types of output configurations.
- If you want to customize an output in a way not supported by the customization options, you can create a modified version of the transformation code. Some transformation scripts export specific forms of extension or customization. You should consult the DITA Open Toolkit for the transformation type that you are interested in to see what customization options it supports. Oxygen XML Editor provides full editing and *debugging support from XSLT and CSS stylesheets*, which you can use to modify transformation code.

You can also write your own output transformation scripts to produce a type of output not supported by the DITA Open Toolkit. Oxygen XML Editor provides a full source editing environment for developing such transformations. You can create Oxygen XML Editor transformation scenarios to run these scripts once they are complete.

Related Information:

Transforming DITA Content on page 1680

DITA Open Toolkit Documentation

Using a Custom Build File

To use a custom build file in a DITA-OT transformation, follow these steps:

1. Use the **Configure Transformation Scenario(s)** action to open the ***Configure Transformation Scenario(s)*** *dialog box*.
2. Select the transformation scenario and click **Edit**.
3. Go to the ***Advanced*** tab and change the **Custom build file** path to point to the custom build file.

As an example, if you want to call a custom script before running the DITA OT, your custom build file would have the following content:

```
<project basedir="." default="dist">
<!--The DITA OT default build file-->
 <import file="build.xml"/>
 <target name="dist">
  <!-- You could run your script here -->
  <!--<exec></exec>-->
   <!--Call the DITA OT default target-->
   <antcall target="init"/>
 </target>
</project>
```

Note: If you use the built-in Ant 1.8.2 build tool that comes bundled with Oxygen XML Editor, it is located in the *[OXYGEN_INSTALL_DIR]*`/tools/ant` directory. Any additional libraries for Ant must be copied to the Oxygen XML Editor Ant `lib` directory.

DITA Map to PDF Output Customization

Oxygen XML Editor comes bundled with the DITA Open Toolkit that provides a mechanism for converting *DITA maps* to PDF output. There are numerous ways that you can customize the PDF output and the topics in this section discuss some of those possibilities.

Creating a DITA Map to PDF Transformation Scenario

To create a *DITA Map to PDF* transformation scenario, follow these steps:

1. Click the **Configure Transformation Scenario(s)** button from the ***DITA Maps Manager*** toolbar.
2. Select **DITA Map PDF** and click the **Edit** button (or use the **Duplicate** button if your *framework* is read-only).
3. Use the various tabs to configure the transformation scenario. In the **Parameters** tab, you can use a variety of parameters to customize the output. For example, the following parameters are just a few of the most commonly used ones:
   - `show.changes.and.comments` - If set to `yes`, user comments, replies to comments, and *tracked changes* are published in the PDF output.
   - `customization.dir` - Specifies the path to a customization directory.
4. Click **OK** and then the **Apply Associated** button to run the transformation scenario.

Creating a Customization Directory for PDF Output

DITA Open Toolkit PDF output can be customized in several ways:

- Creating a DITA Open Toolkit plugin that adds extensions to the PDF plugin. More details can be found in the *DITA Open Toolkit Documentation*.
- Creating a customization directory and using it from the PDF transformation scenario. A small example of this procedure can be found below.

How to Create a Customization Directory for PDF Output

The following procedure explains how to do a basic customization of the PDF output by setting up a customization directory. An example of a common use case is embedding a company logo image in the front matter of the book.

1. Copy the entire directory: *`DITA-OT-DIR`*`\plugins\org.dita.pdf2\Customization` to another location (for instance, `C:\Customization`).
2. Copy your logo image to: `C:\Customization\common\artwork\logo.png`.
3. Rename `C:\Customization\catalog.xml.orig` to: `C:\Customization\catalog.xml`.
4. Open the `catalog.xml` in Oxygen XML Editor and *uncomment* this line:

   ```
   <!--uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/-->
   ```

   It now looks like this:

   ```
   <uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/>
   ```

5. Rename the file: `C:\Customization\fo\xsl\custom.xsl.orig` to: `C:\Customization\fo\xsl\custom.xsl`
6. Open the `custom.xsl` file in Oxygen XML Editor and create the template called **createFrontCoverContents** for DITA-OT 2.4.4 (or createFrontMatter_1.0 for DITA-OT 1.8.5).

   Tip: You can copy the same template from *DITA-OT-DIR*`\plugins\org.dita.pdf2\xsl\fo\front-matter.xsl` and modify it in whatever way necessary to achieve your specific goal. This new template in the `custom.xsl` file will override the same template from *DITA-OT-DIR*`\plugins\org.dita.pdf2\xsl\fo\front-matter.xsl`.

   DITA OT 2.4.4 Example:

   For example, if you are using DITA OT 2.4.4, the `custom.xsl` could look like this:

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www.w3.org/1999/XSL/Format"
    version="2.0">

<xsl:template name="createFrontCoverContents">
 <!-- set the title -->
 <fo:block xsl:use-attribute-sets="__frontmatter__title">
  <xsl:choose>
   <xsl:when test="$map/*[contains(@class,' topic/title ')][1]">
    <xsl:apply-templates select="$map/*[contains(@class,' topic/title ')][1]"/>
      </xsl:when>
      <xsl:when test="$map//*[contains(@class,' bookmap/mainbooktitle ')][1]">
        <xsl:apply-templates select="$map//*[contains
                                        (@class,' bookmap/mainbooktitle ')][1]"/>
      </xsl:when>
      <xsl:when test="//*[contains(@class, ' map/map ')]/@title">
        <xsl:value-of select="//*[contains(@class, ' map/map ')]/@title"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="/descendant::*[contains
           (@class, ' topic/topic ')][1]/*[contains(@class, ' topic/title ')]"/>
   </xsl:otherwise>
  </xsl:choose>
 </fo:block>

 <!-- set the subtitle -->
 <xsl:apply-templates select="$map//*[contains
                                        (@class,' bookmap/booktitlealt ')]"/>
 <fo:block xsl:use-attribute-sets="__frontmatter__owner">
  <xsl:apply-templates select="$map//*[contains(@class,' bookmap/bookmeta ')]"/>
 </fo:block>

 <!-- Load the image logo -->
  <fo:block text-align="center" width="100%">
   <fo:external-graphic
      src="url({concat($artworkPrefix,
                        '/Customization/OpenTopic/common/artwork/logo.png')})"
    />
  </fo:block>
 </xsl:template>

</xsl:stylesheet>
```

   DITA OT 1.8.5 Example:

   For example, if you are using DITA OT 1.8.5, the `custom.xsl` could look like this:

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www.w3.org/1999/XSL/Format"
    version="1.1">

<xsl:template name="createFrontMatter_1.0">
   <fo:page-sequence master-reference="front-matter"
                        xsl:use-attribute-sets="__force__page__count">
     <xsl:call-template name="insertFrontMatterStaticContents"/>
       <fo:flow flow-name="xsl-region-body">
         <fo:block xsl:use-attribute-sets="__frontmatter">
     <!-- set the title -->
           <fo:block xsl:use-attribute-sets="__frontmatter__title">
            <xsl:choose>
             <xsl:when test="$map/*[contains(@class,' topic/title ')][1]">
     <xsl:apply-templates select="$map/*[contains(@class,' topic/title ')][1]"/>
                 </xsl:when>
    <xsl:when test="$map//*[contains(@class,' bookmap/mainbooktitle ')][1]">
     <xsl:apply-templates select="$map//*[contains
                                        (@class,' bookmap/mainbooktitle ')][1]"/>
         </xsl:when>
    <xsl:when test="//*[contains(@class, ' map/map ')]/@title">
      <xsl:value-of select="//*[contains(@class, ' map/map ')]/@title"/>
         </xsl:when>
      <xsl:otherwise>
```

```
    <xsl:value-of select="/descendant::*[contains
           (@class, ' topic/topic ')][1]/*[contains(@class, ' topic/title ')]"/>
      </xsl:otherwise>
         </xsl:choose>
     </fo:block>

  <!-- set the subtitle -->
     <xsl:apply-templates select="$map//*[contains
                                          (@class,' bookmap/booktitlealt ')]"/>

        <fo:block xsl:use-attribute-sets="__frontmatter__owner">
            <xsl:apply-templates select="$map//*[contains
                                                 (@class,' bookmap/bookmeta ')]"/>
        </fo:block>

        <fo:block text-align="center" width="100%">
           <fo:external-graphic src="url({concat($artworkPrefix,
                        '/Customization/OpenTopic/common/artwork/logo.png')})"/>
         </fo:block>

    </fo:block>

   <!--<xsl:call-template name="createPreface"/>-->

            </fo:flow>
        </fo:page-sequence>
        <xsl:call-template name="createNotices"/>
    </xsl:template>
</xsl:stylesheet>
```

7. Edit the **DITA Map PDF** transformation scenario and in the **Parameters** tab, set the customization.dir parameter to `C:\Customization`.

Tip: For other specific examples, see *DITA-OT 2.3 Documentation - PDF Customization Plugin*.

Related Information:

Automatic PDF plugin customization generator by Jarno Elovirta.

DITA-OT 1.8 Documentation - PDF Customization Plugin

DITA-OT 2.3 Documentation - PDF Customization Plugin

Customizing the Header and Footer in PDF Output

The XSLT stylesheet *`DITA-OT-DIR`*`/plugins/org.dita.pdf2/xsl/fo/static-content.xsl` contains templates that output the static header and footers for various parts of the PDF such as the prolog, table of contents, front matter, or body.

The templates for generating a footer for pages in the body are called `insertBodyOddFooter` or `insertBodyEvenFooter`.

These templates get the static content from resource files that depend on the language used for generating the PDF. The default resource file is *`DITA-OT-DIR`*`/plugins/org.dita.pdf2/cfg/common/vars/en.xml`. These resource files contain variables (such as *Body odd footer*) that can be set to specific user values.

Instead of modifying these resource files directly, they can be overwritten with modified versions of the resources in a PDF customization directory as explained in *Creating a Customization Directory for PDF Output* on page 1690.

Adding a Watermark to PDF Output

To add a watermark to the PDF output of a *DITA map* transformation, create a DITA-OT customization directory and use it from a **DITA Map to PDF** transformation scenario, as in the following procedure:

1. Copy the entire directory: *`DITA-OT-DIR`*`\plugins\org.dita.pdf2\Customization` to another location (for instance, `C:\Customization`).
2. Copy your watermark image (for example, `watermark.png`) to: `C:\Customization\common\artwork\watermark.png`.
3. Rename the `C:\Customization\catalog.xml.orig` file to: `C:\Customization\catalog.xml`.
4. Open the `catalog.xml` in Oxygen XML Editor and *uncomment* this line:

```
<!--uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/-->
```

The uncommented line should look like this:

```
<uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/>
```

5. Rename the file: `C:\Customization\fo\xsl\custom.xsl.orig` to: `C:\Customization\fo\xsl\custom.xsl`
6. Open the `C:\Customization\fo\xsl\custom.xsl` file in Oxygen XML Editor to overwrite two XSLT templates:
   - The first template is located in the XSLT stylesheet *DITA-OT-DIR*`\plugins\org.dita.pdf2\xsl\fo\static-content.xsl` and we override it specifying a watermark image for every page in the PDF content, using a *block-container* element that references the watermark image file:

```
<fo:static-content flow-name="odd-body-header">
       <fo:block-container absolute-position="absolute"
          top="-2cm" left="-3cm" width="21cm" height="29.7cm"
          background-image="{concat($artworkPrefix,
'/Customization/OpenTopic/common/artwork/watermark.png')}">
         <fo:block/>
       </fo:block-container>
      <fo:block xsl:use-attribute-sets="__body__odd__header">
          <xsl:call-template name="insertVariable">
              <xsl:with-param name="theVariableID" select="'Body odd header'"/>
              <xsl:with-param name="theParameters">
                 <prodname>
                     <xsl:value-of select="$productName"/>
                 </prodname>
                 <heading>
               <fo:inline xsl:use-attribute-sets="__body__odd__header__heading">
                      <fo:retrieve-marker retrieve-class-name="current-header"/>
               </fo:inline>
                </heading>
               <pagenum>
             <fo:inline xsl:use-attribute-sets="__body__odd__header__pagenum">
                    <fo:page-number/>
             </fo:inline>
            </pagenum>
        </xsl:with-param>
      </xsl:call-template>
    </fo:block>
  </fo:static-content>

</xsl:template>
```

   - The second template that we override is located in the XSLT stylesheet *DITA-OT-DIR*`\plugins\org.dita.pdf2\xsl\fo\commons.xsl` and is used for styling the first page of the output. We also override it by copying the original template content in our `custom.xsl` and adding the `block-container` element that references the watermark image file:

```
<xsl:template name="createFrontMatter_1.0">
      <fo:page-sequence master-reference="front-matter"
xsl:use-attribute-sets="__force__page__count">
          <xsl:call-template name="insertFrontMatterStaticContents"/>
          <fo:flow flow-name="xsl-region-body">
              <fo:block-container absolute-position="absolute"
                  top="-2cm" left="-3cm" width="21cm" height="29.7cm"
                  background-image="{concat($artworkPrefix,
'/Customization/OpenTopic/common/artwork/watermark.png')}">
                   <fo:block/>
              </fo:block-container>
  <fo:block xsl:use-attribute-sets="__frontmatter">
       <!-- set the title -->
    <fo:block xsl:use-attribute-sets="__frontmatter__title">
     <xsl:choose>
      <xsl:when test="$map/*[contains(@class,' topic/title ')][1]">
    <xsl:apply-templates select="$map/*[contains(@class,' topic/title ')][1]"/>
      </xsl:when>
      <xsl:when test="$map//*[contains(@class,' bookmap/mainbooktitle ')][1]">
    <xsl:apply-templates select="$map//*[contains
(@class,' bookmap/mainbooktitle ')][1]"/>
          </xsl:when>
          <xsl:when test="//*[contains(@class, ' map/map ')]/@title">
              <xsl:value-of select="//*[contains(@class, ' map/map ')]/@title"/>
          </xsl:when>
            <xsl:otherwise>
             <xsl:value-of select="/descendant::*[contains
(@class, ' topic/topic ')][1]/*[contains(@class, ' topic/title ')]"/>
             </xsl:otherwise>
      </xsl:choose>
    </fo:block>

   <!-- set the subtitle -->
   <xsl:apply-templates select="$map//*[contains
(@class,' bookmap/booktitlealt ')]"/>

      <fo:block xsl:use-attribute-sets="__frontmatter__owner">
         <xsl:apply-templates select="$map//*[contains
(@class,' bookmap/bookmeta ')]"/>
      </fo:block>

      </fo:block>
```

```
    <!--<xsl:call-template name="createPreface"/>-->

       </fo:flow>
         </fo:page-sequence>
         <xsl:if test="not($retain-bookmap-order)">
             <xsl:call-template name="createNotices"/>
         </xsl:if>
     </xsl:template>
```

7. Edit your **DITA Map PDF** transformation scenario. In the **Parameters** tab, set the customization.dir parameter to `C:\Customization`.

Related Information:

Adding a Watermark in DITA Map to XHTML Output on page 1697

Force Page Breaks Between Two Block Elements in PDF Output

Suppose that in your DITA content you have two *block elements*, such as two paragraphs:

```
<p>First para</p>
<p>Second para</p>
```

and you want to force a page break between them in the PDF output.

Here is how you can implement a DITA Open Toolkit *plugin* that would achieve this:

1. Define your custom processing instruction that marks the place where a page break should be inserted in the PDF, for example:

```
<p>First para</p>
  <?pagebreak?>
<p>Second para</p>
```

2. Locate the **DITA Open Toolkit** distribution and in the `plugins` directory create a new *plugin* folder (for example, *`DITA-OT-DIR`*`/plugins/pdf-page-break`).
3. In this new folder, create a new `plugin.xml` file with the following content:

```
<plugin id="com.yourpackage.pagebreak">
  <feature extension="package.support.name" value="Force Page Break Plugin"/>
  <feature extension="package.support.email" value="support@youremail.com"/>
  <feature extension="package.version"value="1.0.0"/>
  <feature extension="dita.xsl.xslfo" value="pageBreak.xsl" type="file"/>
</plugin>
```

   The most important feature in the *plugin* is that it will add a new XSLT stylesheet to the XSL processing that produces the PDF content.
4. In the same folder, create an XSLT stylesheet named `pageBreak.xsl` with the following content:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                      xmlns:fo="http://www.w3.org/1999/XSL/Format"version="1.0">
  <xsl:template match="processing-instruction('pagebreak')">
    <fo:block break-after="page"/>
  </xsl:template>
</xsl:stylesheet>
```

5. *Install your plugin in the DITA Open Toolkit.*

Customizing `note` Images in PDF

To customize the images that appear next to each type of note in the PDF output, use a *PDF customization folder* with the following procedure:

1. Copy the *`DITA-OT-DIR`*`/plugins/org.dita.pdf2/cfg/common/vars/en.xml` file to the `[CUSTOMIZATION_DIR]\common\vars` folder.
2. Edit the copied `en.xml` file and modify, for example, the path to the image for `<note>` element with the `type` attribute set to `notice` from:

```
<variable id="notice Note Image
  Path">Configuration/OpenTopic/cfg/common/artwork/important.png</variable>
```

   to:

```
<variable id="notice Note Image
  Path">Customization/OpenTopic/common/artwork/notice.gif</variable>
```

3. Add your custom **notice** image to `[CUSTOMIZATION_DIR]\common\artwork\notice.gif`.
4. Edit the **DITA Map PDF** transformation scenario and in the **Parameters** tab set the path for the customization.dir property to point to the customization folder.

Show Comments and Tracked Changes in PDF Output

To include comments and *tracked changes* (stored within your DITA topics) in the PDF output, follow these steps:

1. Click the **Configure Transformation Scenario(s)** button from the ***DITA Maps Manager*** toolbar.
2. Select **DITA Map PDF** and click the **Edit** button (or use the **Duplicate** button if your *framework* is read-only).
3. In the **Parameters** tab, set the value of the show.changes.and.comments parameter to `yes`.
4. Click **OK** and then the **Apply Associated** button to run the transformation scenario.

   Result: Comment threads and tracked changes will now appear in the PDF output. Details about each comment or change will be available in the footer section for each page.

Set a Font for PDF Output Generated with FO Processor

When a *DITA map* is transformed to PDF using an FO processor and it contains some Unicode characters that cannot be rendered by the default PDF fonts, a font that is capable of rendering these characters must be configured and embedded in the PDF result.

The settings that must be modified for configuring a font for the built-in FO processor are detailed in *Add a Font to the Built-in FO Processor* on page 937.

DITA OT PDF Font Mapping

The DITA OT contains a file *`DITA-OT-DIR`*`/plugins/org.dita.pdf2/cfg/fo/font-mappings.xml` that maps logical fonts used in the XSLT stylesheets to physical fonts that will be used by the FO processor to generate the PDF output.

The XSLT stylesheets used to generate the XSL-FO output contain code like this:

```
<xsl:attribute name="font-family">monospace</xsl:attribute>
```

The font-family is defined to be *monospace*, but *monospace* is just an alias. It is not a physical font name. Therefore, another stage in the PDF generation takes this *monospace* alias and looks in the `font-mappings.xml`.

If it finds a mapping like this:

```
<aliases>
      <alias name="monospace">Monospaced</alias>
 </aliases>
```

then it looks to see if the *Monospaced* has a *logical-font* definition and if so, it will use the *physical-font* specified there:

```
<logical-font name="Monospaced">
      <physical-font char-set="default">
        <font-face>Courier New, Courier</font-face>
      </physical-font>
............
</logical-font>
```

Important:

If no alias mapping is found for a font-family specified in the XSLT stylesheets, the processing defaults to **Helvetica**.

Related Information:

DITA Map to PDF WYSIWYG Transformation

Oxygen XML Editor comes bundled with a DITA OT plugin that allows you to convert *DITA maps* to PDF using a CSS layout processor. Oxygen XML Editor also comes bundled with a built-in CSS-based PDF processing engine called **Chemistry**. For those who are familiar with CSS, this makes it very easy to style and customize the PDF output of your DITA projects without having to work with *xsl:fo* customizations.

Oxygen XML Editor supports the following processors (not included in the Oxygen XML Editor installation kit):

- **Oxygen Chemistry** - A built-in processor that is bundled with Oxygen XML Editor.
- **Prince Print with CSS** - A third-party component that needs to be purchased from *http://www.princexml.com*.
- **Antenna House Formatter** - A third-party component that needs to be purchased from *http://www.antennahouse.com/antenna1/formatter/*.

The DITA-OT plugin is located in the following directory: *`DITA-OT-DIR`*`/plugins/com.oxygenxml.pdf.css`.

Although it includes a set of CSS files in its `css` subfolder, when this plugin is used in Oxygen XML Editor, the CSS files located in the `${frameworks}` directory take precedence.

Creating the Transformation Scenario

To create an **DITA Map to PDF WYSIWYG** transformation scenario, follow these steps:

1. Click the **Configure Transformation Scenario(s)** button from the ***DITA Maps Manager*** toolbar.
2. Select **DITA Map PDF - WYSIWYG**.
3. In the **Parameters** tab, configure the following parameters:
   - `css.processor.type` (if you want to use the built-in **Oxygen Chemistry** processor) - Set the value as `chemistry`.
   - `css.processor.path.prince` (if you are using the **Prince Print with CSS** processor) - Specifies the path to the Prince executable file that will be run to produce the PDF. If you installed Prince using its default settings, you can leave this blank.
   - `css.processor.path.antenna-house` (if you are using the **Antenna House Formatter** processor) - Specifies the path to the Antenna House executable file that will be run to produce the PDF. If you installed Antenna House using its default settings, you can leave this blank.
   - `show.changes.and.comments` - When set to `yes`, user comments, replies to comments, and *tracked changes* are published in the PDF output. The default value is no.
   - `dita.css.list` - Allows you to specify a list of CSS URLs to be used by the PDF processor. The files must have URL syntax and be separated using semicolons. If the value is empty, the current selection from the **Styles** drop-down menu is used.
   - `args.css` - Allows you to specify a list of CSS URLs to be used in addition to those specified in the `dita.css.list` parameter OR in addition to the CSS that is currently selected in the **Styles** drop-down menu. The files must have URL syntax and be separated using semicolons. Also, the `dita.css.list` parameter must be left empty to use these files in addition to the selection in the **Styles** drop-down menu.
4. Click **OK** and run the transformation scenario.

Customizing the Styles (for Output and Editing)

If you need to change the styles in the associated CSS, make sure you install Oxygen XML Editor in a folder where you have full read and write privileges (for instance, your user home directory). This is due to the fact that the installed files are usually placed in a read-only folder (for instance, in Windows, Oxygen XML Editor is installed in the `Program Files` folder where the users do not have change rights).

To change the styles of an element you need to create an additional CSS file that will store the customization rules. Once you have created this file, you need to instruct the editor how to use this additional CSS. This can be done in two ways:

- Create an alternate CSS for the DITA document type:
  1. Follow the procedure for adding an alternate CSS file in *Customizing the Main CSS of a Framework* on page 1237.
  2. Once you have configured your CSS as an additional layer, you can *select it from the **Styles** drop-down menu (on the toolbar)*.
  3. Run the **DITA Map PDF - WYSIWYG** transformation scenario and the customization rules from the additional CSS will be visible in the produced PDF.

  This method allows you to have many customization CSS files and simply select the one that you need at any time for both the output and rendering **Author** mode while editing.

- Use the `args.css` parameter that allows you to specify a list of CSS URLs to be used in addition to the `dita.css.list` parameter:

  1. Configure a **DITA map to PDF WYSIWYG** transformation scenario, as described in the procedure *above*.
  2. In the **Parameters** tab, specify the path to your custom CSS files in the `args.css` parameter.
  3. Click **OK** and run the transformation scenario.

  This method is appropriate if you just want to apply the styling customization to the output.

Using the CSS Inspector to See Style Associated with an Element

To find the styles associated with an element, follow this procedure:

1. Open a document in the editor and select **Inspect Styles** from the contextual menu. This opens the ***CSS Inspector*** *view* that shows all the CSS rules that apply to the selected element.
2. Click the particular link for the CSS selector that you need to change and Oxygen XML Editor will open the CSS file and position the cursor at that selector.
3. After you identify the source of the styles you want to modify, copy the CSS rule in your customization CSS file and modify it according to your needs.

   To see the changes in **Author** mode, press **F5** to reload the document.

How to Make Remote Resources Appear in the Output When Using the Prince Processor

If your documentation references external resources (such as images that are stored on a Web-based repository) and your system is behind an HTTP(S) proxy, you may find that they do not appear in the output when using the **Prince Print with CSS** processor. To solve this, follow this procedure:

1. Edit the **`build.xml`** file that is located in *`DITA-OT-DIR`*`/plugins/com.oxygenxml.pdf.css`.
2. There are two instances in the file where a pair of arguments are commented out:

```
<!-- Please remove the comment wrapping the following two arguments
if you are behind a proxy and your documentation refers remote resources,
for example images.

