

Oxygen XML Developer 18.0

Contents

Chapter 1: Introduction.....	15
Chapter 2: Getting Started.....	17
What is Oxygen XML Developer.....	18
Getting Familiar with the Layout.....	18
Resources to Help You Get Started Using Oxygen XML Developer	19
Your First Document or Project.....	20
Creating a New Project.....	20
Getting Help.....	21
Help Menu.....	21
Chapter 3: Installation.....	25
Installation Options.....	26
Windows Installation.....	26
Mac OS X Installation.....	28
Linux Installation.....	29
Windows Terminal Server Installation.....	31
Linux Server Installation.....	32
Java Web Start (JWS) Installation.....	34
Site-wide deployment.....	36
Licensing.....	36
Choosing a License Type.....	37
Obtaining a License.....	37
Register a Named-User License.....	37
Registering a Floating License.....	38
Setting up a License Server.....	40
Setting up an HTTP Floating License Server.....	40
Setting up a TCP Floating License Server Using a 32-bit Windows Installer.....	44
Setting up a TCP Floating License Server Using an All-Platforms Distribution.....	46
Transferring or Releasing a License.....	47
Upgrading.....	47
Installing and Updating Add-ons.....	48
Uninstalling.....	49
Installer Command Line Reference.....	50
Chapter 4: Configuration.....	53
Preferences.....	54

Global Preferences.....	55
Appearance Preferences.....	56
Application Layout Preferences.....	58
Add-ons Preferences.....	59
Document Type Association Preferences.....	59
Encoding Preferences.....	73
Editor Preferences.....	74
CSS Validator Preferences.....	99
XML Preferences.....	99
DITA Preferences.....	122
Data Sources Preferences.....	124
SVN Preferences.....	128
Diff Preferences.....	132
Archive Preferences.....	135
Plugins Preferences.....	136
External Tools Preferences.....	137
Menu Shortcut Keys Preferences.....	139
File Types Preferences.....	140
Open/Find Resource Preferences Page.....	140
Custom Editor Variables Preferences.....	141
Network Connection Settings Preferences.....	142
XML Structure Outline Preferences.....	145
Views Preferences.....	145
Messages Preferences.....	145
Configuring Options.....	146
Customizing Default Options.....	147
Sharing and Storing Options.....	148
Importing / Exporting Global Options.....	149
Reset Global Options.....	149
Associating a File Extension with Oxygen XML Developer.....	150
Configuring the Layout of the Views and Editors.....	150
Configure Toolbars.....	152
Scenarios Management.....	153
Editor Variables.....	154
Custom Editor Variables.....	159
Custom System Properties.....	159
Localizing the User Interface.....	161
Setting a Java Virtual Machine Parameter in the Launcher Configuration File / Start-up Script.....	162
Setting Parameters for the Application Launchers.....	162
Setting Parameters in the Command Line Scripts.....	163
Creating Custom Startup Parameters File.....	163

Chapter 5: Perspectives.....165

Editor Perspective.....	166

Supported Document Types.....	167
XSLT Debugger Perspective.....	167
XQuery Debugger Perspective	168
Database Perspective	169

Chapter 6: Editing Modes.....171

Text Editing Mode.....	172
Text Mode Editor.....	172
Text Mode Views.....	174
Syntax Highlight Depending on Namespace Prefix.....	190
Presenting Validation Errors in Text Mode.....	191
Bidirectional Text Support in Text Mode.....	192
Grid Editing Mode.....	192
Grid Mode Editor.....	192
Layouts: Grid and Tree.....	194
Grid Mode Navigation.....	194
Bidirectional Text Support in Grid Mode.....	195
Design Editing Mode.....	196
XML Schema Diagram Editor (Design Mode).....	196
Navigation in the XML Schema Design Mode.....	197
XML Schema Outline View.....	198
XML Schema Attributes View.....	200
XML Schema Facets View.....	201
XML Schema Palette View.....	202

Chapter 7: Editing Documents.....205

Working with Unicode.....	206
Opening and Saving Unicode Documents.....	206
Inserting Symbols.....	206
Unicode Fallback Font Support.....	208
Creating and Working with Documents.....	209
Creating New Documents and Templates.....	209
Opening Documents.....	214
Saving Documents.....	216
Opening and Saving Remote Documents via FTP/SFTP/WebDAV/SharePoint	217
Switching Between Opened Tabs.....	223
Searching Documents.....	223
Closing Documents.....	230
Contextual Menu of the Current Editor Tab.....	230
Viewing File Properties.....	231
Using Projects to Group Documents.....	231
Creating a New Project.....	231
Project View.....	233

Sharing a Project - Team Collaboration.....	240
Defining Master Files at Project Level.....	242
Editing XML Documents.....	245
Editing XML Documents in Text Mode.....	245
Editing XML Documents in Grid Mode.....	277
Validating XML Documents.....	280
Finding and Replacing Text in the Current File.....	295
Finding and Replacing Text in Multiple Files.....	299
Search and Refactor Actions for IDs and IDREFS.....	302
Search and Refactor Operations Scope.....	303
Associate a Schema to a Document.....	304
Working with Modular XML Files in the Master Files Context.....	308
XML Resource Hierarchy/Dependencies View.....	308
Working with XML Catalogs.....	310
Editing Large XML Documents with DTD Entities or XInclude.....	312
Viewing Status Information.....	315
Making a Persistent Copy of Results.....	315
Editor Highlights.....	316
Printing a File.....	316
XML Quick Fixes.....	318
Refactoring XML Documents.....	319
Editing XSLT Stylesheets.....	335
Editing XSLT Stylesheets in the Master Files Context.....	335
Validating XSLT Stylesheets.....	335
Content Completion in XSLT Stylesheets.....	337
Syntax Highlight.....	342
XSLT Outline View.....	342
XSLT/XQuery Input View.....	345
XSLT Resource Hierarchy/Dependencies View.....	346
XSLT Component Dependencies View.....	349
Highlight Component Occurrences.....	350
Finding XSLT References and Declarations.....	350
XSLT Stylesheet Documentation Support.....	350
Generating Documentation for an XSLT Stylesheet.....	351
XSLT Quick Assist Support.....	358
XSLT Quick Fix Support	359
XSLT Refactoring Actions.....	361
XSLT Unit Test (XSpec).....	364
Editing Ant Build Files.....	365
Editing Ant Build Files in the Context of Master Files.....	365
Validate Ant Build Files.....	366
Content Completion in Ant Build Files.....	366
Syntax Highlight.....	367
Ant Outline View.....	367
Ant Resource Hierarchy/Dependencies View.....	369

Ant Component Dependencies View.....	370
Highlight Component Occurrences.....	370
Find References and Declarations of Ant Components.....	371
Ant Quick Assist Support.....	371
Ant Quick Fix Support.....	372
Ant Refactoring Actions.....	372
Editing XML Schemas.....	373
XML Schema Diagram Editor (Design Mode).....	373
Editing XML Schema in Text Editing Mode.....	404
Editing XML Schema in the Master Files Context.....	404
Validating XML Schema Documents.....	404
Content Completion in XML Schema.....	405
XML Schema Outline View.....	406
XML Schema Attributes View.....	407
XML Schema Resource Hierarchy / Dependencies View.....	409
Component Dependencies View for XML Schema.....	411
Highlight Component Occurrences.....	413
Searching and Refactoring Actions in XML Schemas.....	413
XML Schema Quick Assist Support.....	414
Generating Sample XML Files.....	415
Generating Documentation for an XML Schema.....	420
Converting Schema to Another Schema Language.....	428
Converting Database to XML Schema.....	430
Flatten an XML Schema.....	431
XML Schema Regular Expressions Builder.....	433
XML Schema 1.1.....	435
Setting the XML Schema Version.....	436
Editing XQuery Documents.....	437
XQuery Outline View.....	437
Folding in XQuery Documents.....	438
Formatting and Indenting XQuery Documents.....	439
Generating HTML Documentation for an XQuery Document.....	439
Editing WSDL Documents.....	440
Editing WSDL Documents in the Master Files Context.....	441
Validating WSDL Documents.....	441
Content Completion Assistance in WSDL Documents.....	442
WSDL Outline View.....	442
WSDL Resource Hierarchy/Dependencies View in WSDL Documents.....	445
Component Dependencies View in WSDL Documents.....	448
Highlight Component Occurrences in WSDL Documents.....	448
Searching and Refactoring Operations in WSDL Documents.....	449
Searching and Refactoring Operations Scope in WSDL Documents.....	450
Quick Assist Support in WSDL Documents.....	451
Generating Documentation for WSDL Documents.....	452
WSDL SOAP Analyzer.....	456

Editing CSS Stylesheets.....	459
Validating CSS Stylesheets.....	460
Content Completion in CSS Stylesheets.....	460
CSS Outline View.....	461
Folding in CSS Stylesheets.....	461
Formatting and Indenting CSS Stylesheets (Pretty Print).....	462
Minifying CSS Stylesheets.....	462
Editing LESS CSS Stylesheets.....	462
Validating LESS Stylesheets.....	462
Content Completion in LESS Stylesheets.....	463
Compiling LESS Stylesheets to CSS.....	463
Editing Relax NG Schemas.....	464
Editing Relax NG Schema in the Master Files Context.....	464
Relax NG Schema Diagram Editor.....	464
Validating Relax NG Schema Documents.....	467
Relax NG Outline View.....	468
RNG Resource Hierarchy/Dependencies View.....	469
Component Dependencies View for RelaxNG Schemas.....	472
Searching and Refactoring Actions in RNG Schemas.....	473
RNG Quick Assist Support.....	474
Configuring a Custom Datatype Library for a RELAX NG Schema.....	475
Editing NVDL Schemas.....	475
NVDL Schema Diagram.....	475
Validating NVDL Schema Documents.....	477
Component Dependencies View for NVDL Schemas.....	477
Searching and Refactoring Actions in NVDL Schemas.....	478
Editing JSON Documents.....	479
Editing JSON Documents in Text Mode.....	479
Editing JSON Documents in Grid Mode.....	481
Validating JSON Documents.....	482
JSON Outline View.....	482
XML to JSON Converter.....	482
Editing StratML Documents.....	483
Editing XLIFF Documents.....	484
Editing JavaScript Documents.....	484
JavaScript Editor Text Mode.....	484
Validating JavaScript Files.....	486
Content Completion in JavaScript Files.....	486
JavaScript Outline View.....	487
Editing XProc Scripts.....	488
Editing Schematron Schemas.....	489
Editing Schematron Schema in the Master Files Context.....	490
Validating Schematron Documents.....	490
Content Completion in Schematron Documents.....	490
RELAX NG/XML Schema with Embedded Schematron Rules.....	491

Schematron Outline View.....	492
Schematron Resource Hierarchy/Dependencies View.....	493
Highlight Component Occurrences in Schematron Documents.....	495
Searching and Refactoring Operations in Schematron Documents.....	495
Searching and Refactoring Operations Scope in Schematron Documents.....	496
Quick Assist Support in Schematron Documents.....	497
Editing Schematron Quick Fixes.....	498
Customizing Schematron Quick Fixes.....	498
Validating Schematron Quick Fixes.....	502
Content Completion in SQF.....	502
Highlight Quick Fix Occurrences in SQF.....	503
Searching and Refactoring Operations in SQF.....	503
Embed Schematron Quick Fixes in Relax NG or XML Schema.....	504
Editing SVG Files.....	504
Standalone SVG Viewer.....	505
Integrated SVG Viewer in the Results Panel.....	506
Editing XHTML Documents.....	506
Spell Checking.....	507
Spell Checking Dictionaries.....	508
Learned Words.....	509
Ignored Words (Elements).....	510
Automatic Spell Check.....	510
Spell Checking in Multiple Files.....	510
Loading Large Documents.....	511
File Sizes Smaller than 300 MB.....	511
File Sizes Greater than 300 MB.....	512
Scratch Buffer.....	512
Handling Read-Only Files.....	513
Editing Documents with Long Lines.....	513
XML Digital Signatures.....	513
Digital Signatures Overview.....	513
Certificates.....	515
Canonicalizing Files.....	515
Signing Files.....	516
Verifying Signature.....	518
Example of How to Digitally Sign XML Files or Content.....	518
Compare Files or Directories	519
Compare Files.....	519
Compare Directories.....	530
Chapter 8: Publishing.....	535
Transformation Scenarios.....	536
Built-in Transformation Scenarios.....	536
Creating New Transformation Scenarios.....	560

Editing a Transformation Scenario.....	596
Duplicating a Transformation Scenario.....	597
Configure Transformation Scenario(s) Dialog Box.....	598
Apply Batch Transformations.....	600
Sharing the Transformation Scenarios.....	600
Transformation Scenarios View	601
Debugging PDF Transformations.....	603
Configuring Calabash with XEP	603
Integration of an External XProc Engine.....	604
XSLT Processors.....	604
XSL-FO Processors.....	607
WebHelp System Output.....	611
WebHelp Responsive System	611
WebHelp Classic System.....	642
WebHelp Classic Mobile System.....	667
Context-Sensitive WebHelp System.....	668
Using the Oxygen XML WebHelp Plugin to Automate Output.....	669
 Chapter 9: Querying Documents.....	 679
Running XPath Expressions.....	680
What is XPath.....	680
Oxygen XML Developer XPath Toolbar.....	680
XPath/XQuery Builder View.....	681
XPath Expression Results.....	683
Catalogs.....	684
XPath Prefix Mapping.....	685
Working with XQuery.....	685
What is XQuery.....	685
Syntax Highlight and Content Completion.....	685
XQuery Outline View.....	686
XSLT/XQuery Input View.....	688
XQuery Validation.....	689
Transforming XML Documents Using XQuery.....	690
 Chapter 10: Working with Archives.....	 695
Browsing and Modifying Archives.....	696
Working with EPUB.....	697
Create an EPUB.....	698
Publish to EPUB.....	699
Editing Files From Archives.....	699
 Chapter 11: Databases and CMS.....	 701
Working with Databases.....	702

Data Source Explorer View.....	702
Table Explorer View.....	703
Database Connection Support.....	705
WebDAV Connections.....	741
SQL Execution Support.....	743
XQuery and Databases.....	745
Content Management System (CMS) Integration.....	749
Integration with Documentum (CMS) (deprecated).....	749
Integration with Microsoft SharePoint.....	754
Chapter 12: Importing Data.....	759
Import from Text Files.....	760
Import from MS Excel Files.....	761
Import Data from MS Excel 2007 or Newer.....	763
Import Database Data as an XML Document.....	763
Import from HTML Files.....	766
Import Content Dynamically.....	766
Chapter 13: XSLT and XQuery.....	769
Layout.....	771
Control Toolbar.....	772
Debugging Information Views.....	774
Multiple Output Documents in XSLT 2.0 and XSLT 3.0.....	783
Working with the XSLT / XQuery Debugger.....	783
Steps in a Typical Debugging Process.....	783
Using Breakpoints.....	784
Determining What XSLT / XQuery Expression Generated Particular Output.....	784
Debugging Java Extensions.....	787
Supported Processors for XSLT / XQuery Debugging.....	788
Performance Profiling of XSLT Stylesheets and XQuery Documents.....	788
XSLT/XQuery Performance Profiling Overview.....	788
Viewing Profiling Information.....	788
Working with XSLT/XQuery Profiler.....	790
Chapter 14: Predefined Document Types.....	793
Predefined Document Types (Frameworks).....	794
DocBook 4 Document Type.....	795
DocBook 5 Document Type.....	802
DITA Topics Document Type.....	808
DITA Map Document Type.....	810
XHTML Document Type.....	814
TEI ODD Document Type.....	814
TEI P4 Document Type.....	815

TEI P5 Document Type.....	816
JATS Document Type.....	817
EPUB Document Type.....	817
DocBook Targetset Document Type.....	818

Chapter 15: Extending Oxygen XML Developer Using the SDK.....819

Oxygen XML SDK.....	820
Introduction to Plugins.....	820
General Configuration of an Oxygen XML Developer Plugin.....	820
Installation.....	821
Types of plugin extensions.....	822
Oxygen XML Developer Plugin How to.....	829
Example - A Selection Plugin.....	834
Creating and Running Automated Tests.....	835
Debugging a Plugin Using the Eclipse Workbench.....	837
Disabling a Plugin.....	837

Chapter 16: Tools.....839

XML Refactoring.....	840
Predefined Refactoring Operations.....	842
Custom Refactoring Operations.....	846
Storing and Sharing Refactoring Operations.....	854
Localizing XML Refactoring Operations.....	854
Generating Sample XML Files.....	855
Schema Tab (Generate Sample XML Files Tool).....	856
Options Tab (Generate Sample XML Files Tool).....	857
Advanced Tab (Generate Sample XML Files Tool).....	859
Running the Generate Sample XML Files Tool from the Command Line.....	859
Generate/Convert Schema.....	860
Convert DB Structure to XML Schema.....	862
Flatten Schema.....	863
XML to JSON.....	865
Format and Indent Files.....	866
Generate Documentation.....	867
XML Schema Documentation.....	867
XSLT Stylesheet Documentation.....	870
XQuery Documentation.....	873
WSDL Documentation.....	874
Canonicalize.....	876
Sign.....	878
Verify Signature.....	879
WSDL SOAP Analyzer.....	879
Composing a SOAP Request.....	879