Note: You also need to remove the backslash '\' that appears before the
property name: "http-proxy". That backslash is there because a sequence of
two dashes ('-') is not permited inside a comment.
-->
<!--
  <arg value="-\-http-proxy=${http.proxyHost}:${http.proxyPort}"/>
  <arg value="-\-http-proxy=${https.proxyHost}:${https.proxyPort}"/>
-->
```

3. Remove the comment in both instances.
4. Make sure you also remove the slash that appears before the property name `http-proxy` in each instance.

   Step Result: The arguments should now look like this:

```
<arg value="--http-proxy=${http.proxyHost}:${http.proxyPort}"/>
<arg value="--http-proxy=${https.proxyHost}:${https.proxyPort}"/>
```

5. Save the **`build.xml`** file.
6. Run the *DITA map to PDF WYSIWYG transformation scenario*.

   Result: Your external resources should now appear in your output.

Related Information:

Editing a Transformation Scenario on page 925
Configure Transformation Scenario(s) Dialog Box on page 926
Configuring and Managing Multiple CSS Styles on page 1256
CSS Inspector View on page 232

Adding a Watermark in DITA Map to XHTML Output

To add a watermark to the XHTML output of a *DITA map* transformation, follow these steps:

1. Create a custom CSS stylesheet that includes the watermark image, as in the following example:

```
body {
 background-image: url(MyWatermarkImage.png);
}
```

2. Edit a **DITA Map XHTML** transformation scenario and in the **Parameters** tab set the value of the `args.css` parameter as the path to your watermark image.
3. Set the value of the `args.copycss` parameter to `yes`.
4. Apply the transformation scenario.
5. Copy the watermark image in the output directory of the transformation scenario, next to the CSS file created in step 1.

Related Information:

Adding a Watermark to PDF Output on page 1692

DITA Profiling / Conditional Text

DITA offers support for conditionally profiling content by using profiling attributes. With Oxygen XML Editor, you can define values for the DITA profiling attributes and they can be easily managed to filter content in the published output. You can switch between profile sets to see how the edited content looks like before publishing. The profiling configuration can also be shared between content authors through the project file and there is no need for coding or editing configuration files.

Profiling Attributes

You can profile content elements or map elements by adding one or more of the default DITA profiling attributes (`product`, `platform`, `audience`, `rev`, `props`, and `otherprops`). You can also create your own custom profiling attributes and profiling condition sets. The profiling attributes may contain one or more tokens that represent conditions to be applied to the content when a publication is built.

For example, you could define a section of a topic that would only be included for a publication related to the Windows platform by adding the `platform` profiling attribute:

```
<section platform="windows">
```

For information about creating and editing profiling attributes, see *Creating and Editing Profiling Attributes in DITA* on page 1698 (for information about sharing them, see *Sharing Profiling Attribute Configurations* on page 1700).

Profiling Conditions

DITA allows you to conditionally profile parts of a topic so that certain parts of the topic are displayed when certain profiling conditions are set. Profiling conditions can be set both within topics and in maps. When set in a topic, they allow you to suppress an element (such as paragraph), step in a procedure, item in a list, or even a phrase within a sentence. When set in a map, they allow you to suppress an entire topic or group of topics. You can then create a variety of publications from a single map by applying profiling conditions to the build.

For information about creating and editing condition sets, see *Creating and Editing Profiling Condition Sets in DITA* on page 1701 (for information about sharing them, see *Sharing Condition Set Configurations* on page 1703).

To watch our video demonstration about DITA profiling, go to *https://www.oxygenxml.com/demo/DITA_Profiling.html*.

Creating and Editing Profiling Attributes in DITA

You can filter DITA content or the structure of a document by using profiling attributes or *profiling conditions sets*.

Defining Profiling Attributes for DITA Content

To define or edit profiling attributes for filtering DITA content, follow these steps:

1. *Open the **Preferences** dialog box **(Options** > **Preferences)*** and go to **Editor** > **Edit modes** > **Author** > **Profiling / Conditional Text**.
2. In the ***Profiling Attributes** section*, there are already some default attributes for DITA documents (`audience`, `platform`, `product`, `otherprops`, and `rev`), although if a *Subject Scheme Map* is used for profiling your content, you will see the attributes defined in your *subject scheme map* instead. You can add new attributes and values by clicking the **New** button at the bottom of the table, or customize existing attributes and their values by selecting an attribute and clicking the **Edit** button.

   Step Result: This opens a **Profiling Attribute** configuration dialog box that allows you to define attributes that exist in your schema.
3. In this configuration dialog box, use the **New**, **Edit**, **Delete** buttons to add, edit, or delete possible values of the selected attribute. You can also specify an optional description for each attribute value and you can choose whether the attribute accepts a **Single value** or **Multiple values separated by** a delimiter (DITA only accepts *space* as delimiters for attribute values).
4. Click **OK** to accept your changes.

Result: You should see your changes in the **Profiling Attribute** table.

You can also use the **Profiling Condition Sets** section to apply more complex filters on you DITA content.

Editing Profiling Attribute Values

There are several ways to add values to existing profiling attributes.

- Use the procedure in *Defining Profiling Attributes for DITA Content* on page 1698 to edit an existing attribute and use the **Profiling Attribute** configuration dialog box to add, edit, or delete values for existing profiling attributes.
- You can add values directly to the existing profiling attributes in a document using the *In-Place Attributes Editor* in **Author** mode, the ***Attributes** view*, or in the source code in **Text** mode. However, this just adds them to the document and does not change the conditional text configuration. If you invoke the **Edit Profiling Attributes** action (from the contextual menu in **Author** mode) on the new value, the **Profiling Values Conflict** dialog box will appear and it includes an **Add these values to the configuration** action that will automatically add the new value to the particular profiling attribute. It also includes an **Edit the configuration** action that opens the ***Profiling / Conditional Text** preferences page* where you can edit the profiling configuration. The action selected by default is to preserve the current configuration.

  Note: If the ***Allow additional profiling attribute values collected from the document*** option is not selected in the **Profiling / Conditional Text** preferences page, the **Profiling Values Conflict** dialog box will never appear, the current conditional text configuration will be preserved, and therefore the second method mentioned above will not be available.

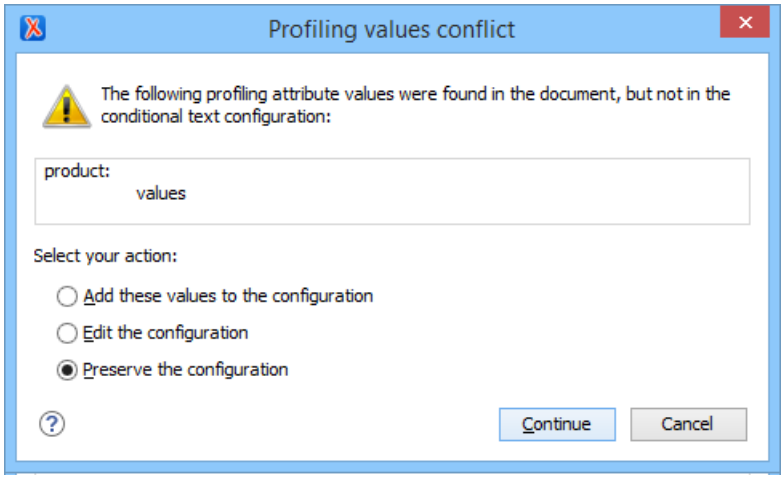


Figure 748: Profiling Values Conflict Dialog Box

Sharing Profiling Attribute Configurations

Your profiling configuration can be shared with other users through a project file. If you select ***Project Options*** at the bottom of the **Profiling/Conditional Text** preferences page, your configuration is stored in the project file and can be shared with others. For instance, if your project file is saved on a version control system (such as SVN, CVS, or Source Safe) or in a shared folder, your team will have the same option configuration that you stored in the project file.

For more information about sharing project files, see *Sharing a Project - Team Collaboration* on page 284.

Related Information:

Apply Profiling Attributes in DITA on page 1700
Creating and Editing Profiling Condition Sets in DITA on page 1701
Apply Profiling Condition Sets in DITA on page 1703
Showing and Filtering Profiled Content in DITA on page 1705
Customizing Colors and Styles for Rendering Profiling in Author Mode on page 1707

Apply Profiling Attributes in DITA

Profiling attributes are applied on element nodes. You can apply profiling attributes on a text fragment (it will automatically be wrapped into a phrase-type element), on a single element, or on multiple elements in the same time. If there is no selection in your document, the profiling attributes are applied on the element at the cursor position.

You can apply *defined DITA profiling attributes* as follows:

DITA Topics

To profile DITA topics, right-click a topic reference in the ***DITA Maps Manager***, select **Edit Properties** from the contextual menu, go to the **Profiling** tab, and select the appropriate values.

DITA Content

To profile DITA content in **Author** mode, highlight the content and select **Edit Profiling Attributes** from the contextual menu and select the appropriate values in the **Edit Profiling Attributes** dialog box.

DITA Elements

To profile specific XML elements in **Author** mode, position the cursor inside the element, right-click, select **Edit Profiling Attributes** (you can also right-click the element in the *breadcrumb* or ***Outline*** view), and select the appropriate values in the **Edit Profiling Attributes** dialog box. You can also use the ***Attributes*** *view* to set the profiling attributes on the element at the current cursor position.

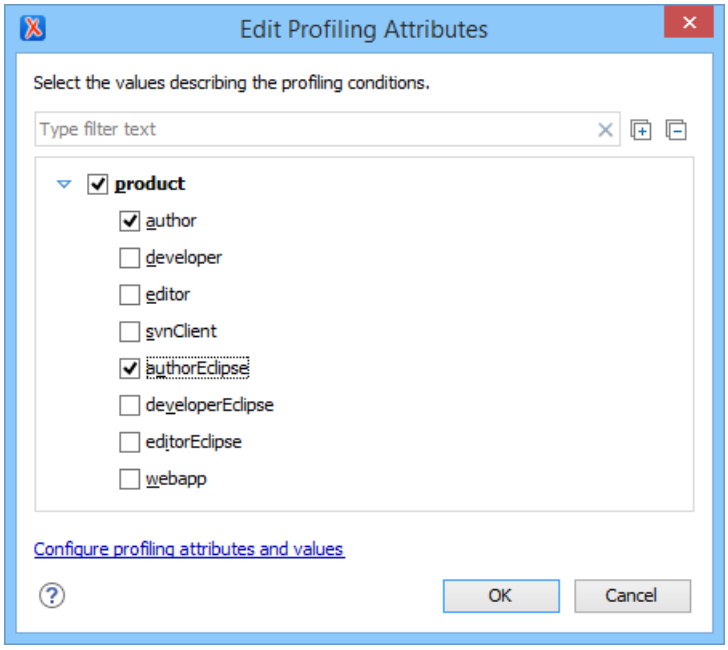


Figure 749: Edit Profiling Attributes Dialog Box

The profiling attributes, and their potential values, that appear in this dialog box depend on what has been configured in Oxygen XML Editor. If you have a large list of profiling attributes, you can use the text filter field to search for attributes or values, and you can expand or collapse attributes by using the **Expand All/ Collapse All** buttons to the right of the text filter or the arrow button to the left of the profiling attribute name.

The attributes and values that appear in the dialog box are determined as follows:

- If your *root map* references a DITA *subject scheme map* that *defines values for the profiling attributes*, those values are used. Oxygen XML Editor collects all the profiling values from the *subject scheme map* that is referenced in the map that is currently opened in the ***DITA Maps Manager*** (or *set as the root map*). In the *image above* (taken from the Oxygen XML Editor documentation project), you see values for eight products. They are the only values that are defined in the *subject scheme map* and thus, are the only ones that appear in the dialog box.
- If you have *defined profiling attribute values* for the DITA document type in the ***Profiling/Conditional Text*** *preferences page* and you store them at *project-level*, those values are displayed in the dialog box.
- If you have *defined profiling attribute values* for the DITA document type in the ***Profiling/Conditional Text*** *preferences page* and you store them at *global-level*, those values are displayed in the dialog box.
- Otherwise, a generic default set of profiling attributes and values are available.

If the ***Show Profiling Attributes*** *option* (available in the **Profiling / Conditional Text** toolbar menu) is selected, a green border is painted around profiled text in the **Author** mode and all profiling attributes set on the current element are listed at the end of the highlighted block. To edit the attributes of a profiled fragment, click one of the listed attribute values. A form control pops up and allows you to add or remove attribute values.

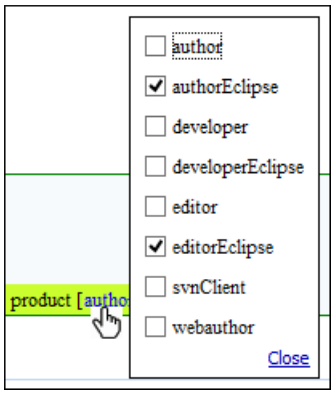


Figure 750: Profiling Attribute Value Form Control Pop Up

Related Information:

Creating and Editing Profiling Attributes in DITA on page 1698
Creating and Editing Profiling Condition Sets in DITA on page 1701
Apply Profiling Condition Sets in DITA on page 1703
Showing and Filtering Profiled Content in DITA on page 1705
Customizing Colors and Styles for Rendering Profiling in Author Mode on page 1707

Creating and Editing Profiling Condition Sets in DITA

In DITA, profiling conditions can be set both within topics and in maps. When set in a topic, they allow you to suppress an element (such as paragraph), step in a procedure, item in a list, or even a phrase within a sentence. When set in a map, they allow you to suppress an entire topic or group of topics.

Creating Profiling Condition Sets

To create a new profiling condition set, follow these steps:

1. *Open the* ***Preferences*** *dialog box* ***(Options > Preferences)*** and go to **Editor** > **Edit modes** > **Author** > **Profiling/ Conditional Text**.
2. In the ***Profiling Condition Sets*** *section*, press the **New** button.

Step Result: The **Condition Set** configuration dialog box is opened.

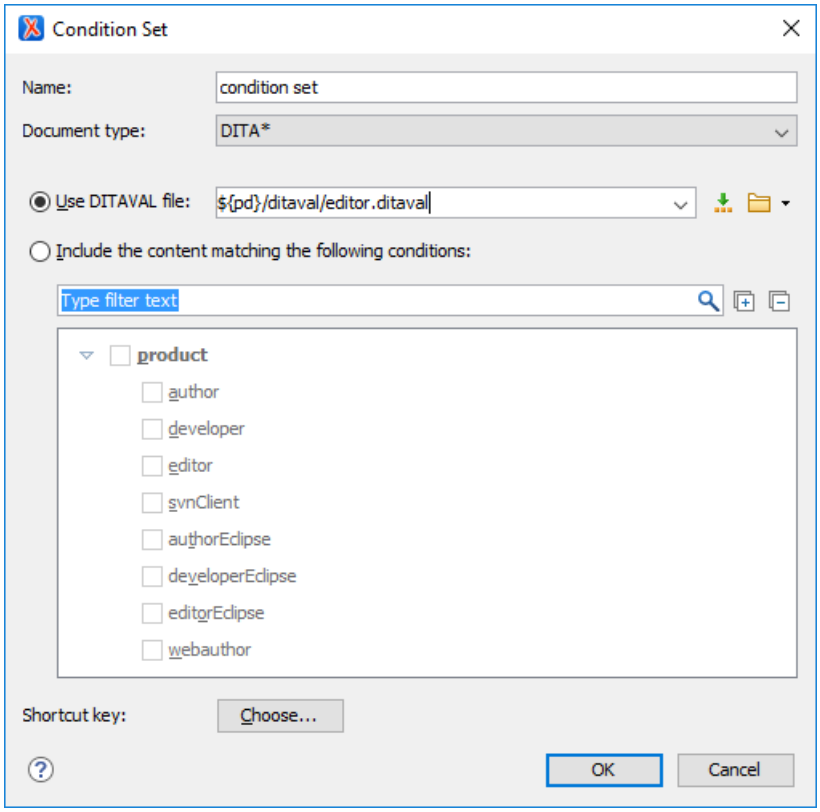


Figure 751: Condition Set Dialog Box

3. Configure your condition set as desired. The following options are available in this dialog box:

   Name

   The name of the new condition set.

   Document type

   Select the document type (*framework*) for which you have defined profiling attributes.

   Use DITAVAL file

   Select this option if you want the *Profiling Condition Set* to reference a *DITAVAL file*. You can specify the path by using the text field, its history drop-down, the ***Insert Editor Variables*** button, or the browsing tools in the **Browse** drop-down list.

   Include the content matching the following conditions

   You can select this option to define the combination of attribute values for your condition set by selecting the appropriate checkboxes for the values you want to be included in this particular condition set. If you have defined a lot of profiling attributes, you can use the **filter** text field to search for specific conditions.

   Shortcut key

   You can click the **Choose** button to open a dialog box that allows you to define a shortcut key for this particular condition set. You can then use that shortcut key anytime you want to select this condition set to filter content.

4. Click **OK** to confirm your selections and close the **Condition Set** configuration dialog box.
5. Click **Apply** to save the condition set. All saved profiling condition sets are available in the ***Profiling / Conditional Text*** *toolbar drop-down menu*.

Editing Existing Profiling Condition Sets

To modify an existing profiling condition set, follow these steps:

1. *Open the **Preferences** dialog box **(Options > Preferences)*** and go to **Editor** > **Edit modes** > **Author** > **Profiling/ Conditional Text**.
2. In the ***Profiling Condition Sets** section*, press the **Edit** button to modify an existing condition set (you can also use **Delete** button to remove a condition set or the **Up** and **Down** buttons to change their priority).

   Step Result: If you use the **Edit** button, the **Condition Set** configuration dialog box is opened:
3. Modify your condition set as desired.
4. Click **OK** to confirm your selections and close the **Condition Set** configuration dialog box.
5. Click **Apply** to save your modifications.

Sharing Condition Set Configurations

Your condition set configuration can be shared with other users through a project file. If you select ***Project Options*** at the bottom of the **Profiling/Conditional Text** preferences page, your configuration is stored in the project file and can be shared with others. For instance, if your project file is saved on a version control system (such as SVN, CVS, or Source Safe) or in a shared folder, your team will have the same option configuration that you stored in the project file.

For more information about sharing project files, see *Sharing a Project - Team Collaboration* on page 284.

Related Information:

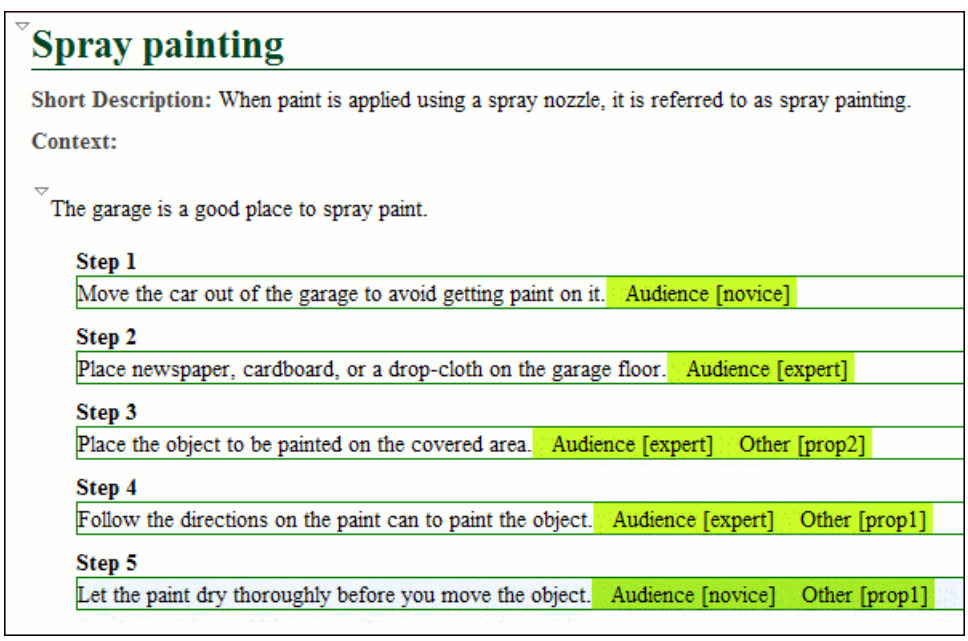
Apply Profiling Condition Sets in DITA on page 1703
Creating and Editing Profiling Attributes in DITA on page 1698
Apply Profiling Attributes in DITA on page 1700
Showing and Filtering Profiled Content in DITA on page 1705
Customizing Colors and Styles for Rendering Profiling in Author Mode on page 1707

Apply Profiling Condition Sets in DITA

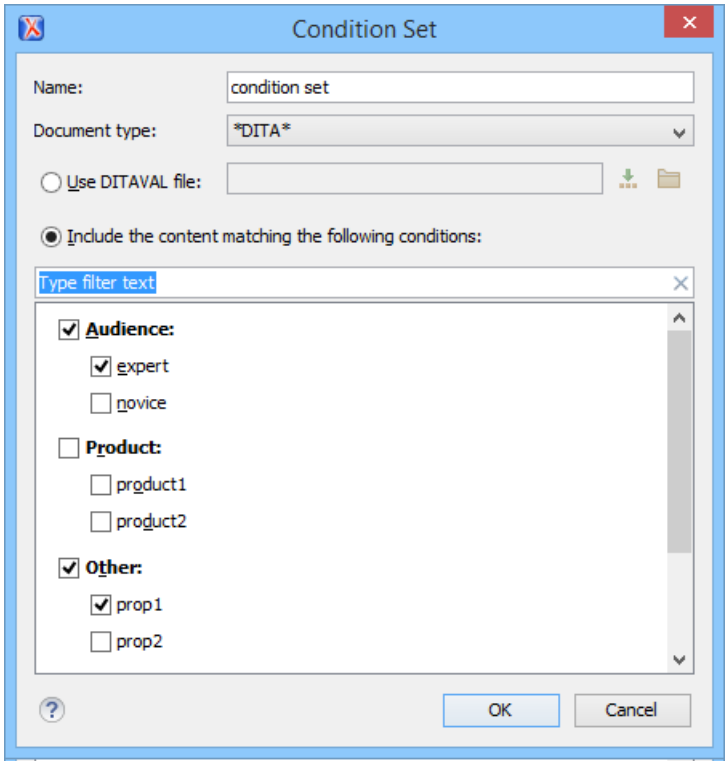
All defined *Profiling Condition Sets* are available as shortcuts in the ***Profiling / Conditional Text** toolbar menu*. Select a menu entry to apply the condition set. The filtered content is then grayed-out in the **Author** mode, ***Outline** view*, and ***DITA Maps Manager** view* (for DITA documents). Your selection will also be used as the *default condition set* in transformation scenarios. An element is filtered-out when one of its attributes is part of the condition set and its value does not match any of the value covered by the condition set.

EXAMPLE:

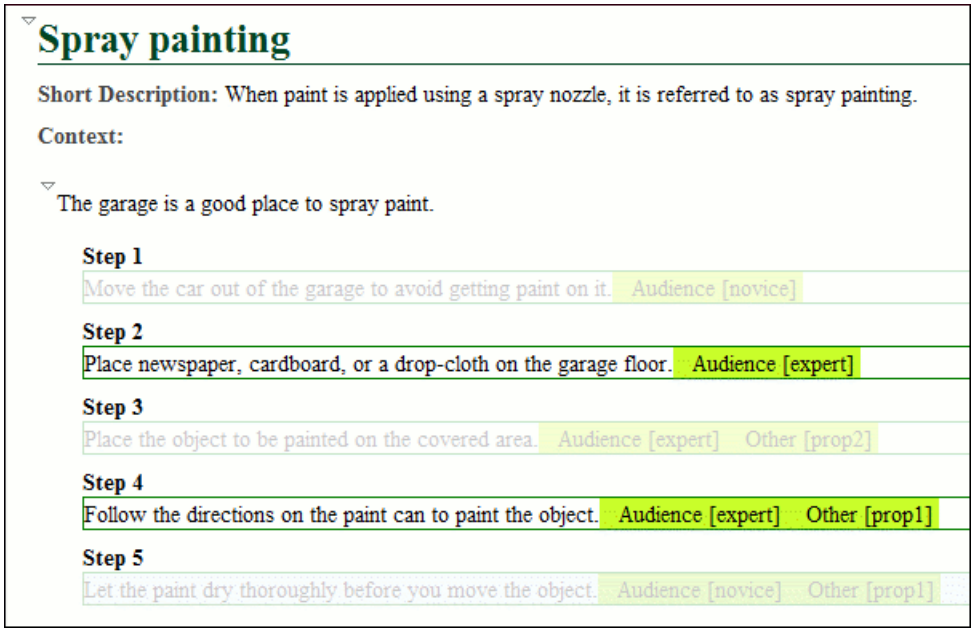
Suppose that you have the following document:



If you apply the following condition set, it means that you want to filter out the content to only include content written for an expert audience and content that has the *Other* attribute value of *prop1*.



This is how the document looks after you apply the condition set:



Related Information:

Creating and Editing Profiling Condition Sets in DITA on page 1701
Creating and Editing Profiling Attributes in DITA on page 1698
Apply Profiling Attributes in DITA on page 1700
Showing and Filtering Profiled Content in DITA on page 1705
Customizing Colors and Styles for Rendering Profiling in Author Mode on page 1707

Showing and Filtering Profiled Content in DITA

You can visualize the effect of profiling content by using the profiling tools in the **Profiling/Conditional Text** drop-down menu that is located on the ***DITA Maps Manager*** toolbar and on the main toolbar. This drop-down menu includes the following filtering options:

Show Profiling Colors and Styles

Select this option to show colors and styles for profiled content in **Author** mode and the ***DITA Maps Manager***. You can configure the colors and styles or specify whether or not this option is selected by default in the ***Profiling/Conditional Text*** > ***Colors and Styles*** *preferences page*.

Show Profiling Attributes

Select this option to display the values of the profiling attributes at the end of profiled content in **Author** mode and next to the nodes in the ***DITA Maps Manager***. You can specify whether or not this option is selected by default in the ***Profiling/Conditional Text*** > ***Attributes Rendering*** *preferences page*.

Show Excluded Content

Controls whether the content filtered out by a particular condition set is hidden or grayed-out in **Author** mode, the ***DITA Maps Manager***, and the ***Outline*** view. When this option is selected and a *condition set is selected in this drop-down menu*, the filtered content is grayed-out. If this option is not selected and a *condition set is selected in this drop-down menu*, the filtered content is hidden. You can specify whether or not this option is selected by default in the ***Profiling/Conditional Text*** *preferences page*.

Choose Condition Set (Available if more than 15 condition sets are defined)

This option is available if you have more than 15 conditions sets defined. It opens a dialog box that makes it easier to find and select condition sets that are not displayed in this drop-down menu.

List of Defined Condition Sets

Up to 15 defined condition sets are listed and you can toggle each one of them on to filter the content in **Author** mode and the ***DITA Maps Manager*** to only show content that will appear in the output for that particular condition set. If there are more than 15 defined condition sets, the rest of them can be accessed in the **More** submenu or by using the ***Choose Condition Set*** *option* to access a dialog box that presents all of them.

Profiling Settings

Opens the ***Profiling/Conditional Text*** *preferences page* where you can add and edit profiling attributes and condition sets.

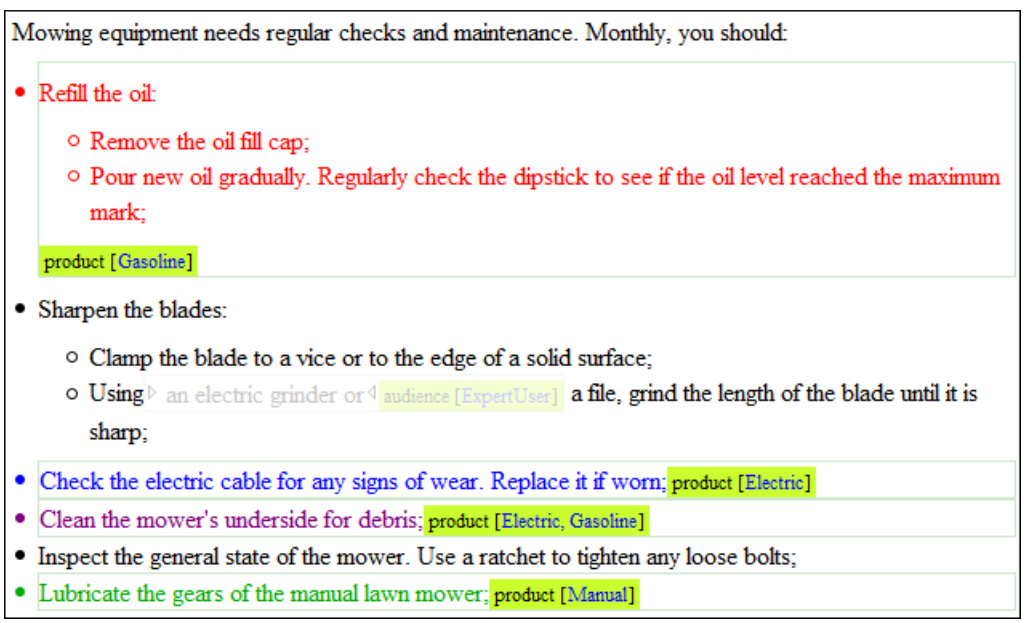


Figure 752: Example: Profiled Content in Author Mode

If the ***Show Profiling Attributes*** *option* is selected, a green border is painted around profiled text in the **Author** mode. Also, all profiling attributes set on the current element are listed at the end of the highlighted block and in

its tooltip message. To edit the attributes of a profiled fragment, click one of the listed attribute values. A form control pops up and allows you to add or remove attribute values.

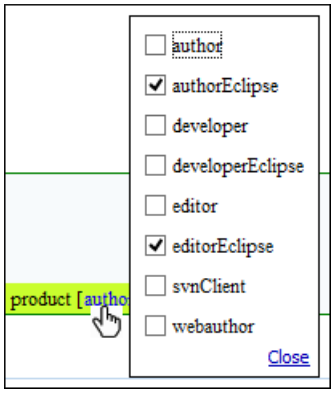


Figure 753: Profiling Attribute Value Form Control Pop Up

Also, the following icons are used to mark profiled and non-profiled topics in the **DITA Maps Manager**:

- - The topic contains profiling attributes.
- - The topic inherits profiling attribute from its ancestors.
- - The topic contains and inherits profiling attributes.
- - (dash) - The topic neither contains, nor inherits profiling attributes.

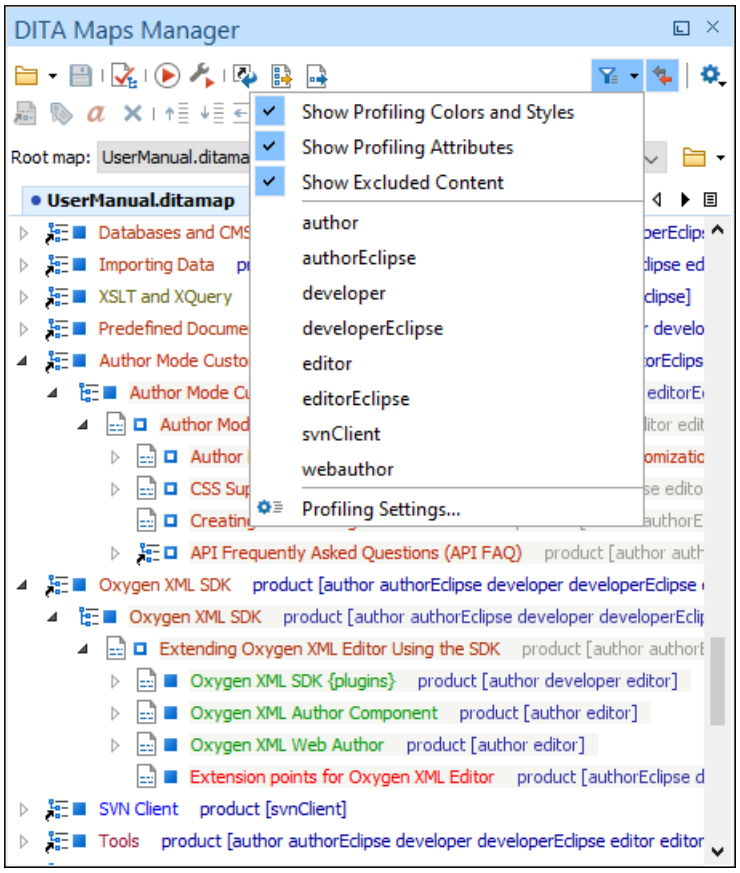


Figure 754: Rendering Profiled Topics in DITA Maps Manager

Related Information:

Creating and Editing Profiling Condition Sets in DITA on page 1701
Apply Profiling Attributes in DITA on page 1700
Creating and Editing Profiling Attributes in DITA on page 1698
Apply Profiling Condition Sets in DITA on page 1703

Customizing Colors and Styles for Rendering Profiling in Author Mode on page 1707

Customizing Colors and Styles for Rendering Profiling in Author Mode

By applying profiling colors and styles, you can mark profiled content in **Author** mode and the ***DITA Maps Manager*** so that you can instantly spot differences between multiple variants of the output. This allows you to preview the content that will go into the published output. The excluded text is grayed-out or hidden in **Author** mode and excluded nodes are grayed-out or hidden in the ***DITA Maps Manager***.

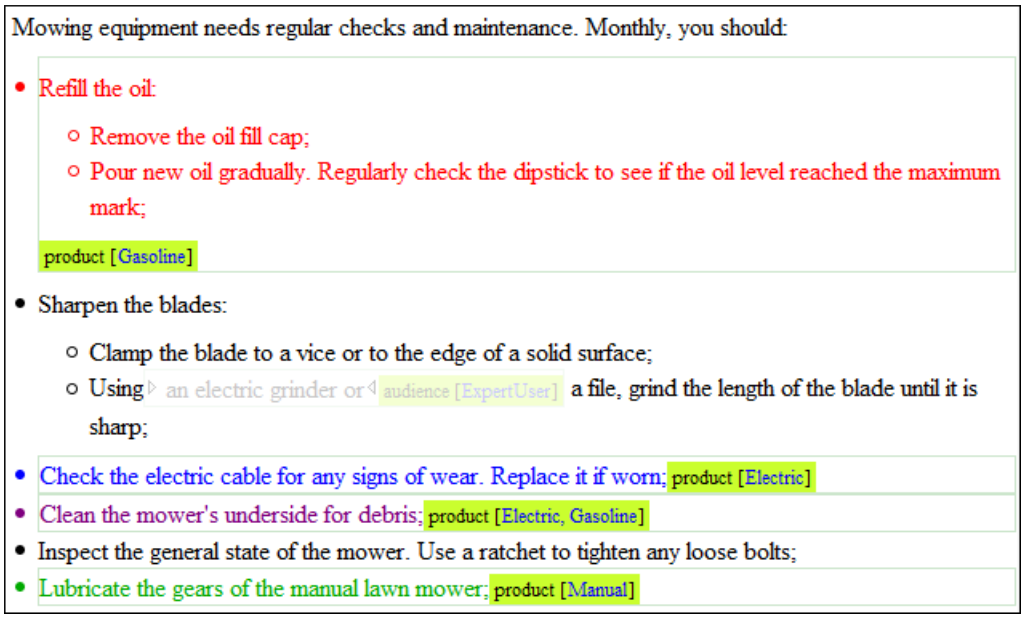


Figure 755: Example: Profiling Colors and Styles in Author Mode

Choosing the right style for a specific profiling attribute is a matter of personal taste, but be aware of the following:

- If the same block of text is profiled with two or more profiling attributes, their associated styles combine. Depending on the styling, this might result in an excessively styled content that may prove difficult to read or work with.
- It is recommended that you only profile the differences. There is no need to profile common content, since excessive profiling can visually pollute the document.
- A mnemonic associated with a style will help you instantly spot differences in the types of content.

Styling Profiling Attribute Values

To set colors and styles for profiling attribute values, follow these steps:

1. Select the ***Show Profiling Colors and Styles*** *option* from the **Profiling / Conditional Text** toolbar drop-down menu.
2. Select ***Profiling Settings*** from the **Profiling / Conditional Text** toolbar drop-down menu. This is a shortcut to the ***Profiling/Conditional Text*** *preferences page*.
3. Go to the ***Colors and Styles*** *preferences page* to configure the colors and styling for the profiling attributes.
4. Go to the ***Attributes Rendering*** *preferences page* to configure how you want the profiling attributes to appear in Oxygen XML Editor.

Result: The styling is now applied in the **Author** editing mode, the ***Outline*** *view*, and in the ***DITA Maps Manager*** *view*. Also, to help you more easily identify the profiling you want to apply in the current context, the styling is applied in the ***Edit Profiling Attributes*** *dialog box* and in the inline form control pop up that allows you to quickly set the profiling attributes.

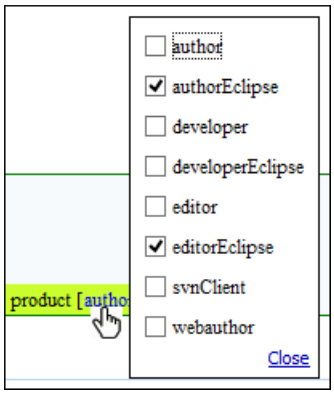


Figure 756: Profiling Attribute Value Form Control Pop Up

Alternate Method with a DITAVAL File: If you are using a DITAVAL filter file to control the filtering of profiled content in DITA topics, you can use a flag filter to define the colors and styles that will be used when rendering the profiling. For detailed information about this alternate method, see the procedure in the *Styling the Rendering of Profiled Content Using a DITAVAL File* on page 1711 topic.

Related Information:

Creating and Editing Profiling Condition Sets in DITA on page 1701

Apply Profiling Attributes in DITA on page 1700

Creating and Editing Profiling Attributes in DITA on page 1698

Apply Profiling Condition Sets in DITA on page 1703

Showing and Filtering Profiled Content in DITA on page 1705

Profiling with a Subject Scheme Map

A *subject scheme map* allows you to create custom profiling values and to manage the profiling attribute values used in the DITA topics without having to write a DITA specialization.

Subject scheme maps use key definitions to define a collection of profiling values. A map that uses the set of profiling values must reference the *subject scheme map* in which the profiling values are defined at its highest level. To do this, you must add the `type="subjectScheme"` attribute on the `topicref` that references the *subject scheme map*, as in the following example:

```
<topicref href="test.ditamap" format="ditamap" type="subjectScheme"/>
```

A profiled value should be a short and readable keyword that identifies a metadata attribute. For example, the `audience` metadata attribute may take a value that identifies the user group associated with a particular content unit. Typical user values for a medical-equipment product line might include `therapist`, `oncologist`, `physicist`, `radiologist`, `surgeon`, and so on. A *subject scheme map* can define a list of these audience values.

EXAMPLE:

The following is an example of content from a *subject scheme*:

```
<subjectScheme>
  <!-- Pull in a scheme that defines audience user values -->
  <subjectdef keys="users">
    <subjectdef keys="therapist"/>
    <subjectdef keys="oncologist"/>
    <subjectdef keys="physicist"/>
    <subjectdef keys="radiologist"/>
    <subjectdef keys="surgeon">
      <subjectdef keys="neuro-surgeon"/>
      <subjectdef keys="plastic-surgeon"/>
    </subjectdef>
  </subjectdef>
  <!-- Define an enumeration of the audience attribute, equal to
       each value in the users subject. This makes the following values
       valid for the audience attribute: therapist, oncologist, physicist,
       radiologist, surgeon, neuro-surgeon and plastic-surgeon. -->
  <enumerationdef>
    <attributedef name="audience"/>
    <subjectdef keyref="users"/>
```

```
  </enumerationdef>
</subjectScheme>
```

Where the Profiling Attributes are Available in Oxygen XML Editor

When you edit a DITA topic in the **Text** or **Author** mode, Oxygen XML Editor collects all the profiling values from the *subject scheme map* that is referenced in the map that is currently opened in the ***DITA Maps Manager*** (or set as the *root map*). The values of profiling attribute defined in a *Subject Scheme Map* are available in the ***Edit Profiling Attribute*** *dialog box*, regardless of their mapping in the ***Profiling/Conditional Text*** *preferences page*. They are also available as proposals for values in the *Content Completion Assistant*.

Note: In the example above, the values `therapist`, `oncologist`, `physicist`, and so on, are displayed in the *Content Completion Assistant* as values for the `audience` attribute.

Filtering Attribute Values

By defining controlled values and using hierarchical levels in the *subject scheme map*, you can classify content for filtering and flagging when the map is transformed. You can also filter attribute values by using a *DITAVAL filter file*.

For more details about how hierarchical filtering, defining controlled values, and using *subject scheme maps*, refer to the following resources:

- *DITA 1.3 Specifications: Subject Scheme Maps*
- *Oxygen Video Tutorial: DITA Subject Scheme*

Related Information:

Filtering Attribute Values with a DITAVAL File on page 1709

Filtering Attribute Values with a DITAVAL File

You can use a DITAVAL filter file to control the filtering or flagging of profiled content or to identify which values are to be used for conditional processing during a particular output.

DITAVAL Filtering Use-Case

Suppose that a medical publication uses the `audience` attribute to profile the content for the following types of users: *therapist*, *physician*, and *surgeon*. Suppose that in the output, you want to exclude any content that is profiled as `surgeon` value for the `audience` attribute.

You could use a DITAVAL filter file to exclude anything that is profiled as `surgeon`:

```
<val>
    <prop action="exclude" att="audience" val="surgeon"/>
</val>
```

If you then transform the main *DITA map* and specify the DITAVAL filter file in the transformation scenario, the output will exclude anything that is profiled as `surgeon`).

DITAVAL Filter File Editor in Author Mode

The **Author** editing mode in Oxygen XML Editor offer a simple and intuitive editor for creating or modifying DITAVAL files. It provides a series of drop-down menus and text fields that allow you to easily define the filters.

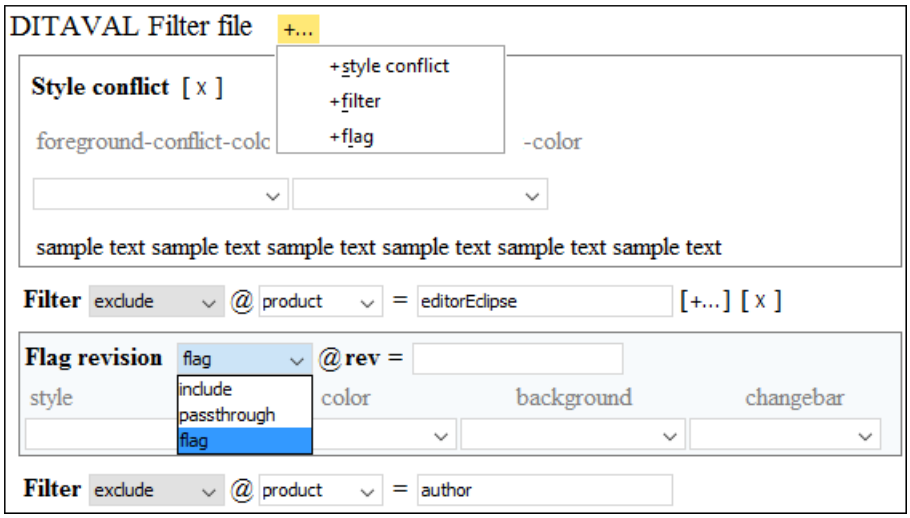


Figure 757: DITAVAL File Editor in Author Mode

Use the **+...** button to display a drop-down list that contains elements that you can add at that particular location in the DITAVAL file. Clicking this button at the top (next to the **DITAVAL FILTER File** title, allows you to insert the following elements:

- **Style Conflict** - Inserts a *`style-conflict` element* that declares behavior to be used when one or more flagging methods collide on a single content element. You can use the simple drop-down menus to select values for the `foreground-conflict-color` and `background-conflict-color` attributes.
- **Filter** - Inserts a *`prop` element* that identifies an attribute to apply a filtering action on. The possible actions that you can select are *include*, *exclude*, *passthrough*, and *flag*. If you select the *flag* action, you can use the drop-down menus to select values for the `style`, `color`, and `background` attributes.
- **Flag** - Inserts a *`revprop` element* that Identifies a value in the `rev` attribute that should be flagged in some manner. The allows actions are *include*, *passthrough*, and *flag*. If you select the *flag* action, you can use the drop-down menus to select values for the `style`, `color`, `background`, and `changebar` attributes.

See the *DITAVAL Element Specifications* for more details about the allowed filters and flags.

How to Create a DITAVAL Filter File

To create a DITAVAL filter file, follow these steps:

1. Go to **File** > **New**.
2. Scroll to the **Framework templates** > **DITAVAL** folder.
3. Select the **Filter** template file and click **Create**.
4. Define your filters in the DITAVAL file (in **Text** or **Author** mode).
5. Save the DITAVAL file.

Result: The DITAVAL filter file can now be used for all of the following:

- To apply a reference to the DITAVAL file in a *Profiling Condition Set* using the ***Use DITAVAL File*** *option* in the ***Condition Set*** *configuration dialog box*.
- You can use the ***Import from DITAVAL*** *option* in the ***Profiling/Conditional Text*** *preferences page* to use the DITAVAL file to define profiling attributes.
- You can use the DITAVAL file to apply the filters to the output by specifying the *DITAVAL file in the transformation scenario*.
- You can use the filter file in the ***DITA Map Completeness Check*** *dialog box* when validating your *DITA map*.
- DITAVAL files are also used when working with the DITA 1.3 *Branch Filtering* mechanism. For more details, see: *Working with DITA 1.3 Branch Filtering* on page 1670.
- You can *define the colors and styles to be used for rendering profiled condition sets* in **Author** mode and the ***DITA Maps Manager*** view by using a **Flag** filter in the DITAVAL file.

Related Information:

DITAVAL Element Specifications

Working with DITA 1.3 Branch Filtering on page 1670
Profiling with a Subject Scheme Map on page 1708
Styling the Rendering of Profiled Content Using a DITAVAL File on page 1711

Styling the Rendering of Profiled Content Using a DITAVAL File

If you are using a DITAVAL filter file to control the filtering of profiled content, you can define the colors and styles to be used for rendering profiled condition sets in **Author** mode and the ***DITA Maps Manager*** by defining the styles in a flag filter that is set in a DITAVAL filter file.

How to Define a Flag for a Condition Set in a DITAVAL Filter File

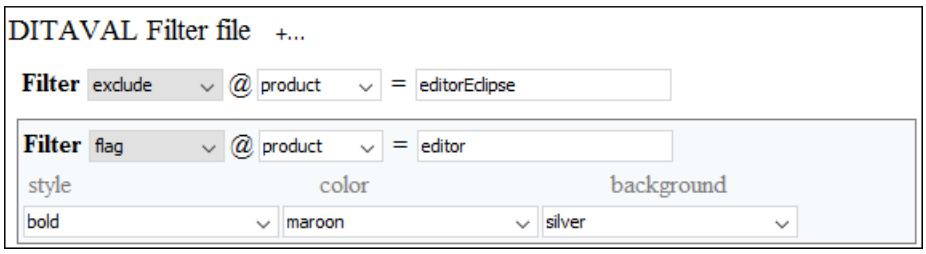
To define the colors and styles to be used for rendering profiled condition sets by using a flag filter in a DITAVAL filter file, follow these steps:

1. *Create or edit your DITAVAL file* to define your *profiling condition set*.
2. In **Author** mode, *define the filters for your condition set*.
3. Select **Flag** from the drop-down menu on in a particular **Filter** or **Flag Revision** to present additional drop-down menus that allow you to configure the colors and styles for the particular condition set.
4. Save the DITAVAL file.

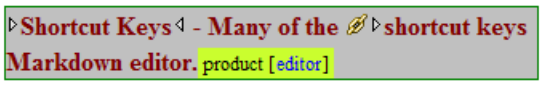
Result: Test your changes by opening profiled content in **Author** mode or the ***DITA Maps Manager*** and use the options in the **Profiling / Conditional Text** drop-down menu to see how the changes in your DITAVAL flag are rendered.

EXAMPLE:

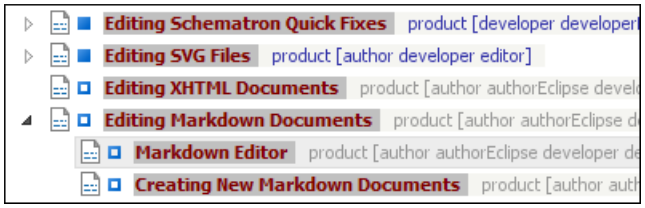
Using a **Flag** on a **Filter** to define the styling for a condition set like this:



will render the styling of the profiled content in **Author** mode to look like this:



and will render the styling in the ***DITA Maps Manager*** *view* to look like this:



Related Information:

Filtering Attribute Values with a DITAVAL File on page 1709

Publishing Profiled DITA Content

You can create a variety of publications or versions of your documentation from a single map by applying profiling conditions to the build.

Oxygen XML Editor includes preconfigured transformation scenarios for DITA. By default, these scenarios take the current *profiling condition set* into account during the transformation, as defined in the ***Filters** tab* when *creating a DITA transformation*.

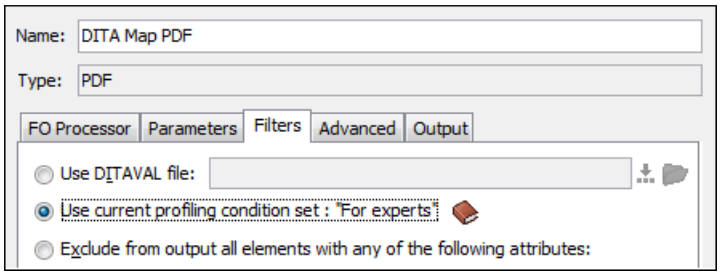


Figure 758: Profiling Option in the Filters Tab (DITA OT Transformations)

DITA Open Toolkit Support

Oxygen XML Editor includes support for the DITA Open Toolkit. This section includes information about how to create a DITA OT *plugin*, how to install *plugins* in the DITA OT, and how to use an external instance of the DITA Open Toolkit.

Related Information:

DITA Open Toolkit Documentation

Creating a DITA OT Customization Plugin

To describe the steps involved in creating a **DITA Open Toolkit** *plugin*, this section uses an example of creating an **XSLT** customization *plugin* that provides syntax highlighting when publishing DITA `codeblock` elements to **HTML** and **PDF** output formats. This *plugin* (**com.oxygenxml.highlight**) that provides syntax highlighting is available in the **DITA Open Toolkit** distribution that comes bundled with the latest version of Oxygen XML Editor, but these instructions show you how to create it as if it were not included.

The steps to help you to create the *plugin* are as follows:

1. Create a folder for your *plugin* in the DITA OT **plugins** folder (*`DITA-OT-DIR`*`/plugins/`).

   For example, if you are using DITA 1.8.5, the path would look like this:

   `[OXYGEN_INSTALL_DIR]``/frameworks/dita/DITA-OT/plugins/com.oxygenxml.highlight`
2. Create a **plugin.xml** file (in the same *plugin* folder) that contains the extension points of the *plugin*.

   Note: You can easily create this file by using the **DITA OT Plugin** new file template that is included in Oxygen XML Editor. From the **New** file wizard you can find this template in **Framework templates** > **DITA** > **plugin**.

   For example, our syntax highlighting plugin example contains the following:

```
<plugin id="com.oxygenxml.highlight">
  <feature extension="package.support.name" value="Oxygen XML Editor Support"/>
  <feature extension="package.support.email" value="support@oxygenxml.com"/>
  <feature extension="package.version" value="1.0.0"/>
  <feature extension="dita.xsl.xhtml" value="xhtmlHighlight.xsl" type="file"/>
  <feature extension="dita.xsl.xslfo" value="pdfHighlight.xsl" type="file"/>
</plugin>
```

   The most important extensions in it are the references to the XSLT stylesheets that will be used to style the HTML and PDF outputs.

   You can find other **DITA OT** *plugin* extension points here: *http://dita-ot.sourceforge.net/1.5.3/dev_ref/extension-points.html*
3. Create an XSLT stylesheet to customize the output types. In our example, to customize the HTML output we need to create an XSLT stylesheet called **xhtmlHighlight.xsl** (in the same *plugin* folder).

   Tip: You can use the ***Find/Replace in Files** action* to find an XSLT stylesheet with content that is similar to the desired output and use it as a template to overwrite parts of your stylesheet. In our example we want to overwrite the creation of the HTML content from a DITA **codeblock** element. Since a DITA `codeblock`

element has the `class` attribute value `+ topic/pre pr-d/codeblock` we can take part of the class attribute value (**topic/pre**) and search the **DITA OT** resources for a similar stylesheet.

Our search found the XSLT stylesheet *DITA-OT-DIR*`/org.dita.xhtml/xsl/xslhtml/dita2htmlImpl.xsl` that contains:

```
<xsl:template match="*[contains(@class,' topic/pre ')]" name="topic.pre">
  <xsl:apply-templates select="."  mode="pre-fmt" />
</xsl:template>
```

We use it to overwrite our **xhtmlHighlight.xsl** stylesheet, which results in the following:

```
<xsl:template match="*[contains(@class,' topic/pre ')]" name="topic.pre">
    <!-- This template is deprecated in DITA-OT 1.7.
         Processing will moved into the main element rule. -->
    <xsl:if test="contains(@frame,'top')"><hr /></xsl:if>
    <xsl:apply-templates select="*[contains(@class,' ditaot-d/ditaval-startprop ')]"
          mode="out-of-line"/>
    <xsl:call-template name="spec-title-nospace"/>
    <pre>
      <xsl:attribute name="class"><xsl:value-of select="name()"/></xsl:attribute>
      <xsl:call-template name="commonattributes"/>
      <xsl:call-template name="setscale"/>
      <xsl:call-template name="setidaname"/>
     <!--Here I'm calling the styler of the content inside the codeblock.-->
     <xsl:call-template name="outputStyling"/>
    </pre>
    <xsl:apply-templates select="*[contains(@class,' ditaot-d/ditaval-endprop ')]"
          mode="out-of-line"/>
    <xsl:if test="contains(@frame,'bot')"><hr /></xsl:if><xsl:value-of select="$newline"/>
  </xsl:template>
```

You could also use another XSLT template that applies the XSLTHL library as a Java extension to style the content.

4. Create additional XSLT stylesheets to customize all other desired output types. In our example, to customize the PDF output we need to create an XSLT stylesheet called **pdfHighlight.xsl** (in the same *plugin* folder).

   In this case we found an appropriate XSLT stylesheet *DITA-OT-DIR*`/plugins/legacypdf/xslfo/dita2fo-elems.xsl` to use as a template that we use to overwrite our **pdfHighlight.xsl** stylesheet, which results in the following:

```
<xsl:template match="*[contains(@class,' topic/pre ')]">
  <xsl:call-template name="gen-att-label"/>
  <fo:block xsl:use-attribute-sets="pre">
    <!-- setclass -->
    <!-- set id -->
    <xsl:call-template name="setscale"/>
    <xsl:call-template name="setframe"/>
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
```

   Note: You can edit the newly created stylesheets to customize multiple outputs in a variety of ways. For example, in our case you could edit the **xhtmlHighlight.xsl** or **pdfHighlight.xsl** stylesheets that we created to customize various colors for syntax highlighting.

5. To install the created *plugin* in the **DITA OT**, run the predefined transformation scenario called **Run DITA OT Integrator** by executing it from the ***Apply Transformation Scenario(s)*** *dialog box*. If the integrator is not visible, select the **Show all scenarios** option that is available in the **Settings** drop-down menu. For more information, see *Installing a Plugin in the DITA Open Toolkit*.

   Results of running the integrator using our example:

   XSLT content is applied with priority when publishing to both HTML and PDF outputs.

   a. For the HTML output, in the XSLT stylesheet *DITA-OT-DIR*`/xsl/dita2html-base.xsl` a new import automatically appeared:

```
<xsl:import href="../plugins/com.oxygenxml.highlight/xhtmlHighlight.xsl"/>
```

      This import is placed after all base imports and thus has a higher priority. For more information about imported template precedence, see: *http://www.w3.org/TR/xslt#import*.

b. Likewise, for the PDF output, in the top-level stylesheet *DITA-OT-DIR*`/plugins/org.dita.pdf2/xsl/fo/topic2fo_shell_fop.xsl` a new import statement appeared:

```
<xsl:import href="../../../com.oxygenxml.highlight/pdfHighlight.xsl"/>
```

Now, you can distribute your *plugin* folder to anyone that has a DITA OT installation along with some simple installation notes. Your customization will work provided that the templates you are overwriting have not changed from one DITA OT distribution to the other.

Related Information:

DITA Open Toolkit Documentation

Installing a Plugin in the DITA Open Toolkit

The architecture of the **DITA Open Toolkit** allows additional *plugins* to be installed.

1. The additional *plugins* should be copied to the *DITA-OT-DIR*`\plugins` directory.

   Note: If the *plugin* is only supported in DITA-OT 2.4.4 version, make sure that you copy the *plugin* in the *[OXYGEN_INSTALL_DIR]*`/frameworks/dita/DITA-OT2.x` directory.

2. Run the predefined transformation scenario called **Run DITA OT Integrator** by executing it from the ***Apply Transformation Scenario(s)*** *dialog box*. If the integrator is not visible, select the **Show all scenarios** option that is available in the **Settings** drop-down menu.

   Important: The folder where the **DITA OT** is located needs to have full write access permissions set to it. For example if you are integrating *plugins* in the **DITA OT** folder bundled with Oxygen XML Editor and if you are running on **Windows** and your application is installed in the **Program Files** folder, you can start the Oxygen XML Editor main executable with administrative rights for the integrator process to be able to modify resources in the **DITA OT** folder.

   Starting with version 17.0, Oxygen XML Editor detects the transformation type (`transtype`) declarations from **DITA OT** *plugins* and presents descriptions, which are contributed in the `transtype` declarations, in the **DITA Transformation Type** dialog box. Oxygen XML Editor also shows the contributed parameters from **DITA OT** *plugins* in the **Parameters** tab in the **Edit DITA Scenario** dialog box.

3. If the *plugin* contributed a new transformation type that is not detected (for instance, if you are using a previous version of Oxygen XML Editor that does not detect the `transtype` declarations), you can create a new **DITA OT** transformation scenario with a predefined type that is similar to the new transformation type. Then edit the transformation scenario, and in the **Parameters** tab add a `transtype` parameter with the value of the new transformation type.

Note: A transformation type can also extend another `transtype`. For example, the *pdf-prince* `transtype` extends a `commons` transformation type that contains all the common DITA OT parameters.

Example:

```
<plugin id="com.oxygenxml.pdf.prince">
  <!-- extensions -->
  <feature extension="dita.conductor.transtype.check" value="pdf-prince"
 type="txt"/>
  <feature extension="dita.conductor.target.relative" value="integrator.xml"
 type="file"/>
  <feature extension="dita.transtype.print" value="pdf-prince"/>
  <transtype name="pdf-prince"extends="commons" desc="PDF (Prince XML)">
    <param name="princeExecPath" type="file" desc="Path to the Prince executable
 file that will produce the PDF"/>
  </ Transtype>
</plugin>
```

Related Information:

Creating a DITA OT Customization Plugin on page 1712

Installing the DITA For Publishers Package

DITA Open Toolkit Documentation

Use an External DITA Open Toolkit in Oxygen XML Editor

Oxygen XML Editor comes bundled with a DITA Open Toolkit, located in the *DITA-OT-DIR* directory. Starting with Oxygen XML Editor version 17, if you want to use an external DITA OT for all transformations and validations, you can *open the **Preferences** dialog box **(Options > Preferences)*** and go to *the **DITA** page*, where you can specify the DITA OT to be used. Otherwise, to use an external DITA Open Toolkit, follow these steps:

1. Edit your transformation scenarios and in the **Parameters** tab change the value for the *dita.dir* parameter to point to the new directory.
2. To make changes in the libraries that come with the DITA Open Toolkit and are used by the Ant process, go to the **Advanced** tab, click the **Libraries** button and deselect **Allow Oxygen to add high priority libraries to classpath**.
3. If there are also changes in the DTDs and you want to use the new versions for content completion and validation, go to the ***Document Type Association** preferences page*, edit the **DITA** and **DITA Map** document types and modify the catalog entry in the **Catalogs** tab to point to the custom catalog file `catalog-dita.xml`.

Related Information:

Editing a Transformation Scenario on page 925
Creating New Transformation Scenarios on page 886
DITA Open Toolkit Documentation

Third-Party DITA Open Toolkit Plugins

The DITA Open Toolkit 1.8.5 and 2.4.4 distributions that are bundled with Oxygen XML Editor include some pre-installed third-party open-source *plugins* that add extra publishing formats and functionality.

The *plugins* that come bundled with Oxygen XML Editor include:

- ***DITA For Publishers*** (installed only in DITA OT 1.8.5) - These *plugins* allow DITA content to be published to additional formats, such as EPUB 2.0 and Kindle.
- ***DITA to Reveal JS*** (installed only in DITA OT 1.8.5) - This *plugin* allows users to publish DITA content to web slides that can be used for presentations.
- ***DITA to Word*** (installed only in DITA OT 2.4.4) - This *plugin* allows users to publish DITA content to MS Word.
- ***DITA Markdown*** (installed only in DITA OT 2.4.4) - This *plugin* allows users to publish DITA content to *Markdown*, or to publish *DITA maps* that reference *Markdown* resources.
- ***Lightweight DITA*** (installed only in DITA OT 2.4.4) - This *plugin* allows users to create, edit, and validate *Lightweight* DITA topics and *maps*.
- ***DITA Community*** (installed only in DITA OT 2.4.4) - These *plugins* allow support for DITA 1.3 with embedded or referenced MathML and SVG images.

DITA Specialization Support

This section explains how you can integrate and edit a DITA specialization in Oxygen XML Editor.

Integration of a DITA Specialization

A DITA specialization usually includes:

- DTD definitions for new elements as extensions of existing DITA elements.
- Optional specialized processing that is new XSLT template rules that match the extension part of the `class` attribute values of the new elements and thus extend the default processing available in the DITA Open Toolkit.

A specialization can be integrated in the application with minimum effort:

1. If the DITA specialization is available as a DITA Open Toolkit plugin, copy the plugin to the location of the DITA OT you are using (by default *DITA-OT-DIR*`\plugins`). Then run the DITA OT integrator to integrate the plugin. In the ***Transformation Scenarios** view* there is a predefined scenario called **Run DITA OT Integrator** that can be used for this.

Important: The directory where the DITA OT is located needs to have full write access permissions set to it.

2. If the specialization is not available as a DITA OT plugin, you have the following options:
   - If the DTD that define the extension elements are located in a folder outside the DITA Open Toolkit folder, add new rules to the DITA OT catalog file for resolving the DTD references from the DITA files that use the specialized elements to that folder. This allows correct resolution of DTD references to your local DTD files and is needed for both validation and transformation of the *DITA maps* or topics. The DITA OT catalog file is called `catalog-dita.xml` and is located in the root folder of the DITA Open Toolkit.
   - If there is specialized processing provided by XSLT stylesheets that override the default stylesheets from DITA OT, these new stylesheets must be called from the Ant build scripts of DITA OT.

     Important: If you are using DITA specialization elements in your DITA files, it is recommended that you activate the **Enable DTD/XML Schema processing in document type detection** option in the ***Document Type Association*** *preferences page*.
   - In your specialization plugin directory, create a folder called `template_folders`, which would contain all the folders with new file templates. In Oxygen XML Editor, the templates contributed by the plugin will be available in the **New** document wizard.

Related Information:

DITA Configuration and Specialization Tutorials

Transformation Scenarios on page 854

Editing DITA Map Specializations

In addition to recognizing the default *DITA map* formats (`map` and `bookmap`), the ***DITA Maps Manager*** *view* can also be used to open and edit specializations of *DITA maps*.

All advanced edit actions available for the map (such as insertion of topic refs, heads, properties editing) allow you to specify the element in an editable combo box. Moreover the elements that appear initially in the combo are all the elements that are allowed to appear at the insert position for the given specialization.

The topic titles rendered in the ***DITA Maps Manager*** *view* are collected from the target files by matching the `class` attribute and not a specific element name.

When editing DITA specializations of maps in the main editor the insertions of topic reference, topic heading, topic group and conref actions should work without modification. For the table actions, you have to modify each action manually to insert the correct element name at cursor position. You can go to the **DITA Map** document type from the ***Document Type Association*** *preferences page* and edit the table actions to insert the element names as specified in your specialization. See *Adding/Editing a Framework (Document Type)* on page 1176 for more details.

Editing DITA Topic Specializations

In addition to recognizing the default DITA topic formats, topic specializations can also be edited in the **Author** mode.

The content completion should work without additional modifications and you can choose the tags that are allowed at the cursor position.

The CSS styles in which the elements are rendered should also work on the specialized topics without additional modifications.

The toolbar/menu actions should be customized to insert the correct element names. You can go to the `DITA` document type from the ***Document Type Association*** *preferences page* and edit the actions to insert the element names, as specified in your specialization. See *Adding/Editing a Framework (Document Type)* on page 1176 for more details.

Master Files Support in DITA

Oxygen XML Editor includes a feature that allows you to define *Master Files* at project level. This feature is typically used in Oxygen XML Editor for XML documents to determine the context for operations such as

validation, content completion, refactoring, searches, or displaying components collected from various modules. For DITA projects, this feature has a more limited purpose in Oxygen XML Editor since it is mainly used to provide the means for updating references to non-DITA resources.

Since you can move or rename DITA resources (such as topics and maps) in the ***DITA Maps Manager***, the *root map* is used as the scope to update all the references to the moved or renamed resources. However, you do not have this option for non-DITA resources (such as folders, images, html files, audio, video, text files, Markdown documents) since they do not appear in the **DITA Maps Manager**. Therefore, the *Master Files* support in DITA is most helpful when you want to rename or move non-DITA resources since it will update all the references to these resources in the scope of the *Master Files* to achieve the same result as if they were normal DITA resources. It also allows you to move multiple DITA resources (or entire folders) at once in the ***Project*** *view*, instead of the **DITA Maps Manager**, while still giving you the option of updating the references to the moved or renamed resources.

How to Enable Master Files Support in DITA

To use the *Master Files* support in DITA, follow these steps:

1. Go to the ***Project*** *view* and enable *Master Files* support with one of the following methods:
   - Select **Enable Master Files Support** from the **Settings** menu in the top-right corner.
   - Select **Enable Master Files Support** from the contextual menu of the project root folder. If a disabled *Master Files* folder exists, you can also select that option from its contextual menu.
   - Click the **Enable** button in the tooltip located at the bottom. This tooltip window is displayed when the *Master Files* support is disabled. Clicking the **Read more** link takes you to the user guide. Clicking the **Enable** button opens the **Enable Master Files** dialog box. This dialog box contains general information about the **Master Files Support** and allows you to enable it.

     Warning: Once you close this window tip, Oxygen XML Editor hides it for all projects. You can make the window tip reappear by *resetting Oxygen XML Editor to its default settings*. However, doing so will result in you losing your customized options.
2. Add the *main DITA map (root map)* to the Master Files folder by doing one of the following:
   - Right-click the project root folder and select **Detect Master Files**.
   - Right-click the *Master Files* folder and select **Detect Master Files from Project**.
   - If you enabled the *Master Files* support from the tooltip at the bottom of the **Project** view, you can also use the **Detect and Enable** button in the resulting dialog box to detect the *master files* from the current project.
   - Manually add the *root map* to the *Master Files* folder by doing one of the following:
     - Right-click a file from your project and select **Add to Master Files** from the contextual menu (or simply drag and drop it into the *Master Files* folder).
     - Select **Add Files** or **Add Edited File** from the contextual menu of the *Master Files* folder.

Moving or Renaming Non-DITA Resources and Updating the References to Them

With the *Master Files* support enabled, you can move or rename non-DITA resources (such as folders, images, html files, audio, video, text files, Markdown documents) or move multiple normal DITA resources (or entire folders) in the ***Project*** *view* and Oxygen XML Editor will offer the option of updating all the references to the moved or renamed resources in the scope of the *Master Files* (in this case the *main DITA map (root map)*).

To move or rename non-DITA resources (or move multiple DITA resources) and update the references to them, follow these steps:

1. Enable *Master Files* support and add your *root DITA map* to the *Master Files* folder as described in the *How to Enable Master Files Support in DITA* on page 1717 section above.
2. Go to the ***Project*** *view*, and use one of the following methods to move or rename the resources:

   Moving Resources

   To move resources in the ***Project*** *view*, do one of the following:

   - Simply drag and drop the resource to the new location in the tree structure (the **Enable drag-and-drop in Project view** option must be selected in the ***View*** *preferences page*).

- Use the **Cut**, **Copy**, and **Paste** actions from the contextual menu.
- Right-click the resource and select **Refactoring** > **Move resource** action from the contextual menu. Note that this method also allows you to specify a new name and destination path in the **Move resource** dialog box.

Result: In all cases, a **Move resource** dialog box will be presented.

Renaming Resources

To rename resources in the ***Project*** *view*, do one of the following:

- Select the resource and press **F2**, or simply left-click again, until the in-place editor allows you to change the title.
- Right-click the resource and select **Rename** or **Refactoring** > **Rename resource** .

Result: In all cases, a **Rename resource** dialog box will be presented.

3. Make sure the **Update references of the moved resource(s)** option is selected in the resulting **Move** or **Rename** dialog box and keep the scope as **master files** to make sure all the references to the moved or renamed resource are updated.

Metadata

Metadata is a broad concept that describes data that explains or identifies other data. Metadata can be used for many purposes, from driving automation of document builds to enabling authors and readers to find content more easily. DITA provides numerous types of metadata, each of which is used and created differently. Some of the most important forms of metadata in DITA are topic and taxonomy.

Topic Metadata

Topic metadata describes the topic and what it is about. Topic metadata can be inserted in the `prolog` element of a topic or inside the `topicref` element that points to a topic from a map. In other words, metadata about the topic can be asserted by the topic itself, or can be assigned to it by the map that includes it in the build. This allows multiple maps to assign different metadata to the same topic. This may be appropriate when you want to describe a topic differently in various documents.

Taxonomy and Subject Scheme

A taxonomy is a controlled vocabulary. It can be used to standardize how many things in your content and metadata are named. This consistency in naming can help ensure that automated processes work correctly, and that consistent terminology is used in content, and in metadata. In DITA, taxonomies are created using *subject scheme maps*. When you are authoring, many of the values you choose from have been defined in *subject scheme maps*.

Migrating MS Office Documents to DITA

Oxygen XML Editor integrates the entire *DITA for Publishers plugins suite* and includes some helpful tools that allows you to migrate content from Microsoft Office® Word (and other similar types of documents) to DITA. Migration from proprietary formats to XML is rarely perfect and manual changes may need to be made to the converted content, but the methods described below should help you find the best approach for your particular case:

Method 1

1. Open the document in MS Office (or other similar application), select all the content, and copy it.
2. Open Oxygen XML Editor and create a new DITA topic.
3. Paste the selected content in **Author** mode. The *Smart Paste functionality* will attempt to convert the content to DITA.

Method 2

1. Save your document as HTML.
2. Once you have converted it to HTML, you have two possibilities:
   - In Oxygen XML Editor, select **File** > **Import** > **HTML File** to import it as XHTML. Then, open the XHTML in Oxygen XML Editor and use one of the ***XHTML to DITA*** *Transformation Scenarios* to convert the content to DITA.
   - Open the HTML file in any Web browser, select all of its content, and copy it. Then, open Oxygen XML Editor, create a new **DITA** topic and paste the selected content in **Author** mode. The *Smart Paste functionality* will attempt to convert the HTML content to DITA.

Method 3

1. Open the document in the free *Libre Office* application and save it as **DocBook**.
2. Open the **DocBook** document in Oxygen XML Editor.
3. Run the predefined *transformation scenario called* ***DocBook to DITA***.

Method 4

1. If the document is in the MS Word DOCX format, you can open it in the ***Archive Browser*** *view* in Oxygen XML Editor and then open the **document.xml** file contained in the archive.
2. Run the predefined *transformation scenario* called **DOCX DITA**. This scenario runs a build file over the DOCX archive and should produce a DITA project that contains a *DITA map* and multiple topics.
3. You may need to do some manual reconfiguring to map DOCX styles to DITA content.

Related Information:

Smart Paste Mechanism on page 343
Importing Data on page 1132
Working with Archives on page 1063

DITA 1.3 Support

Starting with version 17.1, Oxygen XML Editor includes support for some DITA 1.3 features.

To enable DITA 1.3 support in Oxygen XML Editor and use the DITA Open Toolkit 2.4.4 for publishing, *open the* ***Preferences*** *dialog box* ***(Options*** > ***Preferences)***, go to **DITA**, and select the **Built-in DITA OT 2.x** radio button.

The Oxygen XML Editor publication of the full DITA 1.3 specifications can be found at *https://www.oxygenxml.com/dita/1.3/specs/index.html#introduction/dita-release-overview.html*.

The following table is a list of DITA 1.3 features and their implementation status in Oxygen XML Editor:

Table 53: DITA 1.3 Features Implementation Status

Feature	Editing	Publishing [DITA Open Toolkit 2.4.4 is used]
DITA 1.3 DTD, XML Schema, and Relax NG-based maps/topics/tasks/references, etc.	New DITA 1.3 file templates. By default, DITA topics and maps that do not specify version in the DOCTYPE declaration are also considered to be DITA 1.3 Specific annotations presented in the content completion assistance window and documentation tooltips for all new DITA 1.3 elements	N/A

Feature	Editing	Publishing [DITA Open Toolkit 2.4.4 is used]
Learning Object and Group maps	New file templates	No specific support implemented
Troubleshooting specialization	Create and edit new troubleshooting topics	No specific support implemented
XML markup domain	Validation and Content Completion	Special rendering in PDF and XHTML-based outputs
Equation and MathML domain	Validation and content completion Display and Insert equations	Special rendering in PDF and XHTML-based outputs
SVG domain	Validation and content completion Display referenced SVG content	Special rendering in PDF and XHTML-based outputs
Other new DITA 1.3 elements (*div*, *strike-through*, *overline*, etc)	Validation and Content Completion	Special rendering in PDF and XHTML-based outputs
Release management domain	Validation and Content Completion	No specific support implemented
Scoped keys	Define key scopes Validate and check for completeness Resolve *keyrefs* and *conkeyrefs* taking key scopes into account Key scope information is displayed in a tooltip when hovering over an item in the **DITA Maps Manager**	Partially implemented (Various issues may still be encountered)
Branch filtering	Display, create, and edit *ditavalref* elements	Partially implemented (Various issues may still be encountered)
Key-based cross deliverable addressing	Special display for references to *DITA maps* with **scope="peer"** and a defined **keyscope** Gather and present keys from peer maps	Not implemented.
Shorthand to address syntax that identifies elements in the same topic	Properly resolved for validation, links, and conrefs	Implemented

Feature	Editing	Publishing [DITA Open Toolkit 2.4.4 is used]
Various table attributes (*orientation*, *rotation*, *scope*, and *headers* on cells)	Not implemented in the Table Properties action support. However, attributes can be changed from the Attributes view	Not implemented
New Map *topicref* attributes (*cascade*, *deliveryTarget*)	Allow setting new attributes, propose proper values for them	Implemented

Related Information:

Watch our DITA 1.3 video tutorial for more information about key scopes and branch filtering.

Glossary

20

Topics:

- *Active Cell*
- *Alternate CSS Style*
- *Anchor*
- *Apache Ant*
- *Block Element*
- *Bookmap*
- *Callout*
- *Canonicalize*
- *Content Completion Assistant*
- *Dockable*
- *Document Fragment*
- *Document Type Association*
- *DITA Map*
- *DITA-OT-DIR*
- *Foldable Element*
- *Framework*
- *Global Options*
- *IDML*
- *Inline Element*
- *Java Archive*
- *Key Space*
- *Keystore*
- *Main CSS Style*
- *Master File*
- *Named User*
- *Perspective*
- *Plugin*
- *Pretty-Print*
- *Project Options*
- *QName*
- *Quick Assist*
- *Quick Fix*
- *Root Map*
- *Space-Preserved Element*
- *Subject Scheme Map*
- *Track Changes*
- *Working Set*
- *XML Catalog*

Active Cell

Active cell refers to the selected cell where data is entered when you begin typing. Only one cell is active at a time. The *active cell* is bounded by a heavy border.

Alternate CSS Style

The ***Alternate CSS Style*** refers to the choices in the bottom half of **Styles** drop-down menu (on the toolbar) that makes it easy to apply style changes to your documents as they appear in **Author** mode and the output without having to edit the CSS stylesheets. By default, the *alternate styles* are applied like layers, they are merged sequentially with the *main CSS style*, and you can activate any number of them. However, if you deselect the ***Enable multiple selection of alternate CSSs*** *option* in the **CSS** subtab of the ***Document Type*** *configuration dialog box*, the *alternate styles* are treated like *main CSS styles* and you can only select one at a time.

For more information, see *Configuring and Managing Multiple CSS Styles* on page 1256.

Anchor

An ***Anchor*** is used in various types of links to take the user to a specific location within the target document. It is designated in a URL or in the value of the `href` attribute with a # symbol followed by the anchor that is defined in a target ID (for example `href="MyTopic.dita#anchor`).

Apache Ant

Apache Ant (Another Neat Tool) is a software tool for automating software build processes.

Block Element

A ***block element*** is intended to be visually separated from its siblings, usually vertically. For instance, paragraphs and list items are *block elements*. It is distinct from a *inline element*, which has no such separation.

Bookmap

A ***bookmap*** is a specialized *DITA map* used for creating books. A *bookmap* supports book divisions such as chapters and book lists such as indexes.

Callout

A ***callout*** is a string of text inside a graphic and is connected to a specific location in a document by a line. Oxygen XML Editor uses callouts to present comments and other types of review modifications.

Canonicalize

To ***canonicalize*** something means to convert it to a standard format that everyone generally uses. When using the term with regards to XML, it refers to the process of converting data that has more than one possible representations into a standardization that conforms to the specification of an XML document or document subset. It is helpful for applications that require the ability to test whether or not the content of an XML document or subset has been changed.

Content Completion Assistant

The ***Content Completion Assistant*** refers to a very helpful mechanism in Oxygen XML Editor that offers a list of proposed items that could be inserted at the current location, depending on the current context, editing mode, and type of document. It also tries to determine the most logical choice in the current editing context and displays that proposal at the beginning of the list.

For more information about this feature and how to invoke it, depending on your editing context, see the following:

- *Content Completion Assistant in Author Mode* on page 345
- *Content Completion Assistant in Text Mode* on page 300
- *Content Completion Assistant in Grid Mode* on page 329
- *Content Completion in XSLT Stylesheets* on page 514
- *Content Completion in Ant Build Files* on page 545
- *Content Completion in XML Schema* on page 585
- *Content Completion in XQuery* on page 619
- *Content Completion Assistance in WSDL Documents* on page 632
- *Content Completion in CSS Stylesheets* on page 650
- *Content Completion in LESS Stylesheets* on page 653
- *Content Completion in Relax NG Schemas* on page 659
- *Content Completion in NVDL Schemas* on page 671
- *Content Completion in JavaScript Documents* on page 682
- *Content Completion in Schematron Documents* on page 687
- *Content Completion in SQF* on page 701

Dockable

A ***Dockable*** window is one that can be moved and resized, and either floated or pinned to a location, allowing you to configure the workspace according to your preferences.

Document Fragment

A ***document fragment*** represents a portion of an XML document's tree of nodes or content.

Document Type Association

In general terms, a ***Document Type Association*** is a set of rules that associate a document type with a *framework*. In Oxygen XML Editor, ***Document Type Association*** also specifically refers to a *preferences page* where you can create new custom *frameworks* or edit existing ones. Note that predefined *frameworks* (document types) are read-only, but you can ***Extend*** or ***Duplicate*** them to configure them as custom *frameworks*.