Testing Remote WSDL Files.....	881
UDDI Registry Browser.....	881
XML Schema Regular Expressions Builder.....	882
Large File Viewer.....	884
Hex Viewer.....	886
SVG Viewer.....	886
Tree Editor.....	888
Compare Files.....	888
Compare Files Tool Menus.....	894
Compare Toolbar of the Compare Files Tool.....	897
Compare Directories.....	899
Compare Directories Tool Menus.....	900
Compare Toolbar of the Compare Directories Tool.....	901
Compare Images.....	902
SVN Client.....	902
Main Window.....	902
Getting Started.....	912
Syncro SVN Client Views.....	963
Revision Graph of a SVN Resource.....	990
Oxygen XML Developer SVN Preferences.....	993
Entering Local Paths and URLs.....	994
Technical Issues.....	994
External Tools.....	997

Chapter 17: Common Problems.....999

Performance Problems.....	1000
Performance Issues with Large Documents.....	1000
External Processes.....	1000
Common Problems and Solutions.....	1000
Details to Submit in a Request for Technical Support Using the Online Form.....	1000
XML Document Takes a Long Time to Open.....	1001
Oxygen XML Developer Takes Several Minutes to Start.....	1001
Out Of Memory Error When Opening Large Documents.....	1001
Special Characters are Replaced with a Square in Editor.....	1002
XSLT Debugger Is Very Slow.....	1002
Scroll Function of my Notebook Trackpad is Not Working.....	1002
Crash at Startup on Windows with an Error Message About a File nvogl32.dll.....	1002
Oxygen XML Developer Crashed on My Mac OS X Computer.....	1002
Wrong Highlights of Matched Words in a Search in User Manual.....	1003
Keyboard Shortcuts Do Not Work.....	1003
After Installing Oxygen XML Developer I Cannot Open XML Files in Internet Explorer Anymore.....	1003
I Cannot Associate Oxygen XML Developer With a File Type on My Windows Computer.....	1004
Gray Window on Linux With the Compiz / Beryl Window Manager.....	1004

Drag and Drop Without Initial Selection Does Not Work.....	1004
Set Specific JVM Version on Mac OS X.....	1004
Segmentation Fault Error on Mac OS X.....	1004
Damaged File Associations on OS X.....	1004
I Cannot Connect to SVN Repository from Repositories View.....	1005
Signature Verification Failed Error on Open or Edit a Resource from Documentum.....	1005
Compatibility Issue Between Java and Certain Graphics Card Drivers.....	1006
Image Appears Stretched Out in the PDF Output.....	1006
DITA PDF Transformation Fails.....	1006
<i>DITA to CHM</i> Transformation Fails.....	1007
DITA Map Transformation Fails (Cannot Connect to External Location).....	1007
Topic References Outside the Main DITA Map Folder.....	1007
PDF Processing Fails to Use the DITA OT and Apache FOP.....	1007
Navigation to the web page was canceled when viewing CHM on a Network Drive.....	1008
Alignment Issues of the Main Menu on Linux Systems Based on Gnome 3.x.....	1008
SVG 1.2 Rendering Issues.....	1009
MSXML 4.0 Transformation Issues.....	1009
'Address Family Not Supported by Protocol Family; Connect' Error.....	1009

Chapter 18: Glossary.....	1011

Copyright.....	mxv

Index.....	1017

Chapter 1

Introduction

Welcome to the User Manual of Oxygen XML Developer 18.0.

Oxygen XML Developer 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 offers developers a powerful *Integrated Development Environment* and the intuitive *Graphical User Interface* of Oxygen XML Developer 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 Developer 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 Developer 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.

Chapter 2

Getting Started

Topics:

- [What is Oxygen XML Developer](#)
- [Getting Familiar with the Layout](#)
- [Resources to Help You Get Started Using Oxygen XML Developer](#)
- [Your First Document or Project](#)
- [Getting Help](#)

This chapter is designed to help you get started using Oxygen XML Developer as quickly as possible.

This chapter provides you with a variety of resources to help you get the most out of the application. Typically, the first step of getting started with Oxygen XML Developer would be to install the software. For detailed information about that process, see the [Installation](#) on page 25 chapter.

After installation, when you launch Oxygen XML Developer 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.

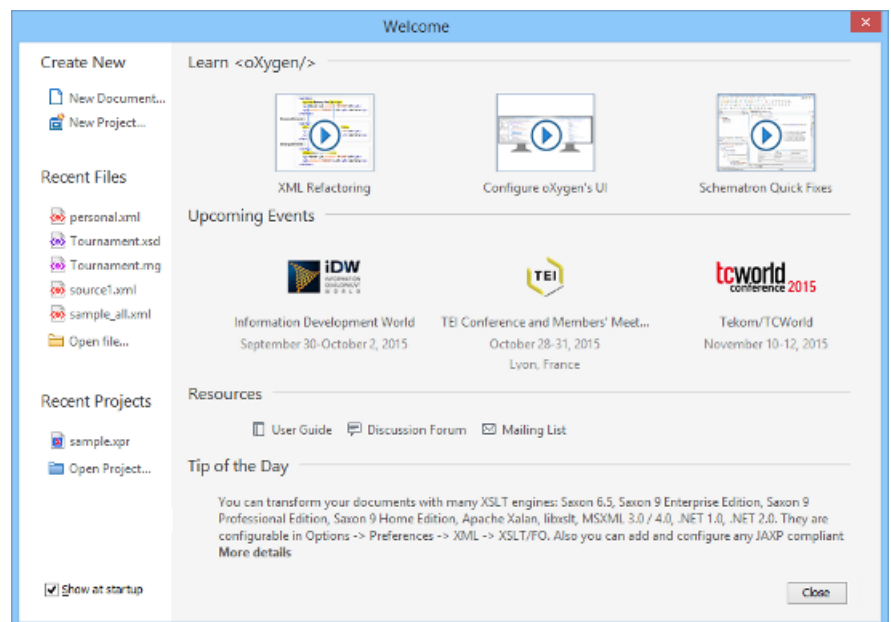


Figure 1: Welcome Dialog Box

If you do not want it to be displayed every time you launch Oxygen XML Developer, disable the **Show at startup** option. To display it any time, go to **Help > Welcome**.

What is Oxygen XML Developer

Oxygen XML Developer 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 Developer 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 Developer, see the [oXygen Website](#).

Getting Familiar with the Layout

Oxygen XML Developer 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 Developer. Most of the menus are common for all types of documents, but Oxygen XML Developer 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 Developer includes a large variety of 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 Developer 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 Developer for the first time. Oxygen XML Developer 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 enabled, 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 Developer website, the announcement of upcoming webinars and conferences where the Oxygen XML Developer 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 enabled, a  **Free unused memory** icon is displayed in the bottom-right corner and you can use this icon to free up unused memory.

Resources to Help You Get Started Using Oxygen XML Developer

Configuring Oxygen XML Developer

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

- See the [Configuring Oxygen XML Developer](#) on page 53 section for details on the various ways that you can configure the application and its features.

Video Tutorials

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

- Go to the [Oxygen XML Developer Videos Page](#) to see the list of video tutorials and webinars.

Oxygen XML Developer Documentation

The Oxygen XML Developer 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 on page 172 Section](#) - Provides information about the **Text** editor.
- [Editing Documents on page 205 Section](#) - Includes information about editing numerous different types of documents.
- [WebHelp System Output on page 611 Section](#) - Provides information about the WebHelp system that can be used for publishing content.
- [Importing Data on page 759 Section](#) - Provides information about importing data from text files, MS Excel files, database data, and HTML files.

Sample Documents

Your installation of Oxygen XML Developer 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 Developer. 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 Developer:

- See the [Oxygen XML Developer 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 Developer 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 Developer, 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 Developer, 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 Developer frameworks and plugins, and to integrate Eclipse plugins.
- For a list of new features that were implemented in the latest version of Oxygen XML Developer, 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 Developer.

Your First Document or Project

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

Creating a New Project

Oxygen XML Developer 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 your 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 much more information on managing projects and their content, see the [Project View](#) on page 174 section.

For more details about how you can share projects with other users, see the [Sharing a Project - Team Collaboration](#) on page 240 section.

Related information

[Using Projects to Group Documents](#) on page 231

Getting Help

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

- [The Oxygen XML Developer Support Section of the Website](#)
- [The Oxygen XML Developer Forum](#)
- [The Oxygen XML Developer 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 Developer **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 Developer.

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 enabled, this action will open the User Manual in an online mode.
Use online help	If this option is enabled, the Help (F1) action will open the Oxygen XML Developer User Manual in an online mode.
Show Dynamic Help view	Use this action to open a dynamic view that automatically loads the relevant help section of the User Manual for the focused editor, view, or dialog box.
Install new add-ons	Opens a dialog box that allows you to install new add-ons to extend the functionality of Oxygen XML Developer.
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 Developer.
Browse <Oxygen/> Website	Opens the Oxygen XML Developer website in your default internet browser.
Register	If you encounter problems with your Oxygen XML Developer license, you can use this option to open a dialog box that provides options for obtaining or using a license key.
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 <i>Class Loader URLs</i> 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 <code>com.oxygenxml.report.problems.url</code> system property. The report is sent in XML format through the <code>report</code> 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 Developer 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 Developer. 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 XML Developer support team.
Tip of the Day	Opens a dialog box that offers tips for using Oxygen XML Developer.
About	Use this option to open a dialog box that contains information about Oxygen XML Developer 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 <i>SGN</i> 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 Developer uses. To view the End User Licence Agreement of each library, double-click it.
- **Frameworks** - This tab contains a list with the frameworks that are bundled with Oxygen XML Developer.
- **System Properties** - This tab contains a list with system properties and their values. The contextual menu allows you to select and copy the properties.

Chapter

3

Installation

Topics:

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

The platform requirements and installation instructions are presented in this chapter.

Installation Options for Oxygen XML Developer

Choosing how Oxygen XML Developer runs

You can install Oxygen XML Developer 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](#).

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 All-platforms installer, which can be used on any supported platform.

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 Developer.

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 Developer, [transfer a license](#), or [uninstall](#) Oxygen XML Developer.

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 Developer on Windows

Choosing an Installer

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

- Install using the [Windows installer](#).
- Install using the [Windows installer in unattended mode](#).
- Install using the [All Platforms installer](#). Choose the all platforms installer if you have trouble installing using the Windows installer.

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 Developer requires Java. If you use the native Windows installer, Oxygen XML Developer 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 Developer 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 Developer may work with JVM implementations from other vendors, but there is no guarantee that those implementations will work with future Oxygen XML Developer updates and releases. Oxygen XML Developer 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 Developer installation, the version in the <code>jre</code> subdirectory of the installation directory is used. 2. Otherwise, if the Windows environment variable <code>JAVA_HOME</code> is set, Oxygen XML Developer uses the Java version pointed to by this variable. 3. Otherwise, the version of Java pointed to by your <code>PATH</code> environment variable is used. If you run Oxygen XML Developer using the batch file, <code>oxygenDeveloper.bat</code>, you can edit the batch file to specify a particular version to use.

Install Using the Windows Installer

To install Oxygen XML Developer 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 Developer using one of the following methods:
 - Using one of the shortcuts created by the installer.
 - By running `oxygenDeveloper.bat`, which is located in the install folder.
6. To license your copy of Oxygen XML Developer 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 Developer 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 Using the All Platforms Installer

To install using the all platforms installer, follow these steps:

1. Download the all platforms installation package (`oxygenDeveloper.tar.gz`) to a folder of your choice.
2. Extract the archive in that folder.
Oxygen XML Developer is now installed in a new sub-folder called `oxygenDeveloper`.

3. Optionally, you can move the directory where you installed Oxygen XML Developer to your applications directory. You can also rename it to contain the product version information. For example, you can rename it as `oxygenDeveloper18.0`.
4. Start Oxygen XML Developer by running `oxygenDeveloper.bat`, which is located in the install directory.
5. To license your copy of Oxygen XML Developer go to **Help > Register** and enter your [license information](#).

Install Oxygen XML Developer on Mac OS X

Choosing an Installer

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

- Install using the Mac OS X installation package, `oxygenDeveloper.zip`.
- Install using the all platforms installer. Choose the all platforms installer if you have trouble installing using the Mac OS X archive installation.

System Requirements

System requirements for a Mac OS X install:

Operating system OS X version 10.6 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 Developer 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 Developer 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 Developer may work with JVM implementations from other vendors, but there is no guarantee that other implementations will work with future Oxygen XML Developer updates and releases.

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

1. If you start Oxygen XML Developer with the application launcher (.app) file then:
 - a. If you use the zip distribution for OS X Oxygen XML Developer 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 Developer will use that bundled JRE
2. If you start Oxygen XML Developer 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 Developer using the `oxygenDeveloper.sh` script, you can change the version of Java used by editing the 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 Developer on OS X:

1. Download the OS X installation package (`oxygenDeveloper.zip`).
The Safari web browser should recognize and expand the compressed file. If it is not automatically expanded, you can expand it manually by double-clicking it.
2. Validate the integrity of the downloaded file by [checking it against the MD5 sum](#) published on the download page.
3. In **Finder**, move the expanded folder to your **Applications** folder.
Oxygen XML Developer is now installed.
4. Start Oxygen XML Developer, using one of the following methods:
 - Double-click `Oxygen XML Developer.app`.
 - Run `sh oxygenDeveloperMac.sh` on the command line.



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

```
open -n OxygenDeveloper.app
```

5. To license your copy of Oxygen XML Developer, go to **Help > Register** to enter your [license key](#).

Install Using the All Platforms Installer

To install using the all platforms installer:

1. Download the all platforms installation package (`oxygenDeveloper.tar.gz`) to a folder of your choice.
2. Extract the archive in that folder.
Oxygen XML Developer is now installed in a new sub-folder called `oxygenDeveloper`.
3. Optionally, you can move the directory where you installed Oxygen XML Developer to your applications directory.
You can also rename it to contain the product version information. For example, you can rename it as `oxygenDeveloper18.0`.
4. Start Oxygen XML Developer by running `oxygenDeveloperMac.sh`, which is located in the install folder.
5. To license your copy of Oxygen XML Developer go to **Help > Register** and enter your [license information](#).

Install Oxygen XML Developer on Linux

Choosing an Installer

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

- Install using the Linux installer.
- Install using the Linux installer in unattended mode.
- Install using the all platforms installer. Choose the all platforms installer if you have trouble installing using the Linux installer.

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 Developer requires Java. Oxygen XML Developer 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 Developer may work with JVM implementations from other vendors, but there is no guarantee that other implementations will work with future Oxygen XML Developer updates and releases. Oxygen XML Developer does not work with the GNU libgcj Java Virtual Machine. Oxygen XML Developer 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 Developer installation, the version in the <code>jre</code> subdirectory of the installation directory is used. 2. Otherwise, if the Linux environment variable <code>JAVA_HOME</code> is set, Oxygen XML Developer uses the Java version pointed to by this variable. 3. Otherwise the version of Java pointed to by your <code>PATH</code> environment variable is used. You can also change the version of the Java Virtual Machine that runs Oxygen XML Developer by editing the script file, <code>oxygenDeveloper.sh</code>. 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 Developer 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 Developer using one of the following methods:
 - Use the `developer` shortcut created by the installer.
 - From a command line, type `sh oxygenDeveloper.sh`. This file is located in the installation folder.
5. To license your copy of Oxygen XML Developer 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 Developer 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 Using the All Platforms Installer

To install using the all platforms installer:

1. Download the all platforms installation package (`oxygenDeveloper.tar.gz`) to a folder of your choice.
2. Extract the archive in that folder.
Oxygen XML Developer is now installed in a new sub-folder called `oxygenDeveloper`.
3. Optionally, you can move the directory where you installed Oxygen XML Developer to your applications directory. You can also rename it to contain the product version information. For example, you can rename it as `oxygenDeveloper18.0`.
4. Start Oxygen XML Developer by running `oxygenDeveloper.sh`, which is located in the install folder.
5. To license your copy of Oxygen XML Developer go to **Help > Register** and enter your [license information](#).

Installing Oxygen XML Developer on Windows Server

Choosing an Installer

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

- Install using the [Windows installer](#).
- Install using the [Windows installer in unattended mode](#).
- Install using the [All Platforms installer](#). Choose the all platforms installer if you have trouble installing using the Windows installer.

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 Developer requires Java. If you use the native Windows installer, Oxygen XML Developer 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 Developer 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 Developer may work with JVM implementations from other vendors, but there is no guarantee that those implementations will work with future Oxygen XML Developer updates and releases.

Oxygen XML Developer 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 Developer 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 Developer 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 Developer using the batch file, `oxygenDeveloper.bat`, you can edit the batch file to specify a particular version to use.

Install Using the Windows Installer

To install Oxygen XML Developer 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 Developer using one of the following methods:
 - Using one of the shortcuts created by the installer.
 - By running `oxygenDeveloper.bat`, which is located in the install folder.
6. To license your copy of Oxygen XML Developer go to **Help > Register** and enter your [license information](#).