DITA Map

A ***DITA map*** is a component of the DITA *framework* that provides the means for a hierarchical collection of DITA topics that can be processed to form an output. Maps do not contain the content of topics, but only references to them. These are known as topic references. Usually the maps are saved on disk or in a CMS with the extension `.ditamap`.

Maps can also contain relationship tables that establish relationships between the topics contained within the map. Relationship tables are also used to generate links in your published document.

You can use your map or *bookmap* to generate a deliverable using an output type such as XHTML, PDF, HTML Help, or Eclipse Help.

DITA-OT-DIR

DITA_OT_DIR refers to the default directory that is specified for your DITA Open Toolkit distribution in the ***Options*** > ***Preferences*** > ***DITA*** *preferences page*.

For example, if you are using DITA OT 2.4.4, `[OXYGEN_INSTALL_DIR]/frameworks/dita/DITA-OT2.x` (or if you are using DITA OT 1.8.5, the default DITA OT directory is: `[OXYGEN_INSTALL_DIR]/frameworks/dita/DITA-OT`). You can also specify a custom directory.

Foldable Element

A ***foldable element*** refers to elements that can be collapsed and expanded in Oxygen XML Editor. *Foldable elements* are marked with a small triangle (/) on the left side of the editor panel and you can use that triangle to quickly collapse or expand them. This feature is helpful when you are working with large documents and you want to temporarily hide blocks of content. You can right-click the triangle to access additional collapse and expand actions (**Collapse Other Folds**, **Collapse Child Folds**, **Expand Child Folds**, **Expand All**).

Framework

A ***framework*** refers to a package that contains resources and configuration information to provide ready-to-use support for an XML vocabulary or document type. Oxygen XML Editor includes a ***Document Type Configuration Dialog Box*** that allows you to define a set of rules and customize various authoring mechanisms for new or existing *frameworks*.

For advanced details about customizing your own *framework*, see the *Advanced Framework Customization* on page 1169 section.

Global Options

Global Options refers to the *storage option* in the Oxygen XML Editor preference pages (**Options** > **Preferences**). If you select ***Global Options***, the options in that particular preferences page are stored locally on your computer and are not accessible to other users (unless you *export them into an XML options file* that can then be shared).

IDML

IDML is an abbreviation for Adobe InDesign Markup files.

Inline Element

An ***inline element*** is intended to be displayed in the same line of text as its siblings or the surrounding text. For instance, strong and emphasis in HTML are *inline elements*. It is distinct from a *block element*, which is visually separated from its siblings.

Java Archive

Java Archive (JAR) is an archive file format. *JAR* files are built on the ZIP file format and have the `.jar` file extension. Computer users can create or extract *JAR* files using the `jar` command or an archive tool.

Key Space

A ***Key Space*** is established when a *root map* defines a set of effective key bindings. When Oxygen XML Editor processes key references, it determines the effective binding of a given key to a resource in the context of the *specified root map*.

Keystore

A ***Keystore*** is an encrypted file that contains private keys and certificates. There are two types of *keystores* that are supported in Oxygen XML Editor:

- **Java Key Store (JKS)**
- **Public-Key Cryptography Standards version 12 (PKCS-12)**

Main CSS Style

The ***Main CSS Style*** refers to the selection in the top half of the **Styles** drop-down menu (on the toolbar) that makes it easy to quickly change the look of your documents as they appear in **Author** mode and the output without having to edit the CSS stylesheets. The *main CSS* applies to the whole document and you can also select one or more *alternate styles* (listed in the bottom half of the drop-down menu) that behave like layers and are merged sequentially with the *main CSS style*.

For more information, see *Configuring and Managing Multiple CSS Styles* on page 1256.

Master File

A ***Master File*** typically refers to the root of an imported or included tree of modules and this support helps you simplify the configuration and development of XML projects. For more information, see the *Master Files Support* on page 286 section.

Named User

Named User is defined as an individual full or part-time employee who is authorized by *You* (the individual or entity who owns the rights to Oxygen XML Editor) to use the software regardless of whether or not the individual is actively using the software at any given time. To avoid any doubt, *Named User* licenses cannot be shared amongst multiple individuals and separate *Named User* licenses must be purchased for each individual user.

A *Named User* license may not be reassigned to another employee except in the following circumstances:

- (a) Upon termination of the *Named User's* employment with your company.
- (b) Permanent reassignment of a *Named User* to a position that does not involve the use of the Software.

For example, suppose Jane has been assigned an Oxygen XML license and she leaves your company. When she leaves, you can simply reassign her license to John, her replacement. In the event that you do reassign the *Named User* license in accordance with the restrictions above, you do not need to notify Syncro of such a reassignment.

Note: This definition is taken from the Oxygen XML Editor *End User License Agreement*.

Perspective

In Oxygen XML Editor, a ***perspective*** refers to an interface layout geared towards a specific editing environment. Each *perspective* includes a unique set of interface objects, toolbars, views, and features. You can change the *perspective* by selecting the respective icon ( ) in the top-right corner of Oxygen XML Editor or by selecting the *perspective* from the **Window** > **Open Perspective** menu.

The *perspectives* that are available in Oxygen XML Editor are:

- ***Editor*** - The most commonly used *perspective* and it is used to edit XML documents.
- ***XSLT Debugger*** - Used to detect problems in an XSLT transformation by executing the process step by step in a controlled environment.
- ***XQuery Debugger*** - Used to detect problems in an XQuery transformation process by executing the process step by step in a controlled environment
- ***Database*** - Used to browse and manage databases.

Plugin

A ***plugin*** (also referred to as *add-on* or *extension*) is a component that adds specific features to an existing application. Oxygen XML Editor supports *plugins* to enable numerous customizations.

Pretty-Print

Pretty-print refers to formatting and indenting the source code in **Text** mode to make the content easier to view and analyze. The formatting actions that are available in Oxygen XML Editor include:

- **Format and Indent Element** - Available in the **Source** submenu of the contextual menu for the current element.
- **Format and Indent** - Available on the toolbar for the entire currently opened document.
- **Format and Indent Files** - Available in the contextual menu of the ***Project*** *view* for one or more selected files.

Project Options

Project Options refers to the *storage option* in the Oxygen XML Editor preference pages (**Options** > **Preferences**). If you select ***Project Options***, the options in that particular preferences page are stored at project level in the project file (`.xpr`), which can easily be *shared with other users*.

QName

QName stands for "qualified name" and defines a valid identifier for elements and attributes. *QNames* are used as URI references to reference particular elements or attributes within XML documents.

Quick Assist

The ***Quick Assist*** feature gives you easy access to some of the most commonly used actions for the specific type of document you are editing. If one or more actions are available in the current context, they are accessible via a yellow bulb help ( ) placed at the current line in the stripe on the left side of the editor in **Text** mode. You can also invoke the quick assist menu by using the **Alt + 1** (**Meta + Alt + 1** on Mac OS X) keyboard shortcuts.

Quick Fix

The ***Quick Fix*** support in Oxygen XML Editor helps you resolve errors that appear in an XML document by offering proposals to fix problems such as missing required attributes or invalid elements. *Quick Fixes* are available in **Text** mode and **Author** mode and they can be presented and activated in several ways.

- When hovering over an area of text where a validation error or warning occurs, the *Quick Fix* proposals can be presented as links in a tooltip pop-up window.
- When hovering over an error or warning in **Author** mode, the *Quick Fix* proposals are presented in a small drop-down menu.

- If you place the cursor in the highlighted area where a validation error or warning occurs, a *Quick Fix* icon ( ) is displayed in the stripe on the left side of the editor. Clicking that icon will allow you select from the available proposals.
- If you place the cursor in the highlighted area where a validation error or warning occurs, you can also access the *Quick Fix* menu by pressing **Alt + 1 (Command + Alt + 1 on OS X)** on your keyboard.

Oxygen XML Editor also provides support for *defining and customizing a library of Quick Fixes using the Schematron language*.

Root Map

A ***Root Map*** (or master map) specifies a *DITA map* that defines a hierarchical structures of submaps that are contained within the *root map*. Essentially, the *root map* defines a scope and provides the mechanism to allow your defined keys to be propagated throughout the entire map structure (this mechanism is also known as a *key space*).

In Oxygen XML Editor, the **DITA Maps Manager** includes an *option on its toolbar where you can easily specify the root map*, but there are also several other ways to *select or change the root map*.

Space-Preserved Element

A ***spaced-preserved element*** refers to elements that require white spaces and line endings to be preserved (for example, DITA `codeblock` and `pre` elements).

Subject Scheme Map

A ***Subject Scheme Map*** allows you to create custom controlled attribute values and to manage metadata. *Subject scheme maps* use a key definition to define a collection of controlled values rather than a collection of topics. The highest level of map that uses the set of controlled values must reference the *subject scheme map* in which those controlled values are defined.

A controlled value is a keyword that can be used as a value in a metadata attribute. For example, the `@audience` metadata attribute may take a value that identifies the user group associated with a particular content unit (for medical equipment, that might include *therapist*, *oncologist*, *surgeon*, *radiologist*, and so on). In a *subject scheme map*, you can define a list of these audience values and you can then *use these values to profile your content*.

Track Changes

The *Track Changes* feature allows you to review changes that you or other authors have made and then accept or reject them. You can also manage the visualization mode of the tracked changes, add comments to changes, and mark them as being done. These actions are easily accessible from contextual menus, the toolbar, or the ***Review view***.

For more information about this feature, see *Managing Tracked Changes* on page 358.

Working Set

A ***Working Set*** refers to a set of files that will be used for the scope of search and refactoring operations. Many of the search and refactoring wizards include a step where you can specify the scope for the operation and you can choose one or more *working sets* to restrict the scope to that specified set of files.

XML Catalog

An ***XML Catalog*** maps a system ID or a URI reference for a resource (stored either remotely or locally) to a local copy of the same resource. Whenever XML processing relies on external resources (such as referenced schemas and stylesheets), the use of an *XML Catalog* becomes a necessity when Internet access is not available or the connection is slow.

Oxygen XML Editor includes default global catalogs as well as default catalogs for each of the built-in *frameworks*, and you can also create your own. Oxygen XML Editor uses these *XML Catalogs* to resolve references for document validation and transformations. For more information, see *Working with XML Catalogs* on page 484.

Index

Numerics

3-way directory comparison and merge tool *757*, *1464*

A

Action dialog box *62*
Add Edit link in your interface *1389*
Add nodes in Grid Mode *326*
Add Schematron Quick Fix *695*
Add-on preferences *57*
Add-ons
    Check for add-on updates *45*
    Install new add-ons *45*
    Manage add-ons *45*
Adding custom views *1344*
Adding media resources in Author Mode *430*
Additional Framework plugin extension *1338*
Administration page in Web Author
    Configure frameworks *1378*
    Configure license server connection *1376*
    Configure plugin *1377*
    Configure proxy settings *1379*
    Create plugin configuration page *1377*
    Reset Admin credentials *1379*
AI image rendering in Author mode *413*
Annotation preferences *102*
Ant build files
    Component Dependencies view *550*
    Content completion *545*
    Editing in Master Files context *544*
    Highlight occurrences *550*
    Outline view *546*
    Quick Assist feature *552*
    Quick fix support *545*
    Refactoring actions *551*
    Resource Hierarchy/Dependencies view *549*
    Search declarations *550*
    Search references *550*
    Syntax highlighting *546*
    Validation *544*
Ant preferences *134*
Ant transformation scenario
    Options tab *908*
    Output tab *909*
    Parameters tab *909*
Antenna House Formatter processor *828*, *884*, *1695*
Appearance preferences *53*
Apply profiling attributes in Author Mode *377*
Apply profiling condition sets in Author Mode *380*
Archive preferences *148*
Archives
    Browse *1063*
    Edit files *1067*
    EPUB *1065*
    File browser *1063*
    Migrate OOXML to DITA *1067*
    Migrate OOXML to TEI *1067*
    Modify *1063*
    ODF *1065*
    OOXML *1065*
Associate schema directly in XML documents *469*
Associate schema in a framework configuration *471*
Associate schema in a validation scenario defined in framework *467*
Associate schema through a validation scenario *465*
Associate schema to an XML document *464*
Attributes view in Author Mode *224*, *350*
Attributes view in Design mode *244*, *588*
Attributes view in Text mode *199*, *305*
Author editing mode
    Adding media resources *430*
    Apply profiling attributes *377*
    Apply profiling condition sets *380*
    Attributes view *224*, *350*
    Bidirectional text *238*
    Content completion *345*
    Contextual menu actions *437*
    Create/Edit profiling attributes *375*
    Create/Edit profiling condition sets *379*
    Displaying referenced content *234*
    Displaying the markup *234*, *334*
    Drag and Drop *343*
    Editing attributes *341*
    Editing content *334*
    Editing XML markup *336*
    Elements view *228*, *353*
    Entities view *202*, *229*, *307*, *353*
    Find/Replace text *357*
    Folding *342*
    Form Controls *436*
    Generating IDs *433*
    Handling whitespaces *238*
    Image Map Editor
        DITA *415*, *1628*
        DocBook *419*
        TEI *422*
        XHTML *426*
    Image rendering
        AI images *413*
        CGM images *412*
        EPS images *413*
        JAI images *413*
        PDF images *412*
        PSD images *412*
    Inserting images *410*
    MathML *431*
    MathML equations in HTML output *433*
    Model view *201*, *227*, *304*, *348*
    Navigation *211*, *332*
    Outline view *222*, *354*
    Profiling *374*
    Profiling colors and styles *383*
    Profiling/Conditional Text menu *382*
    Refreshing content *433*
    Rendering documents *330*
    Review tools
        Callouts *367*
        Comments *364*
        Highlights *366*
        Review view *371*

Track Changes 358
Schema annotations 347
Selecting content 344
Set schema for content completion 347
Smart Paste 343
Tables
DITA 394, 1633
DocBook 387
Editing features 385
Sorting a table 407
Sorting list items 409
Sorting selected table rows 408
Sorting tables with merged cells 409
TEI 406
XHTML 404
Tags Display Mode 234, 334
Tooltips 235
User roles 329
Using Retina/HiDPI Images 413
Validation errors 236, 448
Views 213
Visual hints 235
Author mode default operations 1184
Author mode preferences 80
Author Stylesheet plugin extension 1338
AutoCorrect
Add dictionaries 734
AutoCorrect preferences 90
Automated tests 1312, 1352
Automatic Spell Check 731
Automatic validation 446
Automatically correct misspelled words 733

B

BaseX database connection 1108
BaseX database contextual menu actions 1109
BaseX XQJ connection 1110
Batch transformation 929
Berkeley DB XML database connection 1085
Berkeley DB XML database contextual menu actions 1087
Berkeley DB XML debugging 1090, 1120
Bidirectional text in Author mode 238
Bidirectional text in Grid mode 210
Bidirectional text in Text mode 207

C

Callout preferences 86
Callouts in Author Mode 367
Canonicalize files tool 738, 1437
Certificates preferences 136
CGM image rendering in Author mode 412
Change file permissions on remote FTP server 264
Change language for interface 177
Changing the font size 334
Changing the font size in Text mode 292
Character Map dialog box 250
Check for add-on updates 45
Check Spelling 725
Check Spelling in Files 732
Check Well-Formedness action 444
Class Loader 1352
Class Loader issues 1350
Clear content in Grid Mode 326
Close file 267
Code template preferences 107
Code templates 308, 354
Color preferences 54
Comments in Author Mode 364
Common problems and solutions 1565
Compare Directories Against a Base tool 757, 1464
Compare Directories tool
Compare images 757, 1464
Menus 756, 1463
Toolbar 755, 1462
Compare Files tool
Menus 750, 1457
Toolbar 748, 1455
Comparing files and directories 741
Compile LESS to CSS 654
Compile XSL Stylesheet for Saxon tool 543, 1426
Components Validation plugin extension 1338
Condition set preferences 87
Configuration logs for Web Author 1392
Configure frameworks 1176
Configure frameworks in Web Author 1378
Configure license server connection in Web Author 1376
Configure plugins in Web Author 1377
Configure proxy settings in Web Author 1379
Configure Toolbars 167
Configure transformation scenario 925
Configure Transformation Scenario dialog box 926
Configuring annotations for the Content Completion Assistant 1250
Configuring content completion proposals 1241, 1243, 1248
Configuring options 161
Configuring Oxygen 50
Content Completion Assistant in Grid Mode 329
Content completion configuration file (cc_config.xml) 1241, 1243, 1248
Content completion helper views 303, 348
Content completion in Author Mode 345
Content completion in Text Mode 300
Content completion preferences 100
Content Management System integration 1120
Contextual menu actions in Author mode 437
Contextual menu actions in Text Mode 318
Contextual menu of current editor tab 267
Convert database to XML Schema 612, 1423
Convert schema to another schema language 610, 1421
Convert XML to JSON 678, 1425
Copy/Paste in Grid Mode 327
Create DTD from learned document structure 472
Create Markdown documents 707
Create new transformation scenario 886
Create new validation scenario 452
Create plugin configuration page in Web Author 1377
Create profiling attributes in Author Mode 375
Create profiling condition sets in Author Mode 379
Create Schematron Quick Fix 695
CSS :has relational pseudo class 1266
CSS @font-face rule 1261
CSS @media rule 1261
CSS attr() function 1271
CSS extensions
Additional CSS properties
-oxy-append-content CSS property 1282
-oxy-collapse-text property value 1281

-oxy-display-tags property 1282
-oxy-editable property 1281
-oxy-foldable property 1279
-oxy-folded property 1279
-oxy-link property 1281
-oxy-lower-cyrillic property values 1281
-oxy-morph property value 1281
-oxy-not-foldable-child property 1279
-oxy-placeholder-content property 1280
-oxy-prepend-content CSS property 1282
-oxy-show-placeholder property 1280
-oxy-style property 1283
-oxy-tags-background-color property 1283
-oxy-tags-color property 1283
-oxy-trim-when-ws-only property value 1281
display property 1281
list-style-type property 1281
visibility property 1281
white-space property 1281
Additional CSS selectors 1277
Built-in CSS selectors 1274
Custom CSS pseudo-classes 1310
Custom form controls 1309
Custom functions
oxy_action 1290
oxy_action_list 1291
oxy_add 1294
oxy_attributes 1287
oxy_base-uri 1284
oxy_capitalize 1285
oxy_concat 1285
oxy_divide 1294
oxy_getSomeText 1288
oxy_indexof 1288
oxy_label 1292
oxy_lastindexof 1288
oxy_link-text 1293
oxy_local-name 1283
oxy_lowercase 1285
oxy_modulo 1294
oxy_multiply 1294
oxy_name 1284
oxy_parent-url 1285
oxy_replace 1286
oxy_substring 1287
oxy_subtract 1294
oxy_unescapeURLValue 1293
oxy_unparsed-entity-uri 1286
oxy_uppercase 1285
oxy_url 1284
oxy_xpath 1289
Form Controls
Audio file player 1294
Browser 1295
Button 1296
Button group 1297
Checkbox 1299
Combo box 1300
Date picker 1301
Editing processing instructions 1310
HTML content 1302
Pop-up 1303
Text area 1304
Text field 1306
URL chooser 1307
Video player 1308
CSS Imports 1259
CSS Inspector view 1311
CSS Inspector View 232
CSS namespace selector 1264
CSS properties 1267
CSS selectors 1262
CSS Styles 1256
CSS stylesheets
Content completion 650
Custom CSS properties 650
Editing features 649
Folding 652
Format and indent (pretty print) 652
Minifying 652
Outline view 651
Syntax highlighting 651
Validation 650
CSS subject selector 1266
CSS validation preferences 113
Cursor navigation preferences 83
Custom dictionaries in Web Author 1390
Custom editor variables 156
Custom form controls 1309
Custom Protocol plugin extension 1339
Custom system properties 174
Custom validation engine preferences 111
Custom XML refactoring operations 497, 1406
Custom XSLT/XQuery transformation engine preferences 133
Customize document templates 257
Customizing default options 161
Customizing frameworks/document types 1169
Customizing options for Web Author 1381

D

Data Source Explorer view 1068
Data Sources preferences page 138
Database connection preferences 138
Database drivers 141
Database perspective 184
Databases
Connections
BaseX
Contextual menu actions 1109
XQJ 1110
Berkeley DB XML
Contextual menu actions 1087
Debugging 1090, 1120
Documentum xDB
Contextual menu actions 1101
Parser configuration 1104
eXist
Contextual menu actions 1092
Generic JDBC 1106
IBM DB2
Contextual menu actions 1075
JDBC-ODBC 1107
MarkLogic
Contextual menu actions 1099
Debugging 1097, 1119
MarkLogic development 1096
Microsoft SQL Server
Contextual menu actions 1078
MySQL 1105

Oracle
Contextual menu actions *1081*
PostgreSQL
Contextual menu actions *1085*
SharePoint
Contextual menu actions *1129*
SharePoint Browser view *1127*
WebDAV
Contextual menu actions *1112*
Data Source Explorer view *1068*
SQL support *1113*
Table Explorer view *1069*
XQuery
Debugging
Berkeley DB XML *1090*, *1120*
MarkLogic *1097*, *1119*
Drag and drop from Data Source Explorer *1116*
Transformations *1116*
Validation *1116*
Debug PDF transformation *932*
Debugger preferences *128*
Debugging CSS stylesheets *1311*
Debugging XQuery
Breakpoints *1156*
Identify expressions *1157*
Information views
Breakpoints view *1148*
Context Node view *1146*
Messages view *1149*
Nodes/Values Set view *1153*
Output Mapping Stack view *1150*
Stack view *1150*
Templates view *1153*
Trace History view *1151*
Variables view *1154*
XPath Watch view *1147*
Java extensions *1159*
Layout *1143*
Profiling
Hotspots view *1162*
Invocation Tree view *1161*
Steps in a typical debugging process *1156*
Toolbar *1144*
Debugging XSLT
Breakpoints *1156*
Identify expressions *1157*
Information views
Breakpoints view *1148*
Context Node view *1146*
Messages view *1149*
Nodes/Values Set view *1153*
Output Mapping Stack view *1150*
Stack view *1150*
Templates view *1153*
Trace History view *1151*
Variables view *1154*
XPath Watch view *1147*
Java extensions *1159*
Layout *1143*
Profiling
Hotspots view *1162*
Invocation Tree view *1161*
Steps in a typical debugging process *1156*
Toolbar *1144*
Deploying Web Author *1372*
Design editing mode
Attributes view *244*, *588*
Components
Grouping components *580*
xs:all *575*
xs:alternative *571*
xs:any *576*
xs:anyAttribute *576*
xs:assert *579*
xs:attribute *565*
xs:attributeGroup *567*
xs:choice *575*
xs:complexType *568*
xs:element *562*
xs:field *579*
xs:group *572*
xs:import *573*
xs:include *573*
xs:key *578*
xs:keyRef *578*
xs:notation *574*
xs:openContent *580*
xs:override *574*
xs:redefine *573*
xs:schema *561*
xs:selector *578*
xs:sequence *575*
xs:simpleType *570*
xs:unique *577*
Contextual menu actions *555*
Editing actions *554*
Facets view *246*, *590*
Navigation *241*, *554*
Outline view *242*, *586*
Palette view *247*, *589*
Design mode preferences *92*
Detect Master Files *287*
Detect Master Files from Project *287*
Diff Directories tool *753*, *1460*
Diff Files tool *742*, *1449*
Digital Signature
Canonicalize files *738*, *1437*
Certificates *738*
Sign files *739*, *1438*
Verify signature *740*, *1440*