Install Using the All Platforms Installer

To install using the all platforms installer, follow these steps:

1. Download the all platforms installation package (`oxygenDeveloper.tar.gz`) to a folder of your choice.
2. Extract the archive in that folder.
Oxygen XML Developer is now installed in a new sub-folder called `oxygenDeveloper`.
3. Optionally, you can move the directory where you installed Oxygen XML Developer to your applications directory. You can also rename it to contain the product version information. For example, you can rename it as `oxygenDeveloper18.0`.
4. Start Oxygen XML Developer by running `oxygenDeveloper.bat`, which is located in the install directory.
5. To license your copy of Oxygen XML Developer go to **Help > Register** and enter your [license information](#).

Configuring Windows Terminal Server

Windows Terminal Server configuration procedure.

1. Install Oxygen XML Developer on the server and make its shortcuts available to all users.
2. If you need to run multiple instances of Oxygen XML Developer, 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 Developer 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 Developer on a Linux / UNIX Server

Choosing an Installer

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

- Install using the Linux installer.
- Install using the Linux installer in unattended mode.
- Install using the all platforms installer. Choose the all platforms installer if you have trouble installing using the Linux installer.

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 Developer requires Java. Oxygen XML Developer 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 Developer may work with JVM implementations from other vendors, but there is no guarantee that other implementations will work with future Oxygen XML Developer updates and releases. Oxygen XML Developer does not work with the GNU libgcj Java Virtual Machine. Oxygen XML Developer 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 Developer installation, the version in the <code>jre</code> subdirectory of the installation directory is used. 2. Otherwise, if the Linux environment variable <code>JAVA_HOME</code> is set, Oxygen XML Developer uses the Java version pointed to by this variable. 3. Otherwise the version of Java pointed to by your <code>PATH</code> environment variable is used. You can also change the version of the Java Virtual Machine that runs Oxygen XML Developer by editing the script file, <code>oxygenDeveloper.sh</code>. 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 Developer 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 Developer using one of the following methods:
 - Use the `developer` shortcut created by the installer.
 - From a command line, type `sh oxygenDeveloper.sh`. This file is located in the installation folder.
5. To license your copy of Oxygen XML Developer go to **Help > Register** and enter your [license key](#).

Install Using the All Platforms Installer

To install using the all platforms installer:

1. Download the all platforms installation package (`oxygenDeveloper.tar.gz`) to a folder of your choice.
2. Extract the archive in that folder.
Oxygen XML Developer is now installed in a new sub-folder called `oxygenDeveloper`.

3. Optionally, you can move the directory where you installed Oxygen XML Developer to your applications directory. You can also rename it to contain the product version information. For example, you can rename it as `oxygenDeveloper18.0`.
4. Start Oxygen XML Developer by running `oxygenDeveloper.sh`, which is located in the install folder.
5. To license your copy of Oxygen XML Developer go to **Help > Register** and enter your [license information](#).

Unix / Linux Server Configuration

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

1. Install Oxygen XML Developer on the server and make sure the `oxygenDeveloper.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 Developer 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](#) on page 1000, 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 Developer by typing `sh oxygenDeveloper.sh` from the command line. This file is located in the installation folder.

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

Oxygen XML Developer 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 Developer 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 Developer 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 Developer Java Web Start deployment procedure:

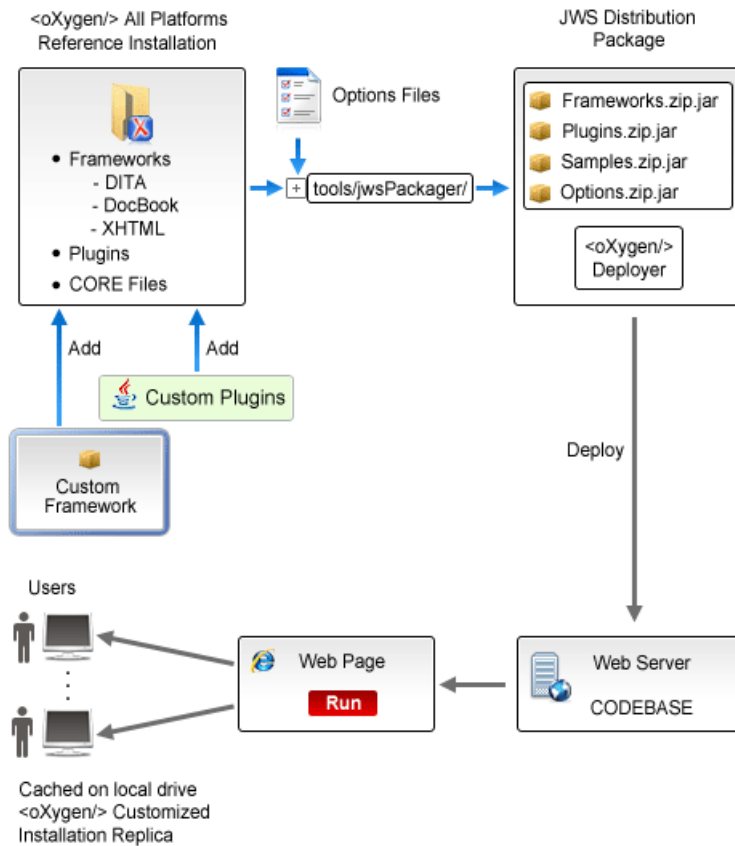


Figure 2: Java Web Start Deployment Procedure

To deploy an Oxygen XML Developer installation on a server.

1. Go to <http://www.oxygenxml.com/InstData/Developer/All/oxygenDeveloper.tar.gz> and download the All Platforms Installation package to a local drive.
2. Expand the archive to a temporary location. The `oxygenDeveloper` folder is created.
3. Optionally, you can customize the content of the **frameworks** folder.
4. Edit the `oxygenDeveloper\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, please check the documentation for your *jarsigner* tool.

5. If you want to modify the default settings, edit the JNLP `oxygenDeveloper\tools\jwsPackager\dist\javawebstart\author\developer.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 Developer 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 Developer with the prefix stripped.
6. Open a command-line console and run `ant` in the `oxygenDeveloper\tools\jwsPackager` folder. The `ant` process creates the `oxygenDeveloper\tools\jwsPackager\dist\InstData\authorJWS.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.

Site-wide Deployment

If you are deploying Oxygen XML Developer for a group, there are a number of things you can do to customize Oxygen XML Developer 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 Developer. 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 Developer, 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 Developer 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 Developer 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 Developer

Oxygen XML Developer is not free software. To enable and use Oxygen XML Developer, you need a license.

For demonstration and evaluation purposes, a time limited license is available upon request at <http://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. If a trial period greater than 30 days is required, please contact support@oxygenxml.com.

Choosing a License Type

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

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 Developer 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 Developer (for example, Standalone and Eclipse Plugin) at the same time on the same computer, consumes a single floating license.

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

Obtaining a License

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

- You can purchase one or more licenses from the Oxygen XML Developer website at <http://www.oxygenxml.com/buy.html>. A license key will be sent to you by email.
- If your company or organization has 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 <http://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 Developer website at <http://www.oxygenxml.com/register.html>.

Register a Named-User License

To register a named-user license on a machine owned by the *Named User*:

1. Save a backup copy of the message containing the new license key.
2. Start Oxygen XML Developer.
If this is a new install of Oxygen XML Developer, the registration dialog box is displayed. If the registration dialog box is not displayed, go to **Help > Register**.

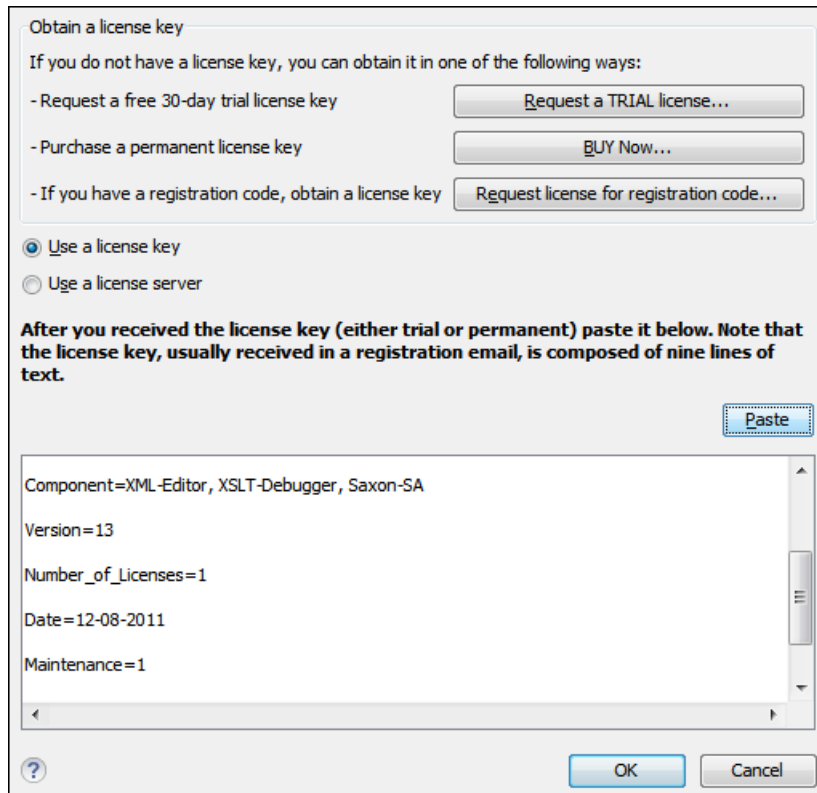


Figure 3: License Registration Dialog Box

3. Select **Use a license key** as licensing method.
4. Paste the license text into the registration dialog box.
5. Press **OK**.

Registering a Floating License

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

- 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 Developer will use your connection to a local area network, without additional notice, to automatically connect to other running instances of Oxygen XML Developer. These connections may transmit your IP address to the local network.



Note: [For System Administrators] Multiple running instances of Oxygen XML Developer 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.

- If the machines sharing 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.

Request a Floating License from a TCP License Server

Use this procedure if your company uses a TCP license server:

1. Contact your server administrator to get network address and login details for the license server.
2. Start Oxygen XML Developer.

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 Developer. 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.

Request a Floating License from an HTTP License Server

1. Contact your server administrator to get network address and login details for the license server.
2. Start Oxygen XML Developer.
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 Developer. 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.

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 Developer running on your machine are using your floating license.
- You register a *Named User* license with your copy of Oxygen XML Developer, and no other copies of Oxygen XML Developer running on your machine are using your floating license.

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 Developer 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 Developer [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 Developer, you must set up an Oxygen XML Developer 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 Developer 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 Developer 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 Developer 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 Developer 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

Setting up the floating license server as an HTTP server.

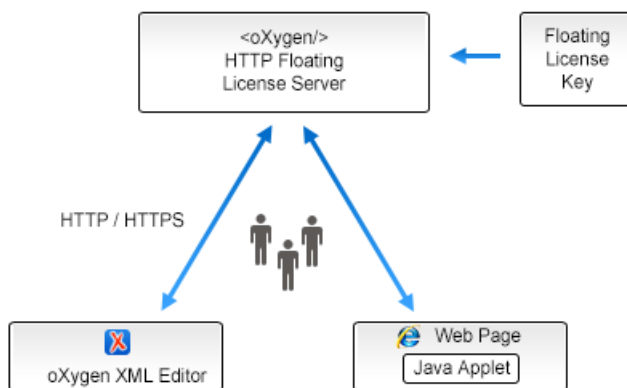


Figure 4: Floating License Server (HTTP Server)

The Oxygen XML Developer 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 Developer 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 Developer license server administrative interface. Optionally you can choose to change the standard 8080 port.
 - b. **Standard user credentials** - used by an Oxygen XML Developer 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 Developer 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 Developer license server administrative interface.
 - b. **Standard user credentials** - used by an Oxygen XML Developer 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 Developer website](#).
3. Configure two Tomcat users:
 - a. One user with the role `user`, used by an Oxygen XML Developer 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 Developer 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 Developer 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 Developer 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.

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 Developer 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 Developer 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.

Replacing a Floating License Key in an HTTP Floating License Server

The following procedure assumes that your Oxygen XML Developer 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 Developer 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.

Getting More Information From the Report Page

You can access a license server activity report at

`http://hostName:port/oxygenLicenseServlet/license-servlet/report`.

It displays the following real time information:

- **License load** - A graphical indicator that shows how many licenses are available. When the indicator turns red, there are no more licenses available.
- **Floating license server status** - General information about the license server status, including the following information:
 - server start time
 - license count
 - rejected and acknowledged requests
 - average usage time
 - license refresh and timeout intervals
 - location of the license key
 - server version
- **License key information** - License key data, including the following information:
 - licensed product
 - registration name
 - company name
 - license category
 - number of floating users

- Maintenance Pack validity
- **Current license usage** - Lists all currently acknowledged users, including the following information:
 - user name
 - date and time when the license was granted
 - name and IP address of the computer where Oxygen XML Developer runs
 - MAC address of the computer where Oxygen XML Developer runs



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

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

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

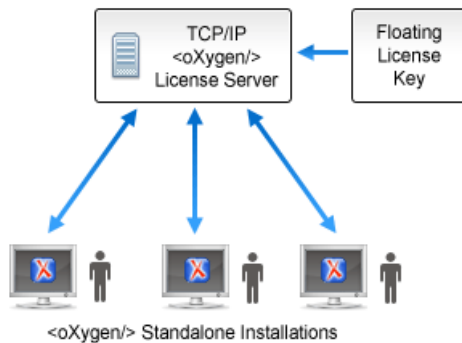


Figure 5: TCP Floating License Server (Process in Windows)

Installation Steps

1. Download the license server installation kit for Windows from the [Oxygen XML Developer 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.voptions` 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 Developer 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 Developer 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.

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 Developer floating license server to its newest version.

Follow this procedure:

1. Go to the Oxygen XML Developer website and download the [latest floating license server](#).
2. Run the installation kit.
3. Leave the default **Update the existing installation** option enabled. 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.

Replacing a Floating License Key in a TCP Floating License Server

The following procedure assumes that your Oxygen XML Developer 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.

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 an 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).

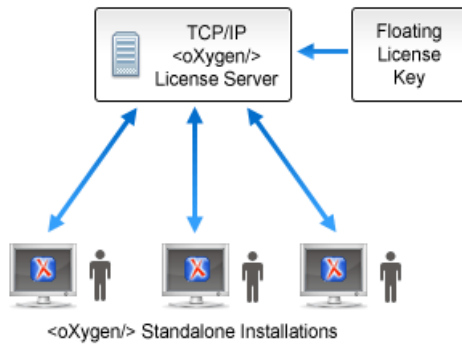


Figure 6: TCP Floating License Server (All-Platforms Distribution)

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 Developer 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 Developer instances. The default value is **12346**.

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.

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 Developer 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 Developer 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.

Replacing a Floating License Key in a TCP Floating License Server

The following procedure assumes that your Oxygen XML Developer 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.

Transferring or Releasing a License Key

If you want to transfer your Oxygen XML Developer license key to another computer (for example, if you are disposing of your old computer or transferring it to another person), or release a [floating license](#) so that someone else can use it, 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 (for the case of releasing a floating license and reverting 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 Developer.

Upgrading Oxygen XML Developer

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

Checking for New Versions of Oxygen XML Developer

Oxygen XML Developer checks for new versions automatically at start up. To disable this check, [open the Preferences dialog box \(Options > Preferences\)](#), go to **Global**, and uncheck **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 Developer, some data is preserved and some is overwritten. If there is a previous version of Oxygen XML Developer 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 [document type \(framework\) 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 [document type \(framework\) 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 Developer.
5. If you require a 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 Developer

Oxygen XML Developer provides an add-on mechanism that can automatically discover and install [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. The add-ons list contains the name, status, update version, Oxygen XML Developer 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, disable **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.

- By default, only the latest versions of the add-ons that are compatible with the current version of Oxygen XML Developer are displayed.
- 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 Developer [preferences directory](#).

Managing installed add-ons

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

- Go to **Help > Manage add-ons**
- The displayed dialog box presents a list of the available updates (compatible with the current version of Oxygen XML Developer) and with the already installed updates. Under the updates list, Oxygen XML Developer presents a short description of each update.
- Check for box 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 Developer 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 Developer version.

To watch a video demonstration about the add-ons support in Oxygen XML Developer, go to <http://www.oxygenxml.com/demo/AddonsSupport.html>.

Related information

[Pack and Deploy Plugins or Frameworks as Add-ons](#) on page 833

Uninstalling Oxygen XML Developer

Uninstalling the Oxygen XML Developer Standalone



CAUTION: The following procedure will remove Oxygen XML Developer 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 Developer.**

- Backup any data you want to keep from the Oxygen XML Developer installation folder.
- 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.
- If you want to remove the user preferences:
 - On Windows Vista/7/8/10**, remove the directory: `%APPDATA%\Roaming\com.oxygenxml.developer` (usually `%APPDATA%` has the value `[user-home-dir]\Application Data`). Note that this is directory is hidden.
 - On Windows XP**, remove the directory: `%APPDATA%\com.oxygenxml.developer` (usually `%APPDATA%` has the value `[user-home-dir]\Application Data`).

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

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

Command line options of the Oxygen XML Developer installer.

The Oxygen XML Developer installers for Windows and Linux create 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`:

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

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 Developer.	true / false. Default setting is true.
downloadResources	Allows Oxygen XML Developer to download resources (links to video demonstrations, webinars and upcoming events) from http://www.oxygenxml.com to populate the application welcome screen.	true / false. Default setting is true.

Command line parameters

The Oxygen XML Developer 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 <code>response.varfile</code> is specified using the <code>-varfile</code> option or individual settings are specified using - on Windows: <pre>oxygenDeveloper.exe -q</pre> - on Linux: <pre>oxygenDeveloper.sh -q</pre>
-overwrite	In unattended mode, the installer does not overwrite files with the same name if a previous version of the Oxygen XML Developer is installed in the same folder. The <code>-overwrite</code> parameter added after the <code>-q</code> parameter forces the overwriting of these files. - on Windows: <pre>oxygenDeveloper.exe -q -overwrite</pre> - on Linux: <pre>oxygenDeveloper.sh -q -overwrite</pre>
-console	To display a console for the unattended installation, add a <code>-console</code> parameter to the command line. - on Windows: <pre>start /wait oxygenDeveloper.exe -q -console</pre>  Note: The use of <code>start /wait</code> on Windows is required to make the installer run in the foreground. If you run it without <code>start /wait</code>, it will run in the background. - on Linux: <pre>oxygenDeveloper.sh -q -console</pre>
-varfile	Points to the location of a <code>response.varfile</code> to be used during an unattended installation. For example: - on Windows: <pre>oxygenDeveloper.exe -q -varfile response.varfile</pre> - on Linux: <pre>oxygenDeveloper.sh -q -varfile response.varfile</pre>
-V	Is used to define a variable to be used by an unattended installation. For example: - on Windows: <pre>oxygenDeveloper.exe -q -VusageDataCollector=false</pre> - on Linux: <pre>oxygenDeveloper.sh -q -VusageDataCollector=false</pre>

The Oxygen XML Developer 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.

Chapter 4

Configuring Oxygen XML Developer

Topics:

- [*Preferences*](#)
- [*Configuring Options*](#)
- [*Associating a File Extension with Oxygen XML Developer*](#)
- [*Configuring the Layout of the Views and Editors*](#)
- [*Configure Toolbars*](#)
- [*Scenarios Management*](#)
- [*Editor Variables*](#)
- [*Custom System Properties*](#)
- [*Localizing the User Interface*](#)
- [*Setting a Java Virtual Machine Parameter in the Launcher Configuration File / Start-up Script*](#)

A description of all of the options that allow you to configure Oxygen XML Developer.

This chapter presents all the user preferences and options that allow you to configure various features, or the application itself, and the editor variables that are available for customizing the user-defined commands.

Preferences

You can configure Oxygen XML Developer 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 box or expand/collapse the tree by using the  **Expand All** and  **Collapse All** buttons above the tree.

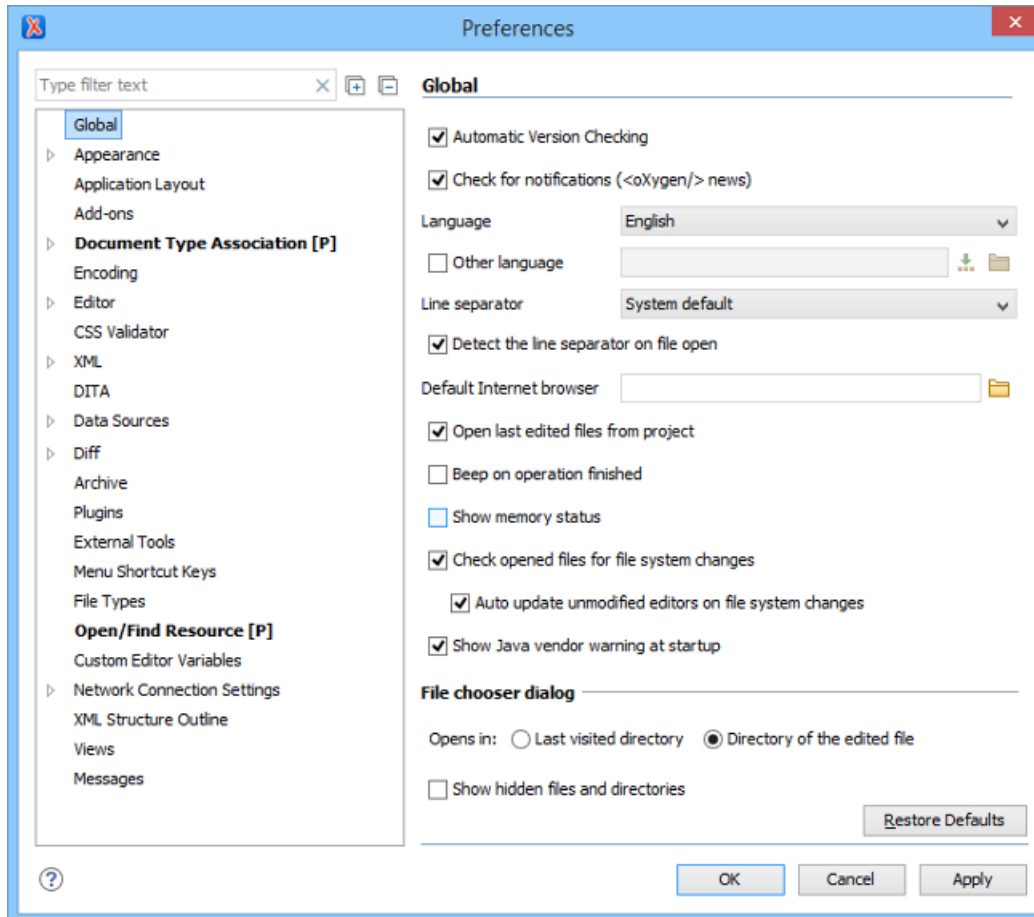


Figure 7: Preferences Dialog Box

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

Press  or **F1** for help on any 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 Developer. 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 enabled, Oxygen XML Developer will check for a new version on startup.
Check for notifications	If enabled, the application will check for various types of messages from the Oxygen XML Developer 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 Developer team will participate, and more. This option is enabled by default.
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 Developer 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 161. You can use this option to set the language of the user interface to a language that is not shipped with Oxygen XML Developer.  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, check 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 Developer 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 enabled, Oxygen XML Developer 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 Developer 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 memory Oxygen XML Developer 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 Developer uses too much memory. You can change the amount of memory available to Oxygen XML Developer by changing the parameters of the application launcher .

Check opened files for file system changes	When this option is selected, Oxygen XML Developer 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 Developer 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 Developer automatically updates unmodified editors if the edited file changes externally.
Show Java vendor warning at startup	If this option is selected, Oxygen XML Developer 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 Developer 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 Developer. 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 Developer 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 Developer. 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 enable 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 Developer.

Colors Preferences

Oxygen XML Developer 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 background - 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.
Foreground Colors	Status bar background - Background color of the status bar at the bottom of the editor.
	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.
Other Colors	Status bar foreground - Foreground color for the text in the status bar at the bottom of the editor.
	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 Developer allows you to choose the fonts to be used in the **Text**, **Design**, and **Grid** editor modes. 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.
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.

Application Layout Preferences

Oxygen XML Developer 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 displays views such as Project, Archive Browser, DITA Maps Manager, Outline, Attributes, Model, and Elements. • Basic - Only the Project view and the 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**, **Transformation Scenarios**, **XSLT/XQuery input** views are displayed.
- **XSLT development** - The **Project**, **Component Dependencies**, **Resource Hierarchy/Dependencies**, **Outline**, **Attributes**, **Model**, **XSLT/XQuery input**, **XPath 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 enabled, Oxygen XML Developer 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 enabled, Oxygen XML Developer saves layouts individually for each project. When you switch projects, the layout you last used for that project is loaded automatically.

Allow detaching editors from main window When this options is enabled, 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 Developer.

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

Add-ons Preferences

You can use add-ons to enhance the functionality of Oxygen XML Developer. 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 Developer 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 Developer 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 document type associations (frameworks), 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 association.

Edit Opens a [Document type configuration dialog box](#) that allows you to edit an existing association.



Note: If you try to edit an existing association type when you do not have write permissions to its store location, a dialog box will be shown asking if you want to extend the document type.

Duplicate Opens a [Document type configuration dialog box](#) that allows you to duplicate the configuration of an existing document type association.

Extend Opens a [Document type configuration dialog box](#) that allows you to extend an existing document type. 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 document type associations.

Enable DTD/XML Schema processing in document type detection When this option is enabled (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 Developer to correctly match the DITA customization.

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

Enable DTD/XML Schema caching When this option is enabled (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.

Locations Preferences

Oxygen XML Developer allows you to change the location where [document types \(frameworks\)](#) 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 either in the application's [.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) can be loaded from the following locations:

- **Internal preferences** - The document type configuration is stored in the application [Internal preferences](#).
- **Additional frameworks 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 *Document Type Association* (framework). 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 *Document Type Association*.
- **Priority** - Depending on the priority level, Oxygen XML Developer establishes the order in which the existing document type associations 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 *Document Type Associations* 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 document type association settings are 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*.

Association Rules Tab

By combining multiple association rules you can instruct Oxygen XML Developer to identify the type of a document. An Oxygen XML Developer *association rule* holds information about *Namespace*, *Root local name*, *File name*, *Public ID*, *Attribute*, and *Java class*. Oxygen XML Developer 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 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 Developer 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, 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 Developer 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, the  [Insert Editor Variables](#) button, or the browsing tools in the  **Browse** drop-down list.
 -  **Tip:** The path can also contain wildcards (for example, `${framework}/lib/*.jar`).
-  **Edit** Opens a dialog box that allows you to edit a resource in the **Classpath** tab. You can specify the path by using the text field, the  [Insert Editor Variables](#) button, or the browsing tools in the  **Browse** drop-down list.
 -  **Tip:** The path can also contain wildcards (for example, `${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 73

[Author Tab](#) on page 63

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 Developer 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, 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 enabled by default for DITA document types.
ignore CSSs from the associated document type	The CSS files set in the CSS tab 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 the CSS tab are merged with the CSS files specified in the document itself.

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 a dialog box that allows you to edit an 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.

The screenshot shows the 'Action' dialog box with the following details:

- ID:** bold
- Name:** \$\{i18n(bold)\}
- Menu access key:** B
- Description:** \$\{i18n(bold_description)\}
- Large icon (20x20):** /images/Bold20.png
- Small icon (16x16):** /images/Bold16.gif
- Shortcut key:** M1+8
- ☒ **Enable platform-independent shortcut keys**
- Operations:**
 - When this XPath expression is true (This XPath expression applies only to elements and attributes)
 - invoke the operation: nc.ecss.extensions.commons.operations.ToggleSurroundWithElementOperation
 - with the arguments:

Name	Description	Type	Value
element	The element to surround with.	Fragment	
- Operation priority:** Increase, Decrease
- ☐ **Evaluate activation XPath expressions even in read-only contexts**

Figure 8: 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 <code>\${i18n('key')}</code> <i>editor variable</i> to allow for multiple translations of the description.								
Large icon	Allows you to select an image for the icon that Oxygen XML Developer uses for the toolbar action.  Tip: A good practice is to store the image files inside the framework directory and use the <code>\${frameworks}</code> editor variable to make the image relative to the framework location. If the images are bundled in a <i>jar</i> archive (for instance, along with some Java operations implementation), it is convenient to reference the images by their relative path location in the <i>class-path</i>.								
Small icon	Allows you to select an image for the icon that Oxygen XML Developer uses for the contextual menu action.								
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 enabled, 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 enabled 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: Activation of Multiple Functions for Actions using XPath Expressions on page 66. The following options are available in this section: <table> <tr> <td>When this XPath expression is true</td><td>An XPath 2.0 expression that applies to elements and attributes. For more details see: Activation of Multiple Functions for Actions using XPath Expressions on page 66.</td></tr> <tr> <td>invoke the operation with the arguments</td><td>Specifies the invoked operation.</td></tr> <tr> <td>Edit</td><td>Specifies the arguments of the invoked operation.</td></tr> <tr> <td>Operation priority</td><td>Allows you to edit the arguments of the operation.</td></tr> </table> 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.	When this XPath expression is true	An XPath 2.0 expression that applies to elements and attributes. For more details see: Activation of Multiple Functions for Actions using XPath Expressions on page 66.	invoke the operation with the arguments	Specifies the invoked operation.	Edit	Specifies the arguments of the invoked operation.	Operation priority	Allows you to edit the arguments of the operation.
When this XPath expression is true	An XPath 2.0 expression that applies to elements and attributes. For more details see: Activation of Multiple Functions for Actions using XPath Expressions on page 66.								
invoke the operation with the arguments	Specifies the invoked operation.								
Edit	Specifies the arguments of the invoked operation.								
Operation priority	Allows you to edit the arguments of the operation.								

Evaluate activation If this checkbox is enabled, the action can be invoked even when the cursor is placed in a read-only XPath expressions location.
even in read-only
contexts

Activation of 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.

Two special XPath extension functions are provided: the `oxy:allows-child-element()` function that you can use to check whether or not an element is valid in the current context, considering the associated schema and the `oxy:current-selected-element()` function that you can use to get the currently selected element.

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

This extension function allows author actions to be available in a context only if the associated schema permits it.

The `oxy:allows-child-element()` 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:current-selected-element() Function

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

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

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

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 Developer 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.

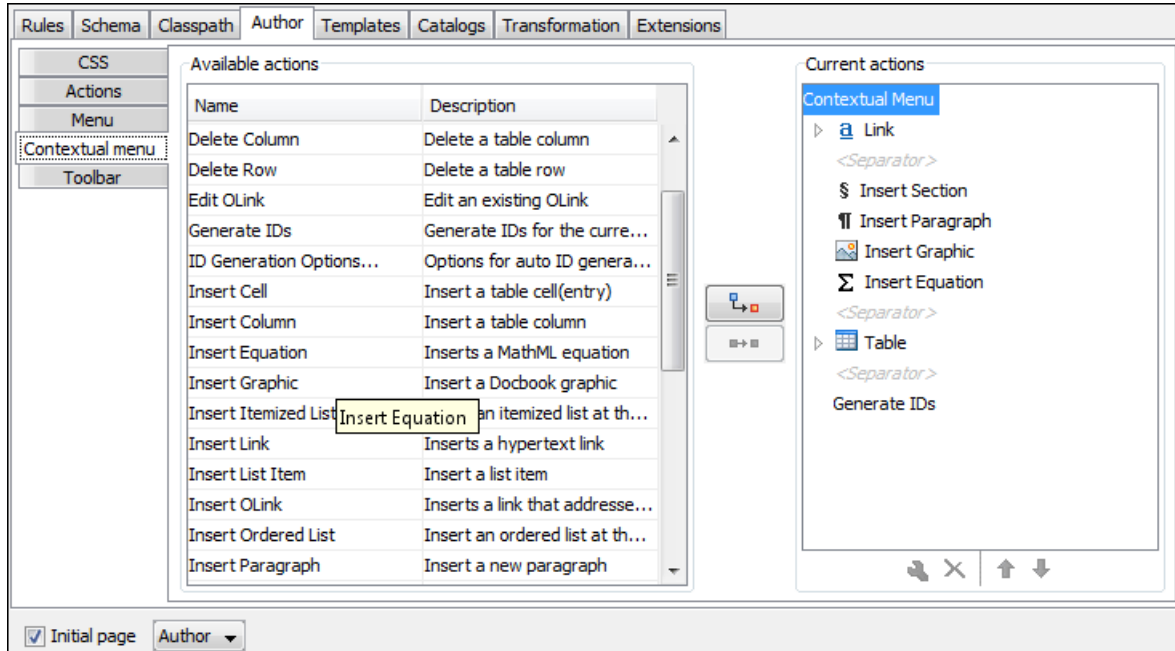


Figure 9: Contextual Menu Subtab

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).

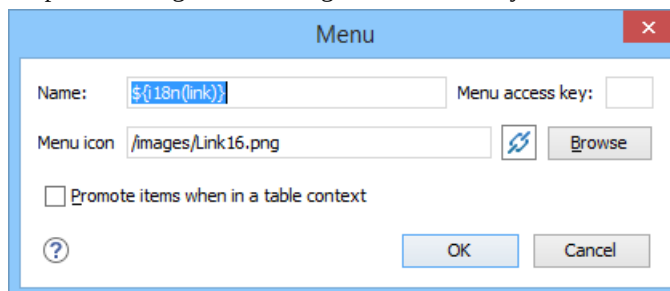


Figure 10: Menu Action Configuration Dialog Box

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 <code>\${i18n('key')}</code> 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 <i>Letter</i> 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 <i>Letter</i> 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 Developer uses for the container (submenu).
Promote items when in a table context	If this option is enabled, 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 Developer 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 Developer 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.

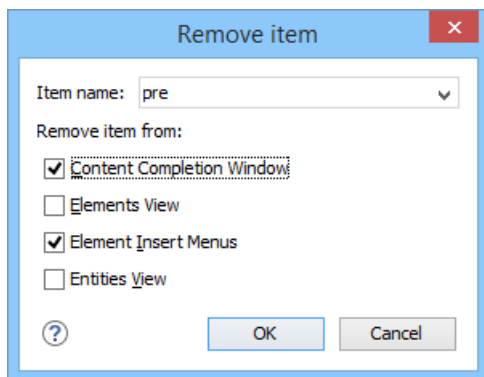


Figure 11: Remove Item Dialog Box

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.