Digital Signatures
Example of how to digitally sign XML content *741*
Overview *736*
Directory comparison appearance preferences *148*
Directory comparison preferences *147*
Display problem on Linux/Solaris *1564*
Displaying markup in Author mode *234*, *334*
Displaying referenced content in Author mode *234*
DITA
Conkeyref *1657*
Conref *1656*
Conref Push *1665*
Content Key References *1657*
Content References *1656*
Cross references *1674*
DITA 1.3 support *1719*
Filtering content *1698*
Image Map Editor *415*, *1628*
Index creation *1608*
Keys *1652*

Linking
Cross reference 1674
File reference 1674
Hierarchical 1674
Related links 1674
Relationship tables 1678
Web link 1674
Maps
Add topics 1592
Change order of topics 1592
Chunking 1609
Create bookmap 1591
Create map 1590
Create subject scheme 1590
Create submap 1591
Create table of contents 1608
Define keys 1602
DITA Maps Manager 1582
Edit Properties dialog box 1603
Find unreferenced resources 1595
Insert Reference dialog box 1596
Insert references 1595
Insert topic groups 1601
Insert topic headings 1600
Managing maps 1592
Move resources 1594
Remove topics 1592
Rename resources 1594
Root map 1591
Validate and Check for Completeness
Run from Command-Line 1613
XML catalogs 1609
Master Files support 1716
Metadata 1718
Migrate Office documents to DITA 1718
Open Toolkit
Create customization plugin 1712
Install additional plugins 1714
Third party plugins 1715
Use external DITA OT 1715
Output
Customizing Transformations
Custom build file 1689
DITA Map to MS Office Word transformation 830
DITA Map to PDF WYSIWYG transformation 828, 884, 1695
DITA OT Transformation Scenario 1680
PDF output customization
Add watermark 825, 880, 1692
Creating a customization directory 823, 878, 1690
Customize 'Note' images 827, 882, 1694
Customize header/footer 825, 880, 1692
Force page breaks 826, 882, 1694
Set font 827, 883, 1695
Show comments and tracked changes 827, 883, 1695
XHTML output customization 1697
PDF output customization
Add watermark 825, 880, 1692
Creating a customization directory 823, 878, 1690
Customize 'Note' images 827, 882, 1694
Customize header/footer 825, 880, 1692
Force page breaks 826, 882, 1694
Set font 827, 883, 1695
Show comments and tracked changes 827, 883, 1695
Profiling
Applying profiling attributes 1700
Applying profiling condition sets 1703
Creating or editing profiling attributes 1698
Creating or editing profiling condition sets 1701
Customizing colors and styles for profiling attributes 1707
DITAVAL filter file 1709
Filtering attribute values 1709
Flagging content with DITAVAL file 1711
Publishing profiling 1711
Showing and filtering profiling attributes 1705
Using subject scheme map 1708
Publishing 1679
Relationship tables 1678
Reusing content
Branch Filtering 1670
Content Key References 1657
Content Reference Push mechanism 1665
Content References 1656
DITA Reusable Components view 1671
Edit Content References 1658
Key Scopes 1669
Reference Topics in multiple maps 1654
Reusable components
Create Reusable Component 1667
Insert Reusable Component 1668
Reuse Content dialog box 1659
Variable text 1668
Specialization
Integration 1715
Maps 1716
Topics 1716
Topics
Add images 1625
Add media resources 1626
Content completion 1622
Convert topic types 1624
Create new topic 1618
Edit topics 1622
Embed HTML Content 431
Fast Create multiple topics 1620
Image maps 415, 1628
MathML equations 1642
Tables 394, 1633
DITA and Markdown documents 714, 1651
DITA Authoring and Publishing 1580
DITA Map document type
Author mode actions 805, 1614
DITA Map menu actions 805, 1614
DITA Map to CHM (Compiled Help) transformation 832, 886
DITA Map to Kindle transformation 832, 886
DITA Map to MS Office Word transformation 830
DITA Map to PDF WYSIWYG transformation 828, 884, 1695
DITA Map to WebHelp Classic Mobile transformation 820, 876
DITA Map to WebHelp Classic transformation 815, 871
DITA Map to WebHelp Responsive transformation 808, 864
DITA Map to WebHelp Responsive with Classic transformation 818, 873
DITA Map to WebHelp Responsive with Feedback transformation 812, 867
DITA Map toolbar actions 805, 1614
DITA Map transformations 808, 863
DITA Maps
Transformation scenarios 808, 863

DITA OT preferences 136
DITA OT transformation scenario
Advanced tab 904, 1686
Filters tab 903, 1685
FO Processor tab 901, 1684
Output tab 905, 1688
Parameters tab 902, 1685
Skins tab 899, 1682
Templates tab 900, 1683
DITA OT Transformation Scenario 1680
DITA preferences 136
DITA tables in Author Mode 394, 1633
DITA Topic document type
Author mode actions 795, 1643
DITA Topic menu actions 795, 1643
DITA Topic toolbar actions 795, 1643
DocBook
Output
PDF output customization
Show comments and tracked changes 788, 862
PDF output customization
Show comments and tracked changes 788, 862
DocBook 4 document type
Author mode actions 765
Transformation scenarios 771, 855
DocBook 4 menu actions 765
DocBook 4 to WebHelp transformation 771, 855
DocBook 4 toolbar actions 765
DocBook 4 transformations 771, 855
DocBook 5 document type
Author mode actions 779
Transformation scenarios 785, 859
DocBook 5 menu actions 779
DocBook 5 to WebHelp transformation 785, 859
DocBook 5 toolbar actions 779
DocBook 5 transformations 785, 859
DocBook 5.1 Assembly 792
DocBook 5.1 document type 778
DocBook 5.1 Topic document type 793
DocBook olink 775, 790
DocBook tables in Author Mode 387
DocBook Targetset document type 794
DocBook to DITA transformation 775, 789, 858, 863
DocBook to EPUB transformation 774, 789, 859, 863
DocBook to PDF transformation 774, 788, 858, 862
Document checking preferences 95
Document plugin extension 1347
Document template preferences 111
Document templates
Creating 256
Customizing 257
Document type association preferences 57
Document type configuration dialog box
Association rules tab 59
Author tab
Actions subtab 62
Content Completion subtab 71
Contextual Menu subtab 69
CSS subtab 61
Menu subtab 69
Toolbar subtab 71
Catalogs tab 73
Classpath tab 61
Extensions tab 75
Schema tab 60
Templates tab 72
Transformation tab 73
Validation tab 74
Document type customization 1169
Document Types 763
Documentum xDB database connection 1100
Documentum xDB database contextual menu actions 1101
Documentum xDB database parser configuration 1104
Documentum xDB troubleshooting 1104
DOM element 1331
Drag and Drop in Author Mode 343
Drag and Drop in Grid Mode 326
Drag and Drop in Text Mode 298
DTD Entities for editing large XML documents 486
Duplicate nodes in Grid Mode 326
Duplicate transformation scenario 926

E

Edit
create new documents 252
Edit cell value in Grid Mode 326
Edit documents with long lines 736
Edit mode preferences 77
Edit profiling attributes in Author Mode 375
Edit profiling condition sets in Author Mode 379
Edit validation scenario 456
Editing attributes in Author Mode 341
Editing content in Author Mode 334
Editing Modes
Author 211
Design 240, 553
Grid 208
Text 186
Editing XML markup in Author Mode 336
Editing XML markup in Text Mode 294
Editing XSLT
Compile XSL Stylesheet for Saxon tool 543, 1426
Editor perspective 181
Editor preferences 76
Editor variables
Custom editor variables 174
Elements view in Author Mode 228, 353
Elements view in Text mode 202, 306
Elements view preferences 159
Encoding preferences 75
Entities view 202, 229, 307, 353
EPS image rendering in Author mode 413
EPUB document type 852
Executing SQF in Other Documents 700
eXist database connection 1090
eXist database contextual menu actions 1092
Export color themes 53
Export Global Options 164
Export Global Transformation Scenarios 169
Export Global Validation Scenarios 169
Export Layout 165
Export Markdown documents 707
Extending Oxygen with plugins 1336
External tool configuration 150
External tool preferences 150
External Tools 1562

F

Facets view in Design mode *246*, *590*
File comparison appearance preferences *147*
File comparison preferences *145*
File extension association *164*
File permissions in Web Author *1390*
File properties *268*
File related actions in Text Mode *311*
File type preferences *153*
Filtering profiled content in Author mode *382*
Find All Elements action *474*
Find All Elements dialog box *474*
Find/Replace action *472*
Find/Replace dialog box *472*
Find/Replace in Files action *476*
Find/Replace in multiple files *476*
Find/Replace text *472*
Find/Replace text in Author Mode *357*
Flatten Schema tool *613*, *1423*
FO processor preferences *129*
Folding elements in Author Mode *342*
Folding in Text Mode *298*
Font preferences *55*
Font size configuration *334*
Font size configuration in Text mode *292*
Form Controls
- Audio file player *1294*
- Browser *1295*
- Button *1296*
- Button group *1297*
- Checkbox *1299*
- Combo box *1300*
- Date picker *1301*
- Editing processing instructions *1310*
- HTML content *1302*
- Pop-up *1303*
- Render HTML frames *1295*
- Render SVG *1295*
- Text area *1304*
- Text field *1306*
- URL chooser *1307*
- Video player *1308*

Form Controls in Author mode *436*
Format and Indent Files tool *316*, *1427*
Format and Indent in Text Mode *312*
Format/Indent multiple files *316*, *1427*
Formatting preferences *96*
Formatting Schematron Quick Fix Content *700*
Framework customization
- Adding/editing *1176*
- Configuring an extensions bundle *1212*
- Configuring annotations for the Content Completion Assistant *1250*
- Configuring content completion proposals *1241*, *1243*, *1248*
- Configuring document templates *1204*
- Configuring transformation scenarios *1206*
- Configuring validation scenarios *1207*
- Configuring XML catalogs *1205*
- Creating a basic document type association *1169*
- Customize the main CSS *1237*
- Customizing Smart Paste mapping *1211*
- Customizing the Content Completion Assistant *1241*
- Deploying as Add-ons *1204*, *1351*
- Extensions
  - Author Action Event Handler *1218*
  - Author Edit Properties Handler *1216*
  - Author Extension State Listener *1220*
  - Author Image Decorator *1219*
  - Author Persistent Highlighter *1237*
  - Author Reference Resolver *1226*
  - Author Schema Aware Editing Handler *1215*
  - Author Table Cell Separator Provider *1235*
  - Author Table Cell Span Provider *1233*
  - Author Table Column Width Provider *1229*
  - Content Completion Handler *1221*
  - CSS Styles Filter *1228*
  - Custom Attribute Value Editor *1217*
  - Custom Drag and Drop Listener *1226*
  - Element Locator Provider *1222*
  - Link Target Element Finder *1222*
  - Profiling Conditional Text Provider *1215*
  - Unique Attributes Recognizer *1236*
  - XML Node Renderer Customizer *1237*
- Localizing *67*, *1203*
- Sharing *1239*
- Sharing extended framework *1240*

Framework directory locations *58*
Frameworks *763*
FTP connection settings *158*

G

General plugin extension *1346*
Generate Documentation tools *1428*
Generate IDs in Author mode *433*
Generate Sample XML Files tool
- Advanced tab *601*, *1419*
- From command line *601*, *1420*
- Options tab *599*, *1417*
- Schema tab *597*, *1416*

Generate WSDL Documentation tool *643*, *1435*
Generate XML Schema Documentation tool
- Customize PDF output *608*
- From command line *609*
- Output formats *605*

Generate XSLT Stylesheet Documentation tool
- Custom format *541*
- From command line *542*
- HTML format *539*

Generic JDBC database connection *1106*
GET parameters for WebHelp Classic *1020*, *1043*
GET parameters for WebHelp Responsive *986*
Getting Started with Oxygen
- Help *15*
- Layout *3*
- Resources to help you with Oxygen *4*
- Shortcut keys *18*
- Supported document types *4*
- Your first DITA document *9*
- Your first XML document *6*

Global Options *162*
Global preferences *52*
Grid editing mode
- Add nodes *326*
- Bidirectional text *210*
- Clear content *326*
- Content Completion Assistant *329*
- Copy/Paste *327*

Drag and Drop 326
Duplicate nodes 326
Edit cell value 326
Insert columns 326
Insert rows 326
Refresh layout 326
Sort columns 326
Grid editing modes
Layout 208
Navigation 209
Grid mode preferences 79

H

Help
Randomize XML text content 17
Support related resources 15
Using the Help menu 16
Hex Viewer tool 1447
Highlight ID occurrences in Text Mode 317
Highlights in Author Mode 366
Highlights in XML documents 489
HTTP authentication schemes 266
HTTP connection settings 157
HTTPS troubleshooting 264

I

IBM DB2 database connection 1073
IBM DB2 database contextual menu actions 1075
IIS reverse proxy with Web Author 1395
Image Map Editor in Author Mode 414
Image Map Editor in DITA 415, 1628
Image Map Editor in DocBook 419
Image Map Editor in TEI 422
Image Map Editor in XHTML 426
Image rendering in Author mode 411
Import color themes 53
Import data dynamically 1139
Import Global Options 164
Import Global Transformation Scenarios 169
Import Global Validation Scenarios 169
Import preferences 134
Importing data 1132
Importing data from a database 1136
Importing data from Excel 1134
Importing data from HTML files 1138
Importing Data from text files 1132
Incorrect Function error 43
Increase stack size 1565
Indenting Schematron Quick Fix Content 700
Information view 488
Information view preferences 159
Insert columns in Grid Mode 326
Insert File in Text mode 311
Insert images in Author Mode 410
Insert rows in Grid Mode 326
Install new add-ons 45
Installing Oxygen
Command line options 47
Floating license server 36
Group deployment 32
HTTP floating license server
Management and Statistics 39
Replacing 40
Upgrading 41
Installation options 23
Java Web Start Installer 30
Licensing 32
Linux Installation 26
Linux/Unix Server Installation 29
Mac OS X Installation 25
TCP floating license server
All Platforms distribution
Replacing 43
Upgrading 44
Windows 32-bit
Incorrect Function error 43
Process Terminated Unexpectedly error 43
Replacing 42
Upgrading 42
Transfer license key 44
Upgrading 45
Windows Installation 24
Windows Server Installation 28
Integrate Schematron Quick Fixes in a framework 703
Integrating external tools 1562
Introduction 1

J

JAI images in Author mode 413
JATS document type
Author mode actions 849
JATS menu actions 849
JATS toolbar actions 849
Java system properties 174
Java VM parameters 178
JavaScript documents
Content Completion 682
Editing features 680
Outline view 683
Syntax highlighting 682
Validation 681
JDBC-ODBC database connection 1107
Jenkins integration with WebHelp plugin 1052
JSON documents
Folding 677
Grid mode editing features 676
Outline view 677
Syntax highlighting 677, 708, 713
Text mode editing features 674
Validation 677
XML to JSON converter 678, 1425
jTEI document type 848

K

Kerberos 266

L

Large documents 734
Large File Viewer tool 1445
Layout configuration 165
Layout preferences 56
LESS stylesheets
Compile to CSS 654

Content completion 653
Editing features 653
Syntax highlighting 654
Validation 653
Licensing Oxygen
Floating license
HTTP license server 35
Multiple users 36
Release floating license 35
TCP license server 34
Named-user license 33
Licensing Web Author 1373
Load Layout 165
Load-balanced server setup for Web Author 1380
Localization file 177
Localizing frameworks 67, 1203
Localizing the user interface 177
Localizing XML refactoring operations 509, 1415
Lock Handler plugin extension 1340
Lock/Unlock XML tags in Text Mode 312

M

Manage add-ons 45
Manage highlighted content in Text Mode 316
Manual validation actions 446
Markdown documents
Actions in Markdown Editor 709
DITA 714, 1651
Markdown Editor 707
Rules and specifications 715
Validation 714
Markdown Editor 707
MarkLogic database connection 1094
MarkLogic database contextual menu actions 1099
MarkLogic debugging 1097, 1119
MarkLogic for the developer 1096
Markup transparency in Text Mode 311
Master Files
Add files 288
Contextual menu 288
Detecting 287
Enabling 287
Overview 286
Transformations 288
Validation 288
MathML equations in HTML output in Author mode 433
MathML notations in Author mode 431
MathML preferences 90
Menu Shortcut Keys 152
Message preferences 160
Microsoft SQL Server database connection 1076
Microsoft SQL Server database contextual menu actions 1078
Model view 201, 227, 304, 348
Modular XML files 481
Move file tabs 266
Move XML resources 483
MSXML 3.0 and 4.0 preferences 124
MSXML.NET preferences 124
MySQL database connection 1105

N

Navigating in Text Mode 186, 290
Navigation in Author Mode 211, 332
Network connection preferences 156
New document from templates 256
New Document Wizard 252
New from Templates Wizard 256
New project 14, 275
NISO Journal Article Tag Suite document type 849
Non-XML files 724
NTLM 266
NVDL schemas
Component Dependencies view 672
Content completion 671
Diagram Editor
Contextual menu actions 671
Full Model view 669
Introduction 669
Logical Model view 670
Outline view 672
Refactoring actions 673
Search actions 673
Syntax highlighting 672
Validation 671

O

Olink element 775, 790
Open file 259
Open File at Cursor in Text mode 311
Open file at specific location 260
Open file at start-up 259
Open file in system application 259
Open preferences 105
Open Redirect plugin extension 1340
Open remote document 261
Open URL dialog box 261
Open/Find Resource
In content 273
In file paths 274
In reviews 275
Open/Find resource preferences 154
Option Page plugin extension 1340
Options Menu
Preferences 50
Oracle database connection 1079
Oracle database contextual menu actions 1081
Out Of Memory error 129, 1445, 1564, 1565
Outline view in Author Mode 222, 354
Outline view in Design mode 242, 586
Outline view in Text mode 197, 308
Outline view preferences 159
oxy:allows-child-element function 65
oxy:allows-global-element function 66
oxy:current-selected-element function 67
oxy:is-required-element function 67
Oxygen media type 1260
Oxygen SDK
Automated tests 1312, 1352
CMS integration 1348
Configuration 1336
Custom Protocol 1351
Debugging a plugin 1353
Debugging an SDK extension 1354
Deploying plugins as add-ons 1204, 1351
Disable a plugin 1354
Installation 1337

Types of Extensions 1338
Oxygen XML Author Component
Customization
Customize frameworks 1357
Customize options 1358
Deployment 1358
Frequently Asked Questions 1368
Installation requirements 1356
Licensing 1355
SharePoint integration 1363
Web deployment
Applet troubleshooting 1360
Insert references from a WebDAV connection 1362
MathML support 1361
Signing an Applet 1359
Supported browsers and operating systems 1360
Using plugins 1363
Oxygen XML Web Author Component
Add Edit link in your interface 1389
Administration page 1374
Configuration logs 1392
Customize client side 1388
Customize frameworks 1383
Customize options 1381
Customize plugins 1386
Deploying 1372
Embed in a CMS page 1391
Enable file browsing for a custom protocol handler 1392
File permissions 1390
Implement CMS authentication mechanism 1392
Licensing 1373
Load-balanced server setup 1380
Open a file as read-only 1393
Register create or open document actions 1393
Reverse proxy 1395
Set up a development environment 1393
Share Tomcat instance 1394
Spell Checker custom dictionaries 1390
Troubleshooting errors 1394
Upgrading 1374

P

PAC 157
Palette view in Design mode 247, 589
PDF image rendering in Author mode 412
Performance preferences 105
Perspectives
Database 184
Editor 181
XQuery Debugger 183
XSLT Debugger 182
Plugin extensions
Additional Framework 1338
Author Stylesheet 1338
Components Validation 1338
Custom Protocol 1339
Document 1347
General 1346
Lock Handler 1340
Open Redirect 1340
Option Page 1340
Refactoring Operations 1345
Resource Locking 1341
Selection 1346
Styles Filter 1341
URL Stream Handler 1341
Workspace Access
Adding custom views 1344
Workspace Access (JS-based) 1344
Plugin preferences 149
Plugins
Automated tests 1312, 1352
CMS integration 1348
Configuration 1336
Custom Protocol 1351
Debugging 1353
Debugging an SDK extension 1354
Deploying plugins as add-ons 1204, 1351
Disable a plugin 1354
Installation 1337
Types of Extensions 1338
PostgreSQL database connection 1083
PostgreSQL database contextual menu actions 1085
Predefined frameworks
DITA Map 804
DITA Topic 794
DocBook 4 764
DocBook 5 778
DocBook 5.1 778
DocBook 5.1 Assembly 792
DocBook 5.1 Topic 793
DocBook Targetset 794
EPUB 852
JATS 849
jTEI 848
TEI ODD 843
TEI P5 837
XHTML 833
Predefined XML refactoring operations 493, 1402
Preferences
Add-ons 57
Appearance
Colors 54
Fonts 55
Application Layout 56
Archive 148
CSS Validator 113
Custom Editor Variables 156
Data Sources
Table Filters 141
Diff
Directories Comparison
Appearance 148
Files Comparison
Appearance 147
DITA 136
Document Type Association
Document type configuration dialog box
Association rules tab 59
Author tab
Actions subtab 62
Content Completion subtab 71
Contextual Menu subtab 69
CSS subtab 61
Menu subtab 69
Toolbar subtab 71
Catalogs tab 73
Classpath tab 61
Extensions tab 75

Schema 60
Templates tab 72
Transformation tab 73
Validation tab 74
Locations 58
Editor
Code Templates 107
Content Completion
Annotations 102
JavaScript 104
XPath 103
XSD 103
XSL 103
Custom Validation Engines 111
Document Checking 95
Document Templates 111
Edit Modes
Author
AutoCorrect
AutoCorrect Dictionaries 91
Cursor Navigation 83
MathML 90
Profiling/Conditional Text
Attributes Rendering 89
Colors and Styles 88
Review
Callouts 86
Schema Aware 83
Schema Design
Properties 93
Grid 79
Text
Diagram 79
Format
CSS 100
JavaScript 100
XML
Whitespaces 99
XPath 100
XQuery 99
Mark Occurrences 111
Open/Save
Save Hooks 106
Print 77
Spell Check
Spell Check Dictionaries 94
Syntax Highlight
Elements/Attributes by Prefix 104
Templates 107
Encoding 75
External Tools 150
File Types 153
Global 52
Menu Shortcut Keys 152
Messages 160
Network Connection Settings
(S)FTP 158
HTTP(S)/WebDAV 157
Proxy 156
SSH 159
Trusted Hosts 159
Open/Find Resource 154
Plugins 149
SVN
Diff 144
Messages 144
Working Copy 143
Views 159
XML
Ant 134
Import 134
Sample XML Files Generator 117
XML Catalog 113
XML Parser
Relax NG 116
Schematron 116
XML Schema 115
XML Refactoring 136
XML Signing Certificates 136
XProc 119
XSLT-FO-XQuery
Custom Engines 133
Debugger 128
FO Processor 129
Profiler 128
XPath 132
XQuery
Saxon-HE/PE/EE
Advanced 127
XSLT
MSXML 124
MSXML.NET 124
Saxon-HE/PE/EE
Advanced 122
Saxon6 120
XSLTProc 123
XML Structure Outline 159
Prince Print with CSS processor 828, 884, 1695
Print documents 490
Print preferences 77
Print Preview dialog box 490
Problems and solutions 1565
Process Terminated Unexpectedly error 43
Profiling attribute preferences 87
Profiling colors and styles in Author Mode 383
Profiling content in Author Mode 374
Profiling/Conditional Text menu in Author mode 382
Project Options 162
Project view 188, 213, 276
Project view preferences 159
Projects
Adding items to projects 14, 275
Creating new projects 14, 275
Image preview 284
Managing resources 188, 213, 276
Master files
Add files 288
Contextual menu 288
Detecting 287
Enabling 287
Overview 286
Transformations 288
Validation 288
Move resources 283
Project view 188, 213, 276
Rename resources 283
Sharing 284
Proxy auto-config script 157
Proxy preferences 156
PSD image rendering in Author mode 412

Publish WebHelp on SharePoint *988*, *1023*, *1045*
Publishing *854*

Q

Quick Assist feature in Text Mode *317*
Quick Find toolbar *475*
Quick fix support in XML documents *462*

R

Read-only files *735*
Rectangular Selection feature *299*
Refactoring Operations plugin extension *1345*
Refactoring XML Documents *491*, *1399*
Refresh layout in Grid Mode *326*
Refreshing content in Author mode *433*
Regular expressions syntax *475*, *479*
Relax NG schemas
    Component Dependencies View *665*
    Content completion *659*
    Custom datatype library *668*
    Diagram Editor
        Contextual menu actions *658*
        Full Model view *655*
        Introduction *655*
        Logical Model view *656*
        Symbols *657*
    Editing in Master Files context *655*
    Moving resources *665*
    Outline view *661*
    Quick Assist feature *667*
    Refactoring actions *666*
    Renaming resources *665*
    Resource Hierarchy/Dependencies view *663*
    Search actions *666*
    Syntax highlighting *660*
    Validation *659*
Rename XML resources *483*
Rendering documents in Author Mode *330*
Reset Admin credentials for Web Author *1379*
Reset Global Options *164*
Reset Layout *165*
Resolve schemas through XML catalogs *471*, *485*
Resource Hierarchy/Dependencies view for XML documents *482*
Resource Locking plugin extension *1341*
Restricting Schematron Quick Fix operations *699*
Results view
    Make persistent copy *488*
Retina/HiDPI Images in Author mode *413*
Reverse proxy with Web Author *1395*
Review preferences *85*
Review tools in Author Mode *357*
Review view in Author Mode *371*

S

Sample XML File Generator preferences *117*
Save file *260*
Save preferences *105*
Save remote document *261*
Saxon 6 XSLT preferences *120*
Saxon HE/PE/EE advanced XQuery preferences *127*
Saxon HE/PE/EE advanced XSLT preferences *122*
Saxon HE/PE/EE XQuery preferences *126*
Saxon HE/PE/EE XSLT preferences *121*
Schema annotations in Author Mode *347*
Schema annotations in Text Mode *302*
Schema Design mode *240*, *553*
Schema Diagram Editor *240*, *553*
Schematron Quick Fix Operations
    Add *697*
    Delete *697*
    Replace *697*
    String Replace *697*
Schematron Quick Fixes
    Content Completion *701*
    Customizing *695*
    Defining *695*
    Embedded *702*
    Executing SQF in Other Documents *700*
    Formatting and Indenting *700*
    Highlight occurrences *701*
    Integrating in a framework *703*
    Refactoring actions *701*
    Restricting operations *699*
    Search actions *701*
    SQF Operations *697*
    Use-When Condition *699*
    User Entry operation *699*
    Validation *700*
Schematron schemas
    Content completion *687*
    Syntax highlighting *688*
Schematron Schemas