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, 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, 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 Developer 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, 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. For more information, see the Creating New Transformation Scenarios on page 560 section.
 Edit	Opens the Edit scenario dialog box allowing you to edit the properties of the currently selected transformation scenario. For more information, see the Creating New Transformation Scenarios on page 560 section.
 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 are used and take precedence over the default scenarios defined for the particular framework. For more information on master files, see the [Defining Master Files at Project Level](#) on page 242 section or [Working with Modular XML Files in the Master Files Context](#) on page 308.

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 .
 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 <http://www.oxygenxml.com/InstData/Editor/SDK/javadoc/> contains details about how each API implementation functions.

Encoding Preferences

Oxygen XML Developer 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 Developer 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 Developer 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 Developer 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 Developer displays.
Show TAB/NBSP/EOL/EOF marks	Makes the <i>TAB/NBSP/EOL/EOF</i> 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 Developer will let you edit, but not save, a read-only file. If the option is disabled, a warning message is displayed when you try to edit a read-only file.
Display quick-assist and quick-fix side hints	Displays the <i>Quick Assist</i> icon (💡) and <i>Quick Fix</i> 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 , Design , Grid).

Print Preferences

Oxygen XML Developer lets you configure how files are printed out of the editor. Note that these settings cover how files are printed directly from Oxygen XML Developer 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 editor. These settings do not apply to the **Grid** and schema **Design** modes.

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 Developer 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**.

The initial edit mode can be one of the following:

- **Text**
- **Grid**
- **Design** (available only for the W3C XML Schema editor).

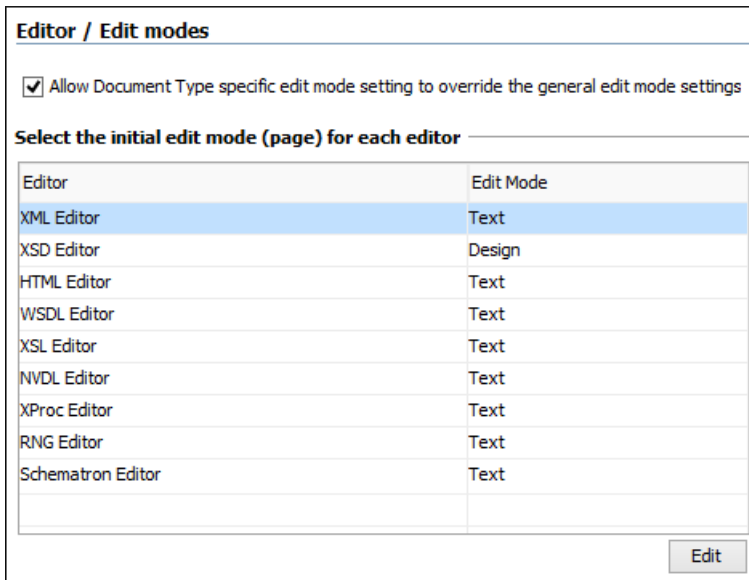


Figure 12: Edit Modes Preferences Page

Text Preferences

Oxygen XML Developer lets you configure how *text mode* editor appears. To configure the **Text** mode editor 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 enabled, the current line is highlighted with the foreground color specified with the color chooser.
Show line numbers	If enabled (default value), line numbers are shown in the editor panels and in the Results 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 enabled, 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 enabled, long lines are automatically wrapped in edited documents. The line wrap does not alter the document content since the application does not use <i>newline</i> characters to break long lines.
Cut / Copy whole line when nothing is selected	If enabled, <i>Cut</i> and <i>Copy</i> actions operate on the entire current line when nothing is selected in the editor.
Enable folding	If enabled (default value), the vertical stripe that holds the <i>folding markers</i> is displayed in Text mode.
Highlight matching tag	If enabled, when you place the cursor on a start or end tag, Oxygen XML Developer highlights the corresponding member of the pair. You can also customize the highlight color.
Lock the XML tags	If enabled, XML are locked and cannot be edited in Text mode.

Diagram Preferences

For certain XML languages, Oxygen XML Developer 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 Developer 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 <i>compact representation</i> of the grid is used: a child element is displayed beside the parent element. In the <i>non-compact representation</i> , 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. There is only one active cell at a time. 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.

Schema Design Preferences

Oxygen XML Developer 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 Developer displays the content of <code>xs:documentation</code> 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 Developer 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 Developer asks if you want to open the imported schema file.
Zoom	Allows you to set the default zoom level of the schema diagram.

Properties

Oxygen XML Developer 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.

Format Preferences

This preferences page contains various formatting options that influence editing and formatting in the **Text** mode.



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 Developer detects how a document is indented when it is opened. Oxygen XML Developer uses a heuristic method of detection by computing a weighted average indent value from the initial document content. You can disable 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 enabled, Oxygen XML Developer 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 enabled.

Use zero-indent, if detected

By default, if no indent was detected in the document, the fixed-size indent is used. Enable 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 enabled 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 Developer 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 enabled, the *indent size* is the number of SPACE characters used for indenting the text lines.

For additional information about changing the *indent size*, see the [Setting an Indent Size to Zero](#) on page 268 topic.

For information about when this setting is used, see the [When Indent Size and Line Width Settings are Used in Oxygen XML Developer](#) on page 80 section.

Hard line wrap (Limit to "Line width - Format and Indent")	If enabled, when typing content in the Text editing mode and the maximum line width is reached, a line break is automatically inserted.
Indent on enter	If disabled, when you press the Enter key to insert a line break in the Text editing mode, no 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.
Detect line width on open	If enabled, Oxygen XML Developer automatically detects the line width when the document is opened.
Format and indent the document on open	If enabled, an XML document is formatted and indented before opening it in Oxygen XML Developer.
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.



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 the [When Indent Size and Line Width Settings are Used in Oxygen XML Developer](#) on page 80 section.

Clear undo buffer before Format and Indent	The Format and Indent operation can be <i>undone</i> , but if used intensively, a considerable amount of the memory allocated for Oxygen XML Developer will be used for storing the undo states. If this option is selected, Oxygen XML Developer 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.

When Indent Size and Line Width Settings are Used in Oxygen XML Developer

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 enabled and the maximum line width is reached while editing in the **Text** mode.

To watch our video demonstration about the formatting options offered by Oxygen XML Developer, go to http://oxygenxml.com/demo/Autodetect_Formatting.html.

XML Formatting 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 enabled, 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 enabled:

```
<root> text <parent>
  <child/>
</parent>
</root>
```

Indent inline elements disabled:

```
<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 attr="v1"></a>`). When not enabled, the same operation represents an empty element in a more compact form (`<a attr="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.



Note: The XPath expressions can accept one 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.

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 <i>before</i> the element, <i>after</i> , 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 <i>Preserve space</i> elements (identified by the <code>xml:space</code> attribute set to <code>preserve</code> 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 enabled 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, disable this option.

Whitespaces Preferences

When Oxygen XML Developer formats and indents XML documents, a whitespace normalization process is applied, thus replacing whitespace sequences with single space characters. Oxygen XML Developer 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. Check 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.



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 Formatting 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 Formatting 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 enabled, 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 Properties Formatting Preferences

Oxygen XML Developer 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 enabled, the <i>class</i> body (including the curly brackets) is placed on a new line. Disabled by default.
Indent class content	When enabled, the <i>class</i> content is indented. Enabled by default.
Add space before the value of a CSS property	When enabled, whitespaces are added between the : (colon) and the value of a style property. Enabled by default.
Add new line between classes	If enabled, an empty line is added between two classes. Disabled by default.
Preserve empty lines	When enabled, the empty lines from the CSS content are preserved. Enabled by default.
Allow formatting embedded CSS	When enabled, CSS content that is embedded in XML is also formatted when the XML content is formatted. Enabled by default.

JavaScript Properties Formatting 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 enabled by default.
- **Allow formatting embedded JavaScript** - Applied only to XHTML documents, this option allows Oxygen XML Developer to format embedded JavaScript code, taking precedence over the [Schema aware format and indent](#) option. This option is enabled by default.

Content Completion Preferences

Oxygen XML Developer 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 enabled, Oxygen XML Developer automatically closes the last open tag when you type <code></</code> .
Automatically rename/delete/comment matching tags	If you rename, delete, or comment out a start tag, Oxygen XML Developer 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 Developer 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 Developer inserts the required elements specified in the DTD, XML Schema, or RELAX NG schema that is <i>associated with the edited XML document</i>. • Add optional content - If enabled, Oxygen XML Developer inserts the optional elements specified in the DTD, XML Schema, or RELAX NG schema. • Add first Choice particle - If enabled, Oxygen XML Developer inserts the first choice particle specified in the DTD, XML Schema, or RELAX NG schema.
Case sensitive search	When enabled, 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 enabled, Oxygen XML Developer 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 <i>Insert the required attributes option</i> is disabled.
Show all entities	Oxygen XML Developer 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, <code>&</code>).
Insert the required attributes	Oxygen XML Developer inserts automatically the required attributes taken from the DTD or XML Schema.
Insert the fixed attributes	If enabled, Oxygen XML Developer 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 enabled, Oxygen XML Developer 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 enabled, Oxygen XML Developer learns the attribute values used in a document.
Learn on open document	Oxygen XML Developer automatically learns the document structure when the document is opened.
Learn words (Dynamic Abbreviations, available on CTRL-SPACE (COMMAND-SPACE on OS X))	When enabled, Oxygen XML Developer 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.

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 Developer 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 Developer 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 Developer 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 disabled 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 Developer 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 enabled, Oxygen XML Developer 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 disabled, 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 256

XSL Preferences

XSL 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 Developer can suggest elements from these vocabularies. To configure the XSL content completion options, [open the Preferences dialog box \(Options > Preferences\)](#) and go to **Editor > Content Completion > XSL**.

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 Developer analyzes the namespaces declared in the root element to find an appropriate schema.

If the detection fails, Oxygen XML Developer 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 Developer 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 Developer provides content completion assistance when you are writing an XML Schema (XSD). To configure XSD preferences, [open the Preferences dialog box \(Options > Preferences\)](#) and go to **Editor > Content Completion > XSD**. The options in this preferences page define what elements are suggested by the **Content Completion Assistant**, in addition to the ones from the XML Schema (defined by the `xs:annotation/xs:appinfo` elements).

The following options are available:

- **None** - The **Content Completion Assistant** offers only the XML Schema schema information.
- **ISO Schematron** - The **Content Completion Assistant** includes ISO Schematron elements in `xs:appinfo`.
- **Schematron 1.5** - The **Content Completion Assistant** includes Schematron 1.5 elements in `xs:appinfo`.
- **Other** - The **Content Completion Assistant** includes in `xs:appinfo` elements from an XML Schema identified by a URL.

JavaScript Preferences

Oxygen XML Developer 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 Developer 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 Developer can also use JavaScript libraries to when making suggestions. List the paths (URIs) of any JavaScript files you want Oxygen XML Developer 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 Developer supports syntax highlighting of XML in the [Text mode](#) editor, DTD, Relax NG (XML and Compact Syntax), Java, JavaScript / JSON, Ant, PHP, CSS, XQuery, C++, C, Perl, Properties, SQL, Shell and Batch documents.

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 language. 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 used also in XSD, XSL, RNG documents. The **Preview** area has separate tabs for XML, XSD, XSL, RNG.

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 Developer 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, enable the **Draw only the prefix with a separate color** option.

Open / Save Preferences

Oxygen XML Developer 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 enabled and you open a file from the local file system or a shared network drive, Oxygen XML Developer locks the file for the current user and the file cannot be modified by other users while the lock exists. However, other users are offered the possibility of editing the file without overwriting it. If they decide to continue, they will be asked to save the changes in a different file or drop them. Newly created files are <i>locked</i> when you first save them. If you enable this option with files already opened in Oxygen XML Developer, it will <i>lock</i> all the currently opened files. If you disable this option with files already opened, it will unlock them by deleting the corresponding .lock files.
Support for Special Characters section	When bidirectional text, Asian languages, or other special characters are detected	You can choose how you want Oxygen XML Developer 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 enabled, bidirectional editing is disabled for files exceeding the specified size, even if the Enable support for special characters option is selected.
Save section	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 enabled, it saves the content to a temporary file and if the save is unsuccessful, the editor preserves its unsaved state status.
	Make backup copy on save (only for local files)	If enabled, 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 Backup files extension text field.

	Enable automatic save	If enabled, your documents are automatically saved after a preset time interval that is specified in the Automatic save interval (minutes) 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 enabled, Oxygen XML Developer runs a validation that checks your document for errors before saving it.
	Save all files before calling external tools	If enabled, 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 Developer will warn you if you open a file that is bigger than the specified size.
	Optimize loading for documents with lines longer than (Characters)	<i>Line wrap</i> 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 the Editing Documents with Long Lines topic.
	Show warning when loading documents with long lines	When selected, Oxygen XML Developer will 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 the Editing Documents with Long Lines on page 513 topic.
	Clear undo buffer on save	If selected, Oxygen XML Developer 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 <i>out of memory</i> 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 <i>apple.awt.use-file-dialog-packages</i> property to <i>true</i> or <i>false</i> in the <i>Info.plist</i> descriptor file of the Oxygen XML Developer application:

```
<key>apple.awt.use-file-dialog-packages</key>
<string>>false</string>
```

Save Hooks Preferences

Oxygen XML Developer 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 can be enabled or disabled:

Automatically compile LESS to CSS when saving If enabled, when you save a LESS file it will automatically be compiled to CSS (disabled by default).



Important: If this option is enabled, 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 Developer 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 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 enabled, 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 enabled, 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 disabled 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:

Parameter	
url	Format: <code>\${ask('message', url, 'default_value')}</code>
	Description: Input is considered a URL. Oxygen XML Developer checks that the provided URL is valid.
	Example: • <code>\${ask('Input URL', url)}</code> - The displayed dialog box has the name Input URL. The expected input type is URL. • <code>\${ask('Input URL', url, 'http://www.example.com')}</code> - The displayed dialog box has the name Input URL. The expected input type is URL. The input field displays the default value <code>http://www.example.com</code>.
password	Format: <code>\${ask('message', password, 'default')}</code>
	Description: The input is hidden with bullet characters.
	Example: • <code>\${ask('Input password', password)}</code> - The displayed dialog box has the name 'Input password' and the input is hidden with bullet symbols. • <code>\${ask('Input password', password, 'abcd')}</code> - The displayed dialog box has the name 'Input password' and the input hidden with bullet symbols. The input field already contains the default <code>abcd</code> value.
generic	Format: <code>\${ask('message', generic, 'default')}</code>
	Description: The input is considered to be generic text that requires no special handling.
	Example: • <code>\${ask('Hello world!')}</code> - The dialog box has a Hello world! message displayed. • <code>\${ask('Hello world!', generic, 'Hello again')}</code> - The dialog box has a Hello world! message displayed and the value displayed in the input box is 'Hello again!'.

Parameter	
relative_url	Format: <code>\${ask('message', relative_url, 'default')}</code> Description: Input is considered a URL. Oxygen XML Developer tries to make the URL relative to that of the document you are editing.  Note: If the <code>\$ask</code> editor variable is expanded in content that is not yet saved (such as an <i>untitled</i> file, whose path cannot be determined), then Oxygen XML Developer will transform it into an absolute URL. Example: <code>\${ask('File location', relative_url, 'C:/example.txt')}</code> - 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: <code>\${ask('message', combobox, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default')}</code> Description: Displays a dialog box that offers a drop-down menu. The drop-down menu is populated with the given <code>rendered_value</code> values. Choosing such a value will return its associated value (<code>real_value</code>).  Note: The 'default' parameter specifies the default selected value and can match either a key or a value. Example: <code>\${ask('Operating System', combobox, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'osx')}</code> - 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 <code>osx</code> key. However, the same result could be obtained if the default value is indicated by <code>Mac OS X</code>, as in the following example: <code>\${ask('Operating System', combobox, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'Mac OS X')}</code> <code>\${ask('Mobile OS', combobox, ('win':'Windows Mobile';'ios':'iOS';'and':'Android'), 'Android')}</code>

Parameter	
editable_combobox	Format: <code>\${ask('message', editable_combobox, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default'))}</code>
	Description: Displays a dialog box that offers a drop-down menu with editable elements. The drop-down menu is populated with the given <code>rendered_value</code> values. Choosing such a value will return its associated real value (<code>real_value</code>) 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.
radio	Example: <code>\${ask('Operating System', editable_combobox, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'osx'))}</code> - 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.
	Format: <code>\${ask('message', radio, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default'))}</code>
	Description: Displays a dialog box that offers a series of radio buttons. Each radio button displays a 'rendered_value' and will return an associated <code>real_value</code> .
	 Note: The 'default' parameter specifies the default selected value and can match either a key or a value.
	Example: <code>\${ask('Operating System', radio, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'osx'))}</code> - 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}` - Current project folder as file path. Usually the current folder selected in the **Project** View.
- `${oxygenInstallDir}` - Oxygen XML Developer installation folder as file path.
- `${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 261

Document Templates Preferences

Oxygen XML Developer 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 Developer 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 Developer 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, 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 Developer looks in these directories by using the **Up** and **Down** buttons.

Spell Check Preferences

Oxygen XML Developer provides support for spell checking in the [text](#) editing mode. To configure the **Spell Check** options, [open the Preferences dialog box \(Options > Preferences\)](#) and go to **Editor > Spell Check**.

The following options are available:

Spell checking engine	Oxygen XML Developer includes two spell check engines, <i>Hunspell</i> and <i>Java spell checker</i> . The two engines come with different dictionaries. When you select an engine here, the list of languages in the Default language option changes based on the available dictionaries for the chosen engine.
Automatic spell check	This option is disabled by default. When enabled, Oxygen XML Developer 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 spelled 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 <code>lang</code> or <code>xml:lang</code> 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 <code>lang</code> and <code>xml:lang</code> attributes are missing, the selection in the Default language list is used. • Do not check - If the <code>lang</code> and <code>xml:lang</code> 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 <i>etc.</i> or <i>i.e.</i>). 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, <i>SpellChecker</i>). Ignore acronyms Available only for the <i>Hunspell spell checker</i>. 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, <i>b2b</i>). 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, <i>www.oxygenxml.com</i> or <i>c:\boot.ini</i>). 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 general prefixes Available only for the <i>Java spell checker</i>. When selected, a word formed by concatenating a recognized prefix and a legal word is accepted. For example if <i>mini-</i> is a registered prefix, the spell check engine accepts the word <i>mini-computer</i>. Allow file extensions When selected, the spell checker accepts any word ending with recognized file extensions (for example, <i>myfile.txt</i> or <i>index.html</i>).

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: <code>/a/*/b</code>.

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 and term lists (`.tdi` files) that Oxygen XML Developer uses and to choose where to save new learned words.

The following options are valid when Oxygen XML Developer 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 Developer 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.  Note: 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 into generic ones (for example, <code>en_UK.dic</code> from the default location is merged with <code>en_US.dic</code> from a custom location, and the result is that a third file is created for a generic dictionary called <code>en.dic</code>). However, if there is already a generic dictionary (for example, <code>en.dic</code>) saved in either the default or custom location, the other specific dictionaries (for example, <code>en_UK.dic</code> and <code>en_US.dic</code>) will not be merged and the existing generic dictionary will simply be used. Also, 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 Developer 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 tasks

[Add Dictionaries for the Hunspell Checker](#) on page 508

Related information

[Adding Term Lists](#) on page 509

Document Checking Preferences

To configure the **Document Checking** (validation) options, [open the Preferences dialog box \(Options > Preferences\)](#) and go to **Editor > Document Checking**. This preferences 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 editor panel and on the

stripe that is displayed at the right side of 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 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 enabled, the current validation error or warning is always visible in the message line at the bottom of the editor panel. This is useful when Automatic validation is enabled 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 Developer.
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](#).

Mark Occurrences Preferences

To configure the **Mark Occurrences** options, [open the Preferences dialog box \(Options > Preferences\)](#) and go to **Editor > Mark Occurrences**:

The following preferences are available in this preferences page:

- **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 and WSDL files** - Activates the [Highlight Component Occurrences](#) feature in XSD and 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** - Color used to highlight the component declaration.
- **Reference highlight color** - Color used to highlight 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.

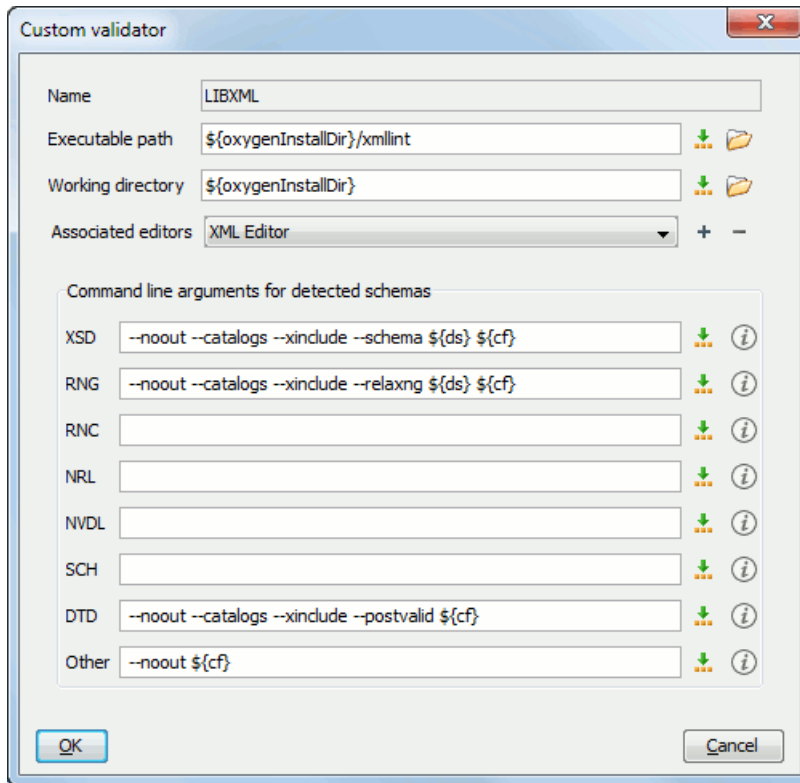


Figure 13: Custom Validator Dialog Box

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: • <i>\${cf}</i> - Current file as file path, that is the absolute file path of the current edited document. • <i>\${currentFileURL}</i> - Current file as URL, that is the absolute file path of the current edited document represented as URL. • <i>\${ds}</i> - The path of the detected schema as a local file path for the current validated XML document. • <i>\${dsu}</i> - The path of the detected schema as a URL for the current validated XML document.

Related information

[Editor Variables](#) on page 154

Increasing the stack size for validation engines

To prevent the appearance of a *StackOverflowException*, 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 $1 \times 1024 \times 1024 = 1048576$.
- 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 in the Launcher Configuration File / Start-up Script](#) on page 162

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 Developer:

- **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 and the CSS extensions specific for Oxygen. 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 (applies also to content completion)** - If checked, Oxygen XML Developer 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.
 - -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 the **XML Catalog** options, [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 enabled, Oxygen XML Developer 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 namespaces through URI mappings for XML Schema

If selected, the target namespace of the imported XML Schemas 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.

Use default catalog

If this option is enabled and Oxygen XML Developer 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 Developer Determines which Catalog to Use](#) on page 312.

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, 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, 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 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 XML catalog 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

<http://xerces.apache.org/xml-commons/components/resolver/resolver-article.html#ctrlresolver>

[Working with XML Catalogs](#) on page 310

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 checked, the XInclude support in Oxygen XML Developer is turned on for validation and transformation of XML documents.
Base URI fix-up	According to the specification for XInclude, processors must add an <code>xml:base</code> 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 <code>xml:base</code> attributes to appear on elements that might be included from different base URIs. If the addition of <code>xml:base</code> and / or <code>xml:lang</code> is not desired by your application, you can disable this option.
Language fix-up	The processor will preserve language information on a top-level included element by adding an <code>xml:lang</code> attribute if its include parent has a different [language] property. If the addition of <code>xml:lang</code> is undesired by your application, you can disable the language fix-up.
DTD post-validation	Enable this option to validate an XML file against the associated DTD, after all the content merged to the current XML file using <code>XInclude</code> was resolved. If you disable this option, the current XML file is validated against the associated DTD before all the content merged to the current XML file using <code>XInclude</code> 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	Allows you to select the default validation engine to be used for XML Schema. You can choose Xerces or Saxon EE .

**Schema
validation
engine**

Xerces **Enable full schema constraint checking**
validation (<http://apache.org/xml/features/validation/schema-full-checking>)
features
section

Sets the *schema-full-checking* feature to `true` enables a full schema validation of parsed XML documents against a schema. While the XML Schema validator is used while the document is parsed.

Enable honour all schema location feature
(<http://apache.org/xml/features/honour-all-schema-location>)

Sets the *honour-all-schema-locations* feature to `true` files that are used for validation. This option is the first XML Schema file encountered. Schema information is taken into

Enable full XPath 2.0 in assertions and alternative types
(<http://apache.org/xml/features/validation/cta-full-xpath-checking>)

When enabled (value), you can use XPath support and alternative types. Otherwise, support of Xerces engine

Assertions can see comments and processing instructions
(<http://apache.org/xml/features/validation/assert-comments-and-pi-checking>)

Controls whether or not comment and processing instructions are visible to the XPath engine for defining XSD 1.1.

Saxon **Multiple schema imports**
EE
validation
features
section

Assertions can see comments and processing instructions

Forces `xs:import` to fetch the referenced schema document. By default, the engine 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 its URI is the same as a schema document already loaded.

Controls whether or not comments and processing instructions are visible to the XPath engine used to define an assertion. By default, they are not made visible (unlike Saxon 9).