    Editing features *685*
    Editing in Master Files context *686*
    Embedded *688*
    Highlight occurrences *692*
    Moving resources *692*
    Outline view *689*
    Quick Assist feature *694*
    Refactoring actions *692*
    Renaming resources *692*
    Resource Hierarchy/Dependencies view *690*
    Search actions *692*
    Validation *686*
Scratch Buffer *735*
Search *357*
Search actions for IDs *479*
Search dialog box *472*
Search in current file *472*
Search multiple files *476*
Search preferences *154*
Searching documents
    Open/Find Resource
        In content *273*
        In file paths *274*
        In reviews *275*
Selecting content in Author Mode *344*
Selecting content in Text Mode *299*
Selection plugin extension *1346*
Set indent to zero in Text Mode *315*
Set schema for content completion in Author Mode *347*
Set schema for content completion in Text Mode *302*
Setting a system property for Web Author *1382*
SFTP connection settings *158*

Share Tomcat instance between Web Author and other apps *1394*
Share transformation scenarios *929*
Share validation scenarios *460*
SharePoint Browser view *1127*
SharePoint contextual menu actions *1129*
SharePoint database connection *1126*
Sharing extended framework *1240*
Sharing frameworks *1239*
Sharing project level options *163*
Sharing project options file
    Minimize differences between versions *286*
Sharing settings *163*
Sharing Transformation Scenarios *284*
Sharing Validation Scenarios *284*
Sharing XML refactoring operations *508*, *1414*
Shortcut actions in Text Mode *292*
Shortcut key configuration *152*
Shortcut Keys *18*
Sign files tool *739*, *1438*
Signing XML document preferences *136*
Single sign-on *266*
Smart Editing in Text Mode *292*
Smart paste preferences *83*
Smart Paste support in Author Mode *343*
Sort table columns in Grid Mode *326*
Sorting a table in Author Mode *407*
Sorting list items in Author Mode *409*
Sorting selected table rows in Author Mode *408*
Sorting tables with merged cells in Author Mode *409*
Special character preferences *105*
Spell check preferences *93*
Spell Checker custom dictionaries in Web Author *1390*
Spell checking
    Add dictionaries *728*
    Add term lists *729*, *730*
    Customizing dictionaries and term lists *727*
    Dictionaries and term lists *727*
    Forbidden words *729*
    Ignored words *731*
    Learned words *730*
    Multiple files *732*
    Replace dictionaries *730*
Spell Checking
    Automatic spell check *731*
SQF
    Content Completion *701*
    Customizing *695*
    Defining *695*
    Embedded *702*
    Executing SQF in Other Documents *700*
    Formatting and Indenting *700*
    Highlight occurrences *701*
    Integrating in a framework *703*
    Operations *697*
    Refactoring actions *701*
    Restricting operations *699*
    Search actions *701*
    Use-When Condition *699*
    User Entry operation *699*
    Validation *700*
SQL transformation scenario *924*
SSH connection settings *159*
StackOverflowException error *112*
Startup parameter
    Application launcher parameters *178*
    Command line script parameters *179*
    Custom startup parameters file *179*
Startup parameters *178*
Storing global options *162*
Storing project level options *162*
Storing XML refactoring operations *508*, *1414*
StratML documents
    Editing features *679*
Styles Filter plugin extension *1341*
Styles in Author mode *330*
Supported document types *763*, *852*
SVG files
    SVG Viewer *704*, *1447*
    SVG viewer in Results panel *705*
SVG Viewer in Results panel *705*
SVG Viewer tool *704*, *1447*
SVN Client
    Authentication *1480*
    Branches/Tags
        Create branch/tag *1503*
        Exporting resources *1527*
        Importing resources *1525*
        Manage resources *1528*
        Merging *1504*
        Patching *1516*
        Relocate working copy *1516*
        Switch repository location *1514*
    Checking out a working copy *1483*
    Define a working copy *1482*
    Entering local paths and URLs *1559*
    History dialog box *1485*
    Manage repository locations *1479*
    Menus *1470*
    Preferences *1559*
    Properties *1502*
    Request history for a resource *1502*
    Request status information for a resource *1501*
    Resource History view
        Directory Change Set view *1547*
    Revision Graph *1555*
    Share projects *1481*
    Sparse checkouts *1529*
    Status bar *1479*
    Synchronize with the SVN repository
        Conflicts *1493*
        Integrating bug tracking tools *1500*
        Send changes to repository *1498*
        Update working copy *1497*
        View differences *1491*
    Technical issues *1560*
    Toolbar *1478*
    Use an existing working copy *1485*
    Views
        Annotations view *1548*, *1555*
        Compare Images view *1553*
        Compare view *1551*
        Dynamic Help view *1555*
        Editor view *1548*
        History view *1543*
        Image Preview panel *1553*
        Properties view *1553*
        Repositories view *1530*
        Working Copy view *1533*

Working copy resource management
Add resources 1486
Copy resources 1488
Delete resources 1488
Edit files 1486
Ignore resources 1488
Lock/Unlock resources 1489
Move resources 1489
Rename resources 1489
SVN message preferences 144
SVN preferences 142
Switch file tabs 266
Syntax highlight preferences 104
Syntax highlighting in Text Mode 308
Syntax highlights in Text mode 206
System properties 174

T

Table editing features in Author Mode 385
Table Explorer view 1069
Tables in Author Mode 385
Tags Display Mode 234, 334
Tags transparency selector 311
TEI ODD document type
Author mode actions 844
TEI ODD menu actions 844
TEI ODD toolbar actions 844
TEI P5 document type
Author mode actions 838
TEI P5 menu actions 838
TEI P5 toolbar actions 838
TEI tables in Author Mode 406
Text editing mode
Attributes view 199, 305
Bidirectional text 207
Content completion 300
Contextual menu actions 318
Drag and Drop 298
Editing XML markup 294
Elements view 202, 306
Entities view 202, 229, 307, 353
File related actions 311
Folding 298
Format and indent 312
Format and indent multiple files 316, 1427
Highlight ID occurrences 317
Lock/Unlock XML tags 312
Manage highlighted content 316
Markup transparency 311
Model view 201, 227, 304, 348
Navigation 186, 290
Outline view 197, 308
Rectangular selection 299
Schema annotations 302
Selecting content 299
Set indent to zero 315
Set schema for content completion 302
Shortcut actions 292
Smart Editing 292
Syntax highlighting 308
Syntax highlights 206
Validation errors 206, 447
Views 188
Text mode preferences 78
Three-way file comparison 742, 1449
Too many nested apply-templates calls 1565
Toolbar configuration 167
Track Changes feature in Author Mode 358
Transform DITA document to MS Word 830
Transformation Scenarios
Ant 907
Batch transformation 929
Built-in transformation scenarios 855
Configure 925
Configure Calabash with XEP 933
Configure Transformation Scenario dialog box 926
Configure XSLT processor extension paths 935
Custom XSLT processor 935
Debugging PDF transformations 932
DITA map 808, 863
DITA Map to CHM (Compiled Help 832, 886
DITA Map to Kindle 832, 886
DITA Map to MS Office Word 830
DITA Map to PDF WYSIWYG 828, 884, 1695
DITA Map to WebHelp Classic 815, 871
DITA Map to WebHelp Classic Mobile 820, 876
DITA Map to WebHelp Classic with Feedback 818, 873
DITA Map to WebHelp Responsive 808, 864
DITA Map to WebHelp Responsive with Feedback 812, 867
DITA OT 898, 1680
DocBook 4 771, 855
DocBook 4 to DITA 775, 789, 858, 863
DocBook 4 to EPUB 774, 789, 859, 863
DocBook 4 to PDF 774, 788, 858, 862
DocBook 4 to WebHelp 771, 855
DocBook 5 785, 859
DocBook 5 to DITA 775, 789, 858, 863
DocBook 5 to EPUB 774, 789, 859, 863
DocBook 5 to PDF 774, 788, 858, 862
DocBook 5 to WebHelp 785, 859
DOCX DITA 1718
Duplicate 926
Integrate external XProc engine 933
New
Ant 907
DITA OT 898
SQL 924
XML with XQuery 893
XML with XSLT 887
XProc 916
XQuery 919
XSLT 910
PDF output using Calabash XProc processor 933
Sharing 929
SQL 924
Supported XSLT processors 933
XML with XQuery 893
XML with XSLT 887
XProc 916
XQuery 919
XSL-FO processors 936
XSLT 910
Transformation Scenarios view 930
Transformation types 925
Transforming Documents 854
Travis CI integration with WebHelp plugin 1053
Tree editor preferences 159
Tree Editor tool 1449
Troubleshooting errors in Web Author 1394

Trusted hosts connection settings 159
Two-way file comparison 742, 1449

U

Unicode support
- Character Map 250
- Fallback Font Support 251
- Inserting symbols 250
- Opening/Saving documents with unsupported characters 249

Uninstalling Oxygen 46
Upgrading Oxygen 45
Upgrading Web Author 1374
Uppercase plugin 1346
URL Stream Handler plugin extension 1341
Use-When SQF Condition 699
User Entry SQF operation 699
User roles in Author Mode 329
UTF-8 BOM handling 75

V

Validating XML Documents
- Against a schema 446
- Against Schematron 686
- Author Mode 236, 448
- Automatic validation 446
- Check Well-formedness 444
- Create new validation scenarios 452
- Custom validators 450
- Customizing error messages 450
- Edit validation scenarios 456
- Manual validation 446
- References to schema specifications 460
- Resolving references to remote schemas 461
- Sharing scenarios 460
- Text Mode 206, 447

Validation engine 'StackOverflowException' error 112
Validation errors in Author mode 236, 448
Validation errors in Text mode 206, 447
Validation preferences 95
Verify Signature tool 740, 1440
Visual hints in Author mode 235

W

Web Author Form controls and properties 1385
WebDAV connection 1111
WebDAV connection settings 157
WebDAV contextual menu actions 1112
WebDAV over HTTPS 264
WebHelp Classic Mobile system
- Customizing 1024

WebHelp Classic system
- Add button in code snippets (DITA) 1001, 1025
- Add custom HTML content 1000, 1024
- Add Facebook widget 1007, 1031
- Add Favicon 968, 1001, 1025
- Add Google+ widget 1007, 1032
- Add logo image 968, 1001
- Add Twitter widget 1008, 1032
- Add video and audio objects in DITA 968, 1002, 1026
- Add videos in DocBook 1003, 1027
- Adding additional resources 966, 1000, 1024
- Changing icons in the table of contents 1003, 1028
- Changing the style of WebHelp Mobile pages 1027
- CSS styling to customize WebHelp output 971, 1004, 1028
- Customize highlights in the table of contents 1004, 1029
- Customize ordered lists 971, 1003, 1027
- Customizing
  - Overriding the XSLT processing step 979, 1012, 1036
  - Overriding XSLT stylesheet from Ant build file 982, 1015, 1039
  - XSLT-Import extension point 979, 1012, 1036
  - XSLT-Parameter extension point 979, 1012, 1036
- Disable caching 1005, 1029
- Edit scoring in search results 972, 1005, 1030
- Exclude DITA topics from search results 973, 1006, 1030
- Flag DITA content 973, 1006, 1030
- GET parameters 1020, 1043
- Indexing Japanese content 978, 1012, 1035
- Integrate Google Analytics 1008, 1033
- Integrate Google Search 1009, 1033
- Localize email notifications 977, 1010
- Localizing the interface for DITA transformations 978, 1011, 1034
- Localizing the interface for DocBook transformations 1011, 1035
- Overriding the XSLT processing step 979, 1012, 1036
- Overriding XSLT stylesheet from Ant build file 982, 1015, 1039
- Publishing on SharePoint Site 1023, 1045
- Remove Previous/Next links 1018, 1041
- Right-to-Left languages 985, 1019, 1043
- Search engine optimization 984, 1018, 1042
- Skin Builder 1019
- Use custom CSS 1005, 1029
- XSLT-Import extension point 979, 1012, 1036
- XSLT-Parameter extension point 979, 1012, 1036

WebHelp Classic with Feedback system
- Admin Panel 949, 998
- Deploying 947, 996
- Installing 947, 996
- Manage comments 949, 998
- Refreshing content 949, 998

WebHelp Mobile system 1023
WebHelp Plugin
- DITA
  - Apply custom styling to an external transformation 1051
  - Configuring the comments database 1052
  - Integrate Output with Jenkins 1052
  - Integrate Output with Travis CI 1053
  - Integration with DITA OT 1046
  - Licensing 1046
  - Running external transformation 1047
  - Upgrading 1047
- DocBook
  - Configuring the comments database 1056
  - Integration with DocBook XSL 1054
  - Licensing 1054
  - Running external transformation 1055
  - Upgrading 1054

WebHelp Responsive system
- Add custom HTML content 967
- Add Facebook widget 974
- Add Favicon 968, 1001, 1025
- Add Google+ widget 974

Add logo image 968, 1001
Add Tweet widget 975
Add video and audio objects in DITA 968, 1002, 1026
Add welcome message 968
Adding additional resources 966, 1000, 1024
Configure tiles 970
CSS styling to customize WebHelp output 971, 1004, 1028
Customization methods 963
Customize ordered lists 971, 1003, 1027
Customize the menu 972
Customizing
Overriding the XSLT processing step 979, 1012, 1036
Overriding XSLT stylesheet from Ant build file 982, 1015, 1039
XSLT-Import extension point 979, 1012, 1036
XSLT-Parameter extension point 979, 1012, 1036
Customizing side TOC 980, 1013, 1037
Edit scoring in search results 972, 1005, 1030
Exclude DITA topics from search results 973, 1006, 1030
Flag DITA content 973, 1006, 1030
GET parameters 986
Indexing Japanese content 978, 1012, 1035
Integrate Google Analytics 976
Integrate Google Search 976
Localize email notifications 977, 1010
Localizing the interface for DITA transformations 978, 1011, 1034
Overriding the XSLT processing step 979, 1012, 1036
Overriding XSLT stylesheet from Ant build file 982, 1015, 1039
Publishing on SharePoint Site 988
Right-to-Left languages 985, 1019, 1043
Search engine optimization 984, 1018, 1042
Templates
Components 957
Directory structure 951
HTML template files 953
Template page types 953
XSLT-Import extension point 979, 1012, 1036
XSLT-Parameter extension point 979, 1012, 1036
WebHelp Responsive with Feedback system
Admin Panel 949, 998
Deploying 947, 996
Installing 947, 996
Manage comments 949, 998
Refreshing content 949, 998
WebHelp System Output
Context-Sensitive WebHelp Classic 1021, 1043
Context-Sensitive WebHelp Responsive 986
WebHelp Classic 988
WebHelp Classic with Feedback 992
WebHelp Mobile 1023
WebHelp Responsive 940
WebHelp Responsive with Feedback 944
Whitespace handling in Author mode 238
Workspace Access (JS-based) plugin extension 1344
Workspace Access plugin extension 1342
WSDL documents
Component Dependencies view 639
Content completion 632
Editing in Master Files context 631
Generate documentation for WSDL documents
Custom format 646
From command line 646
HTML format 645
Highlight occurrences 640
Moving resources 638
Outline view 633
Quick Assist feature 642
Refactoring actions 640
Renaming resources 638
Resource Hierarchy/Dependencies view 636
Search actions 640
SOAP analyzer
Test remote files 648, 1442
UDDI Registry Browser 649, 1442
Syntax highlighting 633
Validation 632
WSDL SOAP Analyzer tool
Test remote files 648, 1442
UDDI Registry Browser 649, 1442

X

XHTML document type
Author mode actions 834
XHTML documents
Editing features 706
XHTML menu actions 834
XHTML tables in Author Mode 404
XHTML toolbar actions 834
XInclude for editing large XML documents 486
XLIFF documents
Editing features 679
XML catalog preferences 113
XML Catalogs 484
XML documents
Associate schema 464
Associate schema directly in XML documents 469
Associate schema in a framework configuration 471
Associate schema in a validation scenario defined in framework 467
Associate schema through a validation scenario 465
Author Mode editing
Adding media resources 430
Apply profiling attributes 377
Apply profiling condition sets 380
Attributes view 224, 350
Content completion 345
Contextual menu actions 437
Create/Edit profiling attributes 375
Create/Edit profiling condition sets 379
Drag and Drop 343
Editing attributes 341
Editing content 334
Editing XML markup 336
Elements view 228, 353
Entities view 202, 229, 307, 353
Find/Replace text 357
Folding 342
Form Controls 436
Generating IDs 433
Image Map Editor
DITA 415, 1628
DocBook 419
TEI 422
XHTML 426
Image rendering
AI images 413
CGM images 412

EPS images 413
JAI images 413
PDF images 412
PSD images 412
Inserting images 410
MathML 431
MathML equations in HTML output 433
Model view 201, 227, 304, 348
Navigation 211, 332
Outline view 222, 354
Profiling 374
Profiling colors and styles 383
Profiling/Conditional Text menu 382
Refreshing content 433
Rendering documents 330
Review tools
Callouts 367
Comments 364
Highlights 366
Review view 371
Track Changes 358
Schema annotations 347
Selecting content 344
Set schema for content completion 347
Smart Paste 343
Tables
DITA 394, 1633
DocBook 387
Editing features 385
Sorting a table 407
Sorting list items 409
Sorting selected table rows 408
Sorting tables with merged cells 409
TEI 406
XHTML 404
User roles 329
Using Retina/HiDPI Images 413
Code templates 308, 354
DTD Entities for editing large documents 486
Editing in Master Files context 481
Find All Elements dialog box 474
Find/Replace dialog box 472
Find/Replace in multiple files 476
Find/Replace text 472
Grid Mode editing
Add nodes 326
Clear content 326
Content Completion Assistant 329
Copy/Paste 327
Drag and Drop 326
Duplicate nodes 326
Edit cell value 326
Insert columns 326
Insert rows 326
Refresh layout 326
Sort columns 326
Highlights 489
Learn document structure 471
Moving resources 483
Navigating Find/Replace matches 475
Printing 490
Quick Assist feature 317
Quick Find toolbar 475
Quick fix support 462
Refactoring
Custom operations 497, 1406
Localizing operations 509, 1415
Predefined operations 493, 1402
Sharing operations 508, 1414
Storing operations 508, 1414
Renaming resources 483
Resolve schemas through XML catalogs 471, 485
Resource Hierarchy/Dependencies view 482
Results view
Make persistent copy 488
Search actions for IDs 479
Status information 488
Text Mode editing
Attributes view 199, 305
Content completion 300
Contextual menu actions 318
Drag and Drop 298
Editing XML markup 294
Elements view 202, 306
Entities view 202, 229, 307, 353
File related actions 311
Folding 298
Format and indent 312
Format and indent multiple files 316, 1427
Highlight ID occurrences 317
Lock/Unlock XML tags 312
Manage highlighted content 316
Markup transparency 311
Model view 201, 227, 304, 348
Navigation 186, 290
Outline view 197, 308
Rectangular selection 299
Schema annotations 302
Selecting content 299
Set indent to zero 315
Set schema for content completion 302
Shortcut actions 292
Smart Editing 292
Syntax highlighting 308
Validation
Against a schema 446
Against Schematron 686
Author Mode 236, 448
Automatic validation 446
Check Well-formedness 444
Create new validation scenario 452
Custom validators 450
Customizing error messages 450
Edit validation scenario 456
Manual validation 446
References to schema specifications 460
Resolving references to remote schemas 461
Sharing scenarios 460
Text Mode 206, 447
XInclude for editing large documents 486
XML catalogs 484
XML documents without a schema 471
XML parser preferences 115
XML refactoring preferences 136
XML Refactoring tool
Custom operations 497, 1406
Localizing operations 509, 1415
Predefined operations 493, 1402
Sharing operations 508, 1414

Storing operations 508, 1414
XML Schema Design mode 240, 553
XML Schema Diagram Editor
Attributes view 244, 588
Components 561
Contextual menu actions 555
Edit namespaces 583
Editing actions 554
Facets view 246, 590
Navigation 241, 554
Outline view 242, 586
Palette view 247, 589
Validation 582
XML Schema Regular Expression Builder tool 615, 1443
XML Schemas
Attributes view 244, 588
Component Dependencies view 593
Content completion 585
Convert DB Structure to XML Schema tool 612, 1423
Design mode editing
Components
Grouping components 580
xs:all 575
xs:alternative 571
xs:any 576
xs:anyAttribute 576
xs:assert 579
xs:attribute 565
xs:attributeGroup 567
xs:choice 575
xs:complexType 568
xs:element 562
xs:field 579
xs:group 572
xs:import 573
xs:include 573
xs:key 578
xs:keyRef 578
xs:notation 574
xs:openContent 580
xs:override 574
xs:redefine 573
xs:schema 561
xs:selector 578
xs:sequence 575
xs:simpleType 570
xs:unique 577
Contextual menu actions 555
Edit namespaces 583
Editing actions 554
Facets view 246, 590
Navigation 241, 554
Palette view 247, 589
Validation 582
Editing in Master Files context 583
Flatten Schema tool 613, 1423
Generate documentation for XML Schema
Customize PDF output 608
From command line 609
Output formats 605
Generate Sample XML Files tool
Advanced tab 601, 1419
From command line 601, 1420
Options tab 599, 1417
Schema tab 597, 1416
Generate/Convert Schema tool 610, 1421
Highlight occurrences 595
Moving resources 593
Outline view 242, 586
Quick Assist feature 596
Refactoring actions 595
References to specification 584
Regular Expressions Builder 615, 1443
Renaming resources 593
Resource Hierarchy/Dependencies view 591
Search actions 595
Set version 617
Syntax highlighting 585
Text mode editing 583
Validation 584
XML Schema 1.1 support 616
XML to JSON tool 678, 1425
XML transformation with XQuery 893
XML transformation with XSLT 887
XPath Expressions
Catalogs 1062
Prefix mapping 1062
Toolbar 1057
XPath Builder view 1059
XPath Results view 1061
XProc Scripts
Editing features 683
XProc transformation scenario
Inputs tab 917
Options tab 918
Outputs tab 917
Parameters tab 917
XProc tab 917
XQuery
Content completion 619
Folding 620
Format and Indent 620
Generate HTML documentation 627, 1434
Input view 522, 624
Outline view 621
Sequence view 628
Syntax highlighting 619
Transformations
Sequence view 628
Updating XML documents 630
XQJ 1118
Updating XML documents 630
Validation 618
XQuery Builder view 622
XQuery Debugger perspective
Breakpoints 1156
Identify expressions 1157
Information views
Breakpoints view 1148
Context Node view 1146
Messages view 1149
Nodes/Values Set view 1153
Output Mapping Stack view 1150
Stack view 1150
Templates view 1153
Trace History view 1151
Variables view 1154
XPath Watch view 1147
Java extensions 1159
Layout 1143

Profiling
Hotspots view *1162*
Invocation Tree view *1161*
Steps in a typical debugging process *1156*
Toolbar *1144*
XQuery Documentation tool *627*, *1434*
XQuery preferences *125*
XQuery Profiler preferences *128*
XQuery transformation scenario
FO Processor tab *897*, *923*
Output tab *897*, *923*
XQuery tab *894*, *919*
XSLT
Component Dependencies view *528*
Component documentation *530*
Content completion *514*
Content completion in XPath expressions *515*
Editing features *510*
Editing in Master Files context *510*
Generate documentation for XSLT stylesheets
Custom format *541*
From command line *542*
HTML format *539*
Highlight occurrences *529*
Input view *522*, *624*
Moving resources *527*
Outline view *519*
Quick Assist feature *533*
Quick fix support *512*
Refactoring actions *531*
Renaming resources *527*
Resource Hierarchy/Dependencies view *525*
Search declarations *529*
Search references *529*
Syntax highlighting *518*
Unit test (XSpec) *535*
Validation *510*
XPath Tooltip Helper *518*
XSLT Debugger perspective
Breakpoints *1156*
Identify expressions *1157*
Information views
Breakpoints view *1148*
Context Node view *1146*
Messages view *1149*
Nodes/Values Set view *1153*
Output Mapping Stack view *1150*
Stack view *1150*
Templates view *1153*
Trace History view *1151*
Variables view *1154*
XPath Watch view *1147*
Java extensions *1159*
Layout *1143*
Profiling
Hotspots view *1162*
Invocation Tree view *1161*
Steps in a typical debugging process *1156*
Toolbar *1144*
XSLT Import extension points for WebHelp *979*, *979*, *1012*, *1012*, *1036*, *1036*
XSLT Parameter extension points for WebHelp *979*, *979*, *1012*, *1012*, *1036*, *1036*
XSLT preferences *120*
XSLT Profiler preferences *128*
XSLT transformation scenario
FO Processor tab *892*, *915*
Output tab *892*, *915*
XSLT tab *887*, *910*
XSLTProc preferences *123*

Copyright

Oxygen XML Editor User Manual

Syncro Soft SRL.

Copyright © 2002-2017 Syncro Soft SRL. All Rights Reserved.

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher. Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

Trademarks. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this document, and Syncro Soft SRL was aware of a trademark claim, the designations have been rendered in caps or initial caps.

Notice. While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Link disclaimer. Syncro Soft SRL is not responsible for the contents or reliability of any linked Web sites referenced elsewhere within this documentation, and Syncro Soft SRL does not necessarily endorse the products, services, or information described or offered within them. We cannot guarantee that these links will work all the time and we have no control over the availability of the linked pages.

Warranty. Syncro Soft SRL provides a limited warranty on this product. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Oxygen XML Editor End User License Agreement, as well as information regarding support for this product, while under warranty, is available through the *Oxygen XML Editor website*.

Third-party components. Certain software programs or portions thereof included in the Product may contain software distributed under third party agreements ("Third Party Components"), which may contain terms that expand or limit rights to use certain portions of the Product ("Third Party Terms"). Information identifying Third Party Components and the Third Party Terms that apply to them is available on the *Oxygen XML Editor website*.

Downloading documents. For the most current versions of documentation, see the *Oxygen XML Editor website*.

Contact Syncro Soft SRL. Syncro Soft SRL provides telephone numbers and e-mail addresses for you to report problems or to ask questions about your product, see the *Oxygen XML Editor website*.