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: Enabling 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 check this option for faster ISO Schematron validation.
	Allow foreign elements (allow-foreign)	Enables support for <code>allow-foreign</code> 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 enabled, Saxon EE is used for <code>xslt2/xslt3</code> query binding. If this option is disabled, 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 enabled 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 checked, all elements are generated, including the optional ones (having the <code>minOccurs</code> attribute set to 0 in the schema).
Generate optional attributes	When checked, all attributes are generated, including the optional ones (having the <code>use</code> attribute set to <code>optional</code> 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 XML Instances Generator preferences page). Note that type restrictions are ignored when this option is enabled. 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 XML Instances Generator 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 <code>minOccurs</code> and <code>maxOccurs</code> facets defined in the XML Schema. • If the value set here is between <code>minOccurs</code> and <code>maxOccurs</code>, then that value is used. • If the value set here is less than <code>minOccurs</code>, then the <code>minOccurs</code> value is used. • If the value set here is greater than <code>maxOccurs</code>, then <code>maxOccurs</code> 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 <code>xs:alternative</code> 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 <code>xs:choice</code> or <code>substitutionGroup</code> elements. The possible strategies are: • First - The first branch of <code>xs:choice</code> or the head element of <code>substitutionGroup</code> is always used. • Random - A random branch of <code>xs:choice</code> or a substitute element or the head element of a <code>substitutionGroup</code> is used.
Generate the other options as comments	If enabled, generates the other possible choices or substitutions (for <code>xs:choice</code> and <code>substitutionGroup</code>). 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 checked, the value of an element or attribute starts with the name of that element or attribute. For example, for an <code>a</code> element the generated values are: <code>a1</code> , <code>a2</code> , <code>a3</code> , and so on. If not checked, the value is the name of the type of that element / attribute (for example: <code>string</code> , <code>decimal</code> , 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 415

XProc Engines Preferences

Oxygen XML Developer includes a built-in XProc engine called *Calabash*. You can add or configure external XProc engines by using the **XProc Engines** 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.

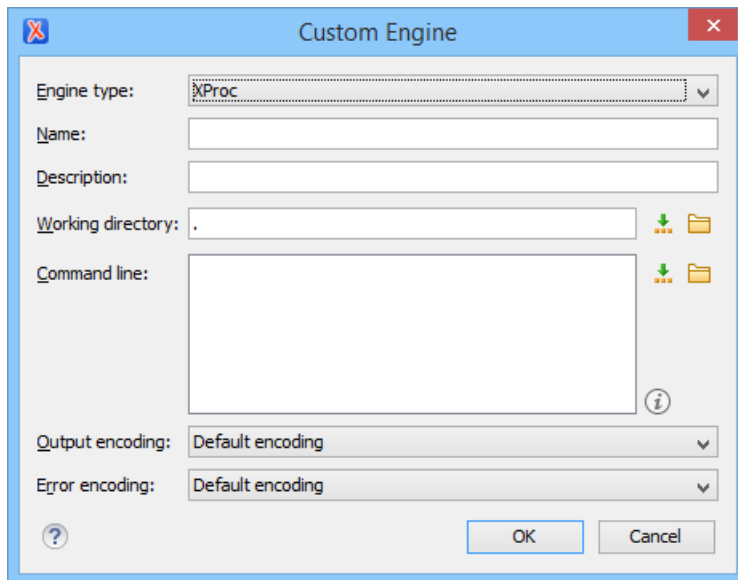


Figure 14: Creating an XProc external engine

The following options can be configured 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 the **XSLT/FO/XQuery** options, *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 Developer offers the possibility of using an XSLT transformer implemented in Java (other than the XSLT transformers that come bundled with Oxygen XML Developer). To use another XSLT transformer, specify the name of the transformer factory class. Oxygen XML Developer 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 <code>TransformerFactory</code> 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>.

Saxon6 Preferences

To configure the **Saxon 6** options, *open the Preferences dialog box (Options > Preferences)* and go to **XML > XSLT/FO/XQuery > XSLT > Saxon > Saxon 6**.

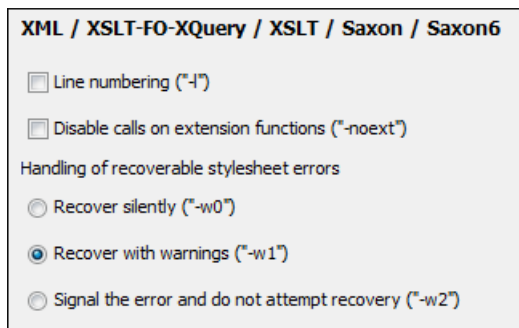


Figure 15: Saxon 6 XSLT Preferences Panel

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 enabled, external function calls are not allowed. Checking this 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**.

Oxygen XML Developer allows you to configure the following XSLT options for the Saxon 9.6.0.7 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE):

Use a configuration file ("-config")	Sets a Saxon 9.6.0.7 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 <i>XSLT Debugger</i> to <i>step into</i> 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 <code>document()</code> 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 <i>strip whitespaces</i> operation is handled. You can choose one of the following values: • All ("all") - Strips <i>all</i> whitespace text nodes from source documents before any further processing, regardless of any <code>xml:space</code> attributes in the source document. • Ignore ("ignorable") - Strips all <i>ignorable</i> whitespace text nodes from source documents before any further processing, regardless of any <code>xml:space</code> 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 <i>no</i> 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.

The following options are available for Saxon 9.6.0.7 Professional Edition (PE) and Enterprise Edition (EE) only:

Allow calls on extension functions ("-ext")	If checked, 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 <code>http://[URL]</code>). 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 9.6.0.7 processors.  Note: Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Developer 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.

The options available specifically for Saxon 9.6.0.7 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 <code>document()</code> or similar functions. It can have the following values: • Schema validation ("strict") - This mode requires an XML Schema and enables parsing the source documents with strict schema-validation enabled. • Lax schema validation ("lax") - If an XML Schema is provided, this mode enables 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. Enabling 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 enable 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**.

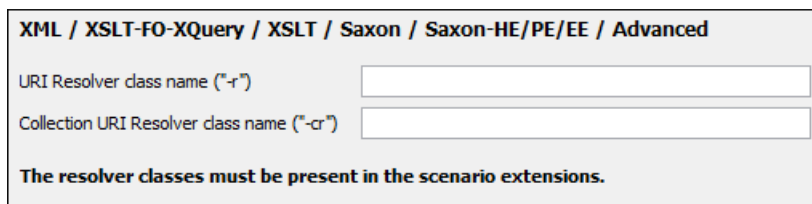


Figure 16: Saxon HE/PE/EE XSLT Advanced Preferences Panel

You can configure the following advanced XSLT options for the Saxon 9.6.0.7 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 9.6.0.7 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 9.6.0.7 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 options of the XSLTProc processor are the same as the ones available in the command line:

- **Enable XInclude processing** - If checked, XInclude references will be resolved when XSLTProc is used as transformer in *XSLT transformation scenarios*.
- **Skip loading the document's DTD** - If checked, the DTD specified in the DOCTYPE declaration will not be loaded.

- **Do not apply default attributes from document's DTD** - If checked, 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 checked, 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 checked, the transformation will output detailed status messages about the transformation process in the **Warnings** view.
- **Show version of libxml and libxslt used** - If checked, Oxygen XML Developer will display in the **Warnings** view the version of the **libxml** and **libxslt** libraries invoked by XSLTProc.
- **Show time information** - If checked, the **Warnings** view will display the time necessary for running the transformation.
- **Show debug information** - If checked, the **Warnings** view will display debug information about what templates are matched, parameter values, and so on.
- **Show all documents loaded during processing** - If checked, Oxygen XML Developer will display in the **Warnings** view the URL of all the files loaded during transformation.
- **Show profile information** - If checked, Oxygen XML Developer 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 checked, Oxygen XML Developer 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 checked, 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 checked, 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 checked, 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 checked, the resolution is disabled.
Strip non-significant whitespaces	If checked, strips non-significant white space from the input XML document during the load phase. Enabling this option can lower memory usage and improve transformation performance while, in most cases, creating equivalent output.
Show time information	If checked, 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 checked, XInclude references will be resolved when MSXML.NET is used as the transformer in the <i>XSLT transformation scenario</i> .
Validate documents during parse phase	If checked, 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 checked, the resolution is disabled.
Strip non-significant whitespaces	If checked, strips non-significant white space from the input XML document during the load phase. Enabling this option can lower memory usage and improve transformation performance while, in most cases, creating equivalent output.
Show time information	If checked, 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 checked, 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 <i>exsl:document extension element</i> .
Use named URI resolver class	This option allows you to specify a custom URI resolver class to resolve URI references in <code>xsl:import</code> and <code>xsl:include</code> instructions (during XSLT stylesheet loading phase) and in <code>document ()</code> 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 <i>Use named URI resolver class option</i> specifies a partially or fully qualified URI resolver class name (for example, <code>Acme.Resolvers.CacheResolver</code>). Such a name requires additional assembly specification using this option or the <i>Assembly GAC name for URI resolver class option</i> , but fully qualified class name (which always includes an assembly specifier) is <i>all-sufficient</i> . See MSDN for more info about <i>fully qualified class names</i> .
Assembly GAC name for URI resolver class	This option specifies partially or fully qualified name of the assembly in the <i>global assembly cache</i> (GAC) where the specified resolver class can be found. See MSDN for more info about <i>partial assembly names</i> .
List of extension object class names	This option allows to specify <i>extension object</i> 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 <i>plugged-in</i> 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 <i>username:password@domain</i> 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 <i>username:password@domain</i> 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 Developer 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 enabled, Oxygen XML Developer 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 Developer allows you to configure the following XQuery options for the Saxon 9.6.0.7 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE):

Use a configuration file ("-config")	Sets a Saxon 9.6.0.7 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 <i>strip whitespaces</i> operation is handled. You can choose one of the following values: • All ("all") - Strips <i>all</i> whitespace text nodes from source documents before any further processing, regardless of any <code>xml:space</code> attributes in the source document. • Ignore ("ignorable") - Strips all <i>ignorable</i> whitespace text nodes from source documents before any further processing, regardless of any <code>xml:space</code> 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 <i>no</i> 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 enabled (default value), Saxon runs the XQuery transformation with the XQuery 3.0 support.

The following option is available for Saxon 9.6.0.7 Professional Edition (PE) and Enterprise Edition (EE) only:

Allow calls on extension functions ("-ext")	If checked, calls on external functions are allowed. Checking 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 9.6.0.7 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 <code>document()</code> or similar functions. It can have the following values: • Schema validation ("strict") - This mode requires an XML Schema and enables parsing the source documents with strict schema-validation enabled. • Lax schema validation ("lax") - If an XML Schema is provided, this mode enables 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. Enabling 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 enable 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 checked, backup versions for any XML files updated with an XQuery Update are generated. This option is available when the **Enable XQuery update** option is enabled.

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 17: Saxon HE/PE/EE XQuery Advanced Preferences Panel

The advanced XQuery options that can be configured for the Saxon 9.6.0.7 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 9.6.0.7 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 9.6.0.7 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 checked, the debugger presents the output of `xsl:result-document` instructions into the debugger output view.

Infinite loop detection

Enable this option to receive notifications when an infinite loop occurs during transformation.

Enable Saxon optimizations

This option is disabled by default and this means that the optimization level for the debugging process is set to zero. If it is enabled, 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 <code>xsl:template</code> 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 Developer 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 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 114).

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	<i>Hotspots</i> are ignored below this specified amount (in milliseconds). For more information, see Hotspots View on page 789.
Ignore invocation less than	<i>Invocations</i> are ignored below this specified amount (in microseconds). For more information, see Invocation Tree View on page 788.
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 Developer 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 Developer 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 Developer 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.

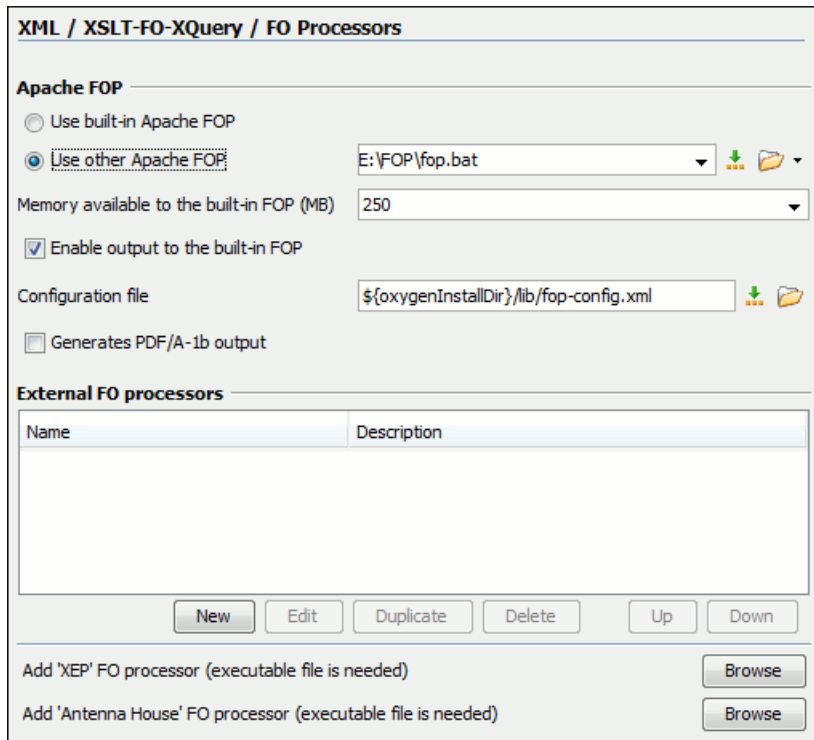


Figure 18: FO Processors Preferences Page

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 Developer to use the built-in Apache FO processor.
Use other Apache FOP	Instructs Oxygen XML Developer 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 Developer 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 <i>true type</i> 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(AHF62_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.

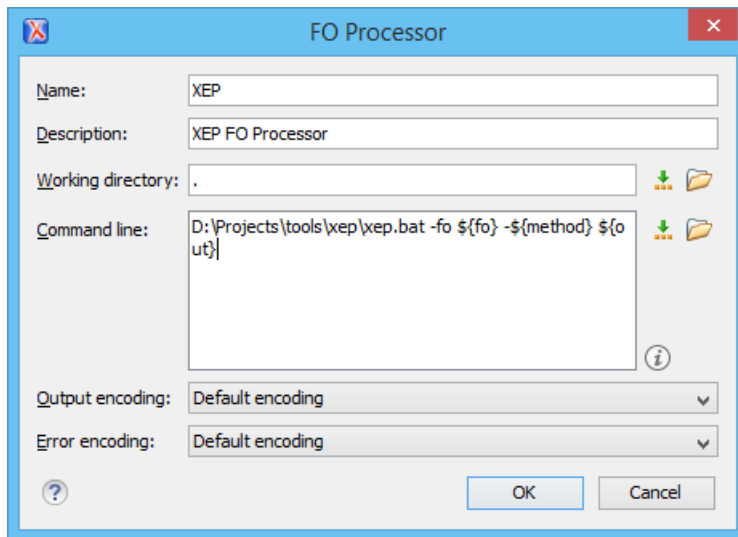


Figure 19: External FO Processor Configuration Dialog Box

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 Developer 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 Developer installation directory.
- **\${oxygenInstallDir}** - The Oxygen XML Developer 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 Developer 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 Developer window.

XPath Preferences

To configure XPath options, [open the Preferences dialog box \(Options > Preferences\)](#) and go to **XML > XSLT/FO/XQuery > XPath**.

Oxygen XML Developer allows you to customize the following options:

Unescape XPath expression

When enabled, 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 with:

```
//varlistentry[starts-with(@os, 's')]
```

Multiple XPath results Enable this option to display the results of an XPath expression in separate tabs in the [Results View](#) on page 188.

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 Developer 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 Developer 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 Developer considers unprefix 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 unprefix 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

You can configure and run XSLT and XQuery transformations with processors other than *the ones which come with the Oxygen XML Developer distribution*.



Note:

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.

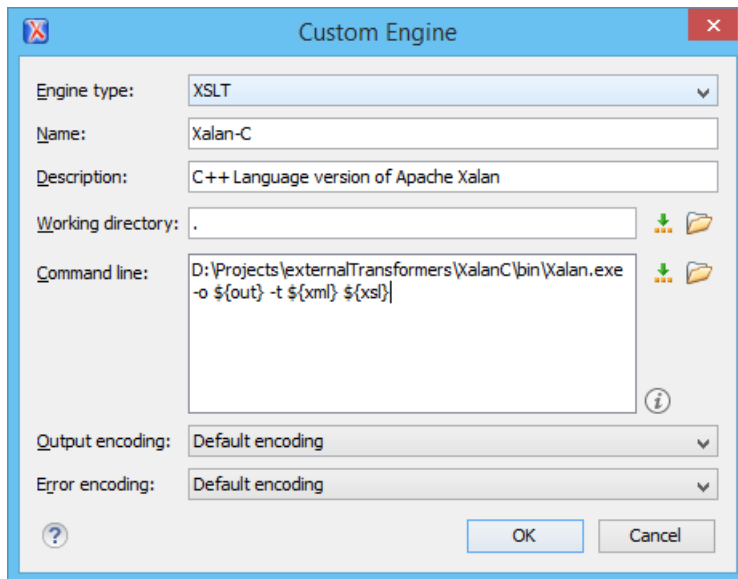


Figure 20: Parameters of a Custom Engine

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 Developer install directory.

Command line	The command line that must be executed by Oxygen XML Developer 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 Developer 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 Developer 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 checked, 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 checked, null values from a database column are imported as empty elements.
Escape XML content	Enabled by default, this option instructs Oxygen XML Developer to escape the imported content to an XML-safe form.
Add annotations for generated XML Schema	If checked, 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 <code>xs:string</code>. • XML Schema date format - The XML Schema-specific format ISO8601 is used for imported date / time data (<code>yyyy-MM-dd'T'HH:mm:ss</code> for <code>datetime</code>,

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`.

Date / Time Patterns Preferences

Table 2: 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*

TwoDigitHours must be between 00 and 23.

XML Signing Certificates Preferences

Oxygen XML Developer 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*:

The screenshot shows a dialog box titled "Certificates for Signing XML Documents". It has five input fields: "Keystore type" (a dropdown menu currently showing "JKS"), "Keystore file" (a text box with a file icon button to its right), "Keystore password" (a text box), "Certificate alias" (a text box), and "Private key password" (a text box). At the bottom right of the dialog is a "Validate" button.

Figure 21: Certificates Preferences Panel

- **Keystore type** - The type of *keystore* that Oxygen XML Developer 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 Developer 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. The builtin DITA OT 1.8 directory is: <code>[OXYGEN_INSTALL_DIR]/frameworks/dita/DITA-OT.</code> 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.x. This also gives you access to DITA 1.3 file templates when you create new documents from templates. The default DITA OT directory is: <code>[OXYGEN_INSTALL_DIR]/frameworks/dita/DITA-OT2.x.</code> 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.
Prefer navigation title for topicref rendering	If enabled and there is a <code>navtitle</code> attribute set on a <code>topicref</code>, then the <code>navtitle</code> 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, 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 enabled, the <code>format</code> attribute will always be automatically populated with a detected value. • Scope - If enabled, the <code>scope</code> attribute will always be automatically populated with a detected value. • Type - If enabled, the <code>type</code> attribute will always be automatically populated with a detected value. • Navigation title - If enabled, the <code>navtitle</code> 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 enabled, the <code>format</code> attribute will always be automatically populated with a detected value. • Scope - If enabled, the <code>scope</code> attribute will always be automatically populated with a detected value. • Type - If enabled, the <code>type</code> attribute will always be automatically populated with a detected value.
Use 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 ". " instead of <code>#./elementId</code>. For more information, see https://www.oxygenxml.com/dita/1.3/specs/index.html#archSpec/base-uri-based-addressing.html.

- 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.

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 127.

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.

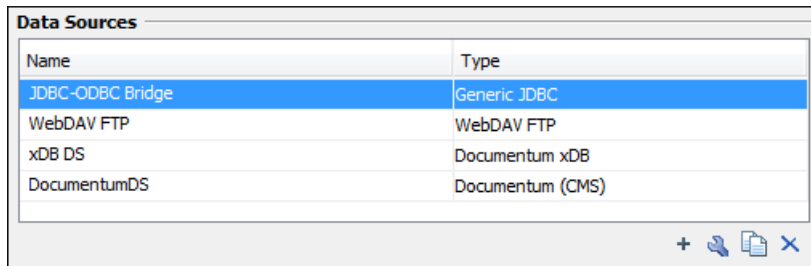


Figure 22: Data Sources Preferences Panel

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.

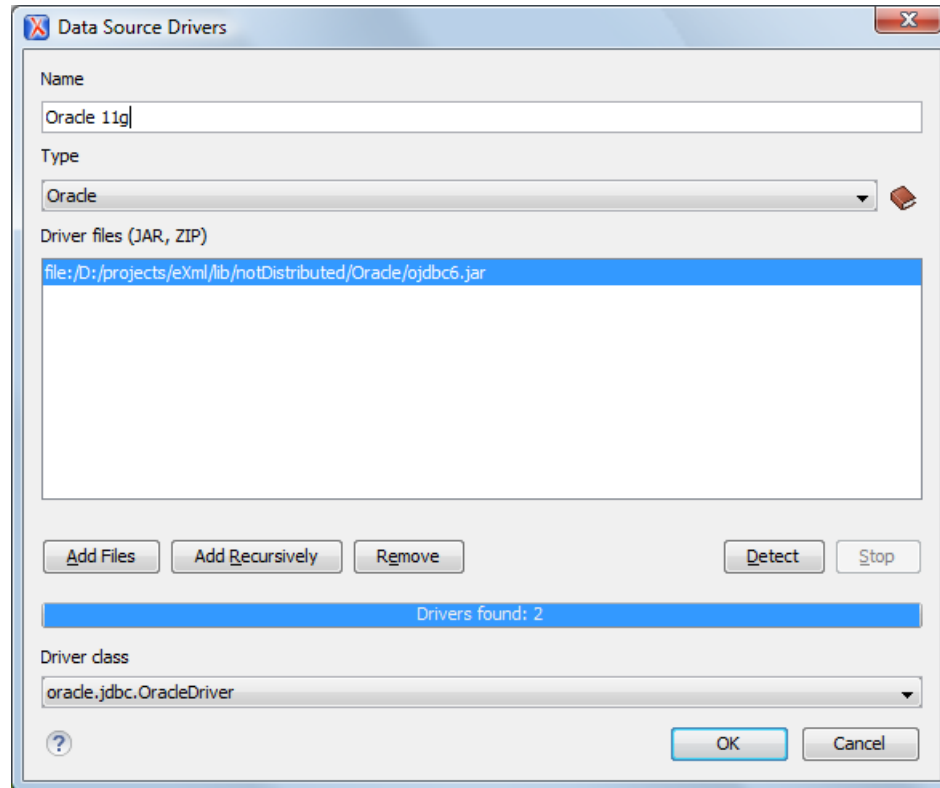


Figure 23: Data Sources Drivers Dialog Box

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 Developer.
- **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.

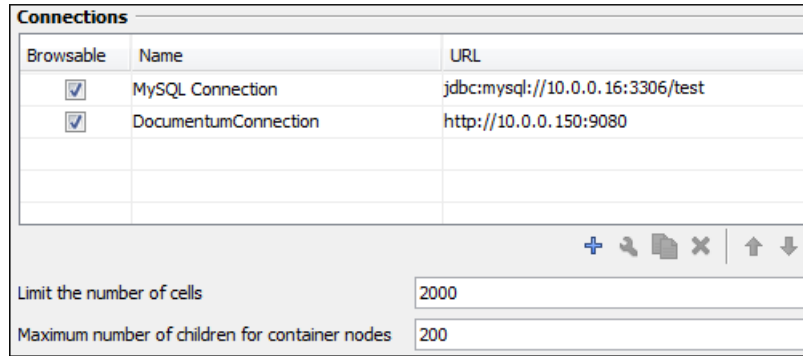


Figure 24: Connections Preferences Panel

The following buttons and options are available at the bottom of the **Connections** panel:



Opens the **Connection** dialog box that allows you to configure a new database connection.

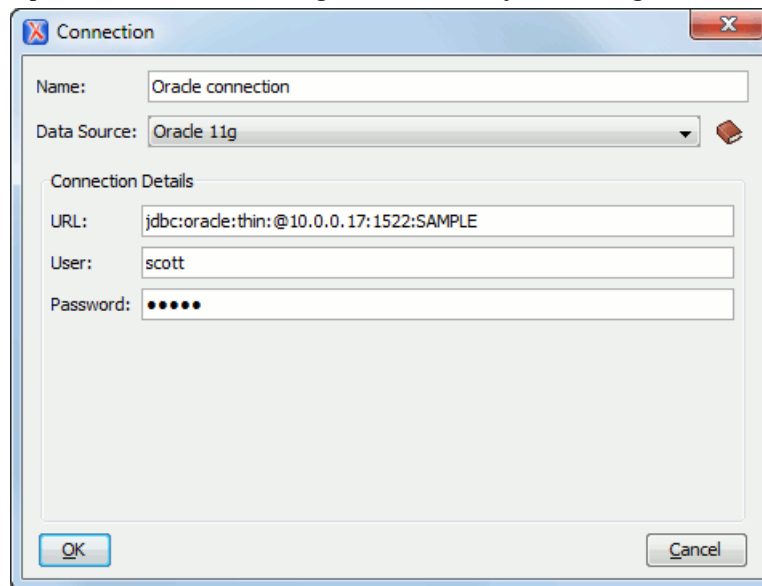


Figure 25: Connection Dialog Box

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. Navigation to other contained resources from the same container is enabled by the <i>Up</i> and <i>Down</i> 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 http://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 Developer:

- **Berkeley DB XML database** - Copy the *jar* files from the Berkeley database install directory into the Oxygen XML Developer 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 Developer for *configuring a DB2 data source*.

- **eXist database** - Copy the *jar* files from the eXist database install directory to the Oxygen XML Developer 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 Developer 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.

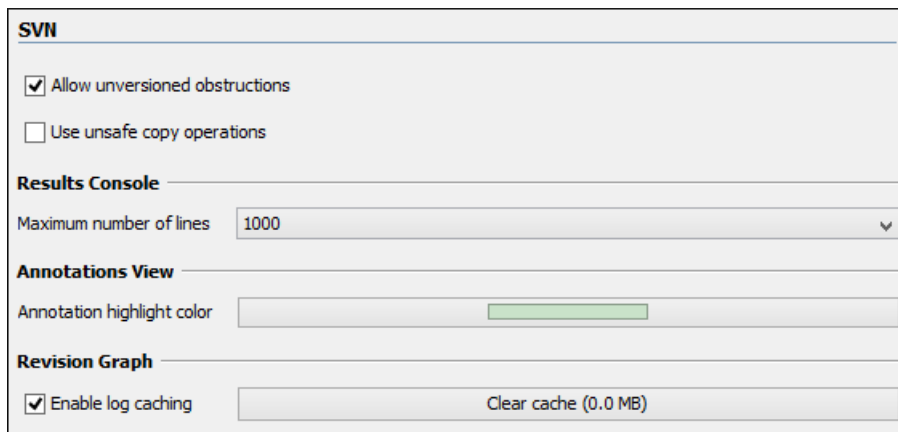


Figure 26: SVN Preferences Panel

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

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 enabled, 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 disabled, the update operation will fail when encountering files in this situation, possibly leaving other files not updated. By default, this option is enabled.
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 checked, the content of the working copies is cached for refresh operations.

Automatically refresh the working copy	If checked, 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. Disabled by default.
Allow moving/renaming mixed revision directories	If enabled, Oxygen XML Developer 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 disabled 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 <i>the Working Copy view</i> . 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 options are available:

Default algorithm	The default algorithm used for performing comparisons. 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).
Ignore Whitespaces	If enabled, 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 Developer 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 Developer signals a difference.

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 enabled, you can enter an XPath 2.0 expression to ignore certain nodes from the comparison. The XPath expression specified in this option is used as the default ignore instructions when the application is started. If you enter an XPath 2.0 expression in the similar option on the toolbar that expression will be used instead.
	Merge adjacent differences	If enabled, the application considers two adjacent differences as one when the differences are painted in the side-by-side editors. If disabled, every difference is represented separately.
	Mark end tags as different for modified elements	If enabled, 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 enabled, empty elements in both expansion states are considered matched (that is <code><element/></code> and <code><element></element></code> are considered equal).
External Diff section	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 <i>a conflict is edited</i>. The parameters <code>\${firstFile}</code> and <code>\${secondFile}</code> specify the positions of the two compared files in the command line for the external diff application. The parameter <code>\${ancestorFile}</code> 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 <i>tokens</i> must be double-quoted for the Oxygen XML Developer to correctly parse and identify them. For example, <code>C:\Program Files\compareDir\app name.exe</code> must be written as <code>"C:\Program Files\compareDir\app name.exe"</code>.

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 <i>proceeds with the visual editing of the conflicting file</i> .

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 27: Files Comparison Preferences Page

This preferences page allows you to configure the following options:

Ignore Whitespaces If enabled, 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 Developer 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 Developer 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 Developer, 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 enabled, you can enter an XPath 2.0 expression to ignore certain nodes from the comparison. The XPath expression specified in this option is used as the default ignore instructions when the application is started. If you enter an XPath 2.0 expression in the similar option on the toolbar that expression will be used instead.

Merge adjacent differences	If enabled, the application considers two adjacent differences as one when the differences are painted in the side-by-side editors. If disabled, every difference is represented separately.
Mark end tags as different for modified elements	If enabled, 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 enabled, empty elements in both expansion states are considered matched (that is <code><element/></code> and <code><element></element></code> 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:

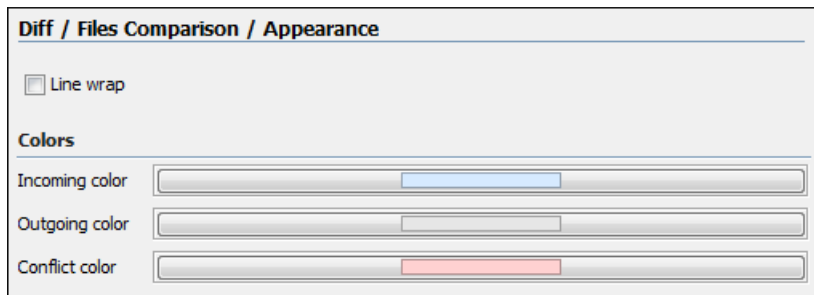


Figure 28: Files Comparison Appearance Preferences Panel

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**.

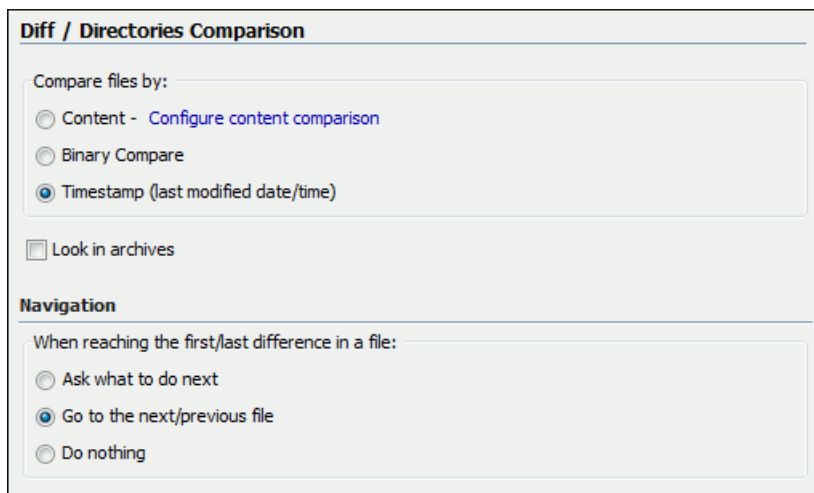


Figure 29: Diff Preferences Page

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 checked, [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**.

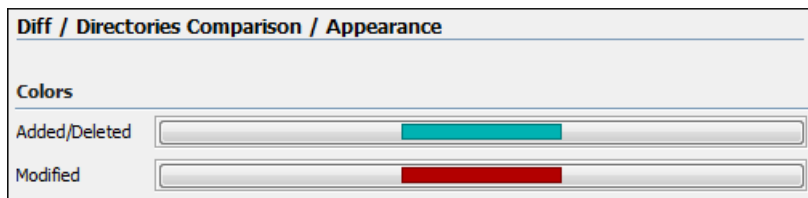


Figure 30: Diff Appearance Preferences Panel

- **Added/Deleted** - Color used for marking added or deleted files and folders.
- **Modified** - Color used for marking modified files.

Archive Preferences

To configure *Archive* preferences, [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 Developer. 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.

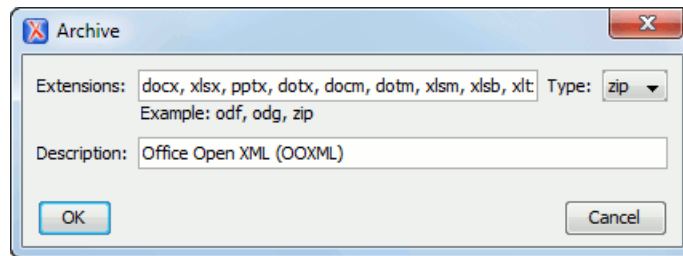


Figure 31: Edit Archive Extension Mappings



Important: You have to restart Oxygen XML Developer 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 Developer. 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 Developer 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 Developer 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 check-boxes 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.

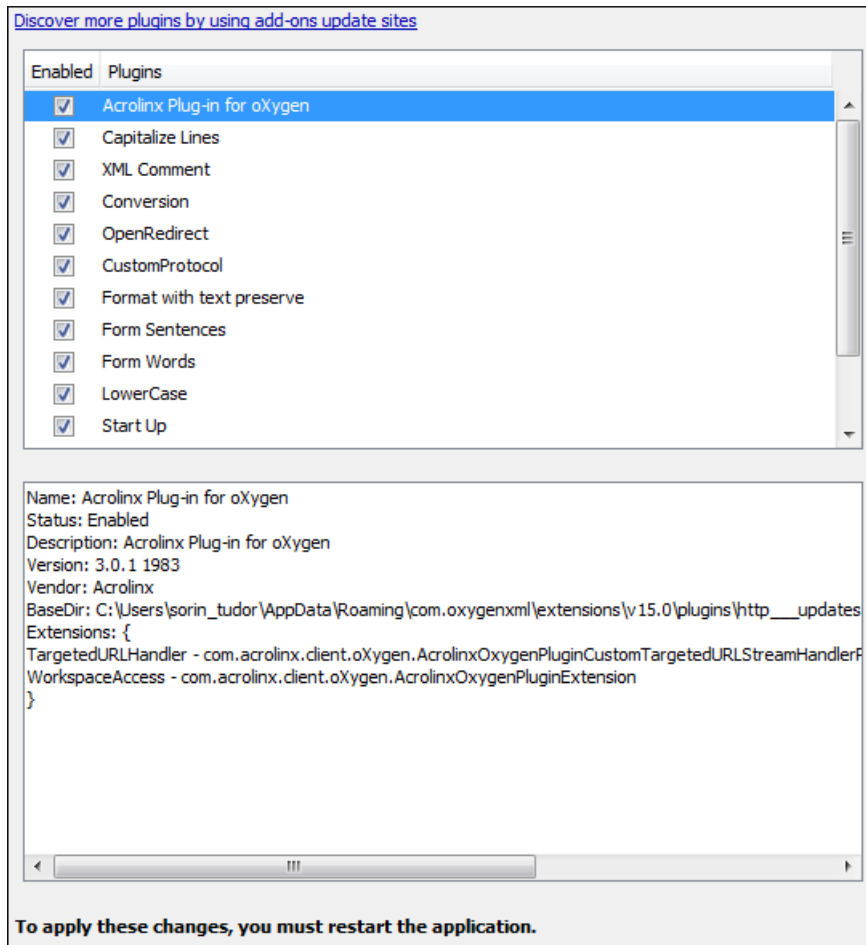


Figure 32: Plugins Preferences Panel

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 Developer 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 Developer. 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 enabled 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.

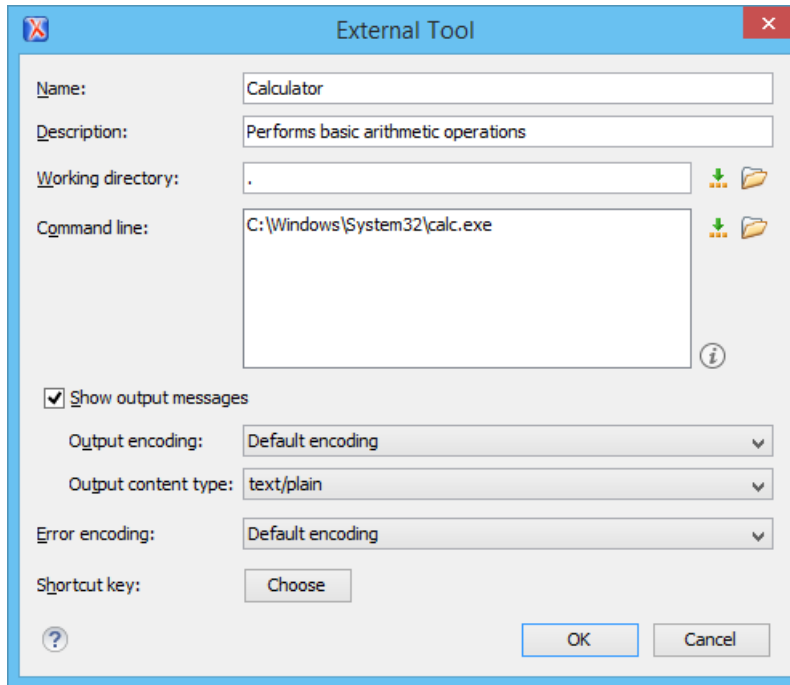


Figure 33: External Tools Configuration Dialog Box

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: <code>\${cfd}</code> , <code>\${pd}</code> , <code>\${oxygenInstallDir}</code> , <code>\${homeDir}</code> , <code>\${system(var.name)}</code> , <code>\${date(pattern)}</code> , <code>\${xpath_eval(expression)}</code> . 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: <code>\${homeDir}</code> , <code>\${home}</code> , <code>\${cfn}</code> , <code>\${cfne}</code> , <code>\${cf}</code> , <code>\${currentFileURL}</code> , <code>\${cfd}</code> , <code>\${cfdU}</code> , <code>\${tsf}</code> , <code>\${pd}</code> , <code>\${pdu}</code> , <code>\${oxygenInstallDir}</code> , <code>\${oxygenHome}</code> , <code>\${frameworksDir}</code> , <code>\${frameworks}</code> , <code>\${ps}</code> , <code>\${timeStamp}</code> , <code>\${uuid}</code> , <code>\${id}</code> , <code>\${afn}</code> , <code>\${afne}</code> , <code>\${af}</code> , <code>\${afu}</code> , <code>\${afd}</code> , <code>\${afdu}</code> , <code>\${ask('message', type, 'default_value')}</code> , <code>\${dbgXML}</code> , <code>\${dbgXSL}</code> , <code>\${env(VAR_NAME)}</code> , <code>\${system(var.name)}</code> , <code>\${date(pattern)}</code> , and <code>\${xpath_eval(expression)}</code> . You can also use the browsing tools to select a file path.
Show output messages	When this option is enabled, all the messages emitted by the external tool are displayed in the Results view. When this option is disabled, 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 by Oxygen XML Developer to read the output of the tool. • Output content type - A list of predefined content type formats that instruct Oxygen XML Developer how to display the generated output. For example, setting the Output content type to <code>text/xml</code> 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 Developer 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** preference page to configure shortcut keys for the actions available in Oxygen XML Developer. The shortcuts assigned to actions are displayed in the table in this preference page. To configure the shortcut keys, *open the [Preferences dialog box \(Options > Preferences\)](#)* and go to **Menu Shortcut Keys** (or **Options > Menu Shortcut Keys**).

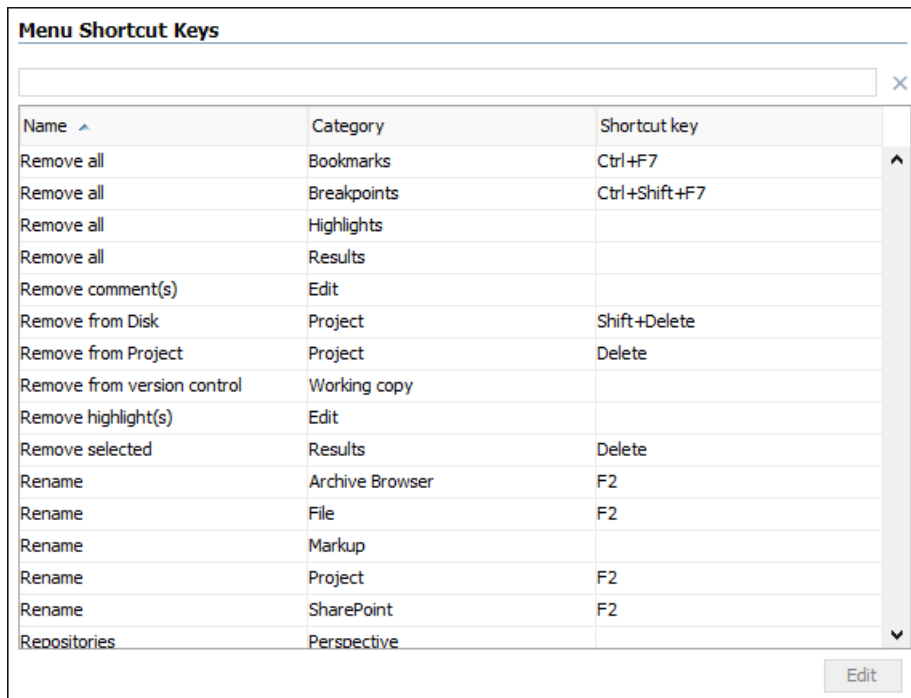


Figure 34: Menu Shortcut Keys Preferences Page

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 to search through any of the columns in the table. You can also click on a column header to sort the column.

The table includes the following columns:

- **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. For example, by default, the *Remove all* action has one shortcut key assigned for *bookmarks* and another one assigned for *breakpoints*.
- **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 enabled, the default behavior of these keys is overridden and the cursor only moves on the current line.

File Types Preferences

Oxygen XML Developer 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**.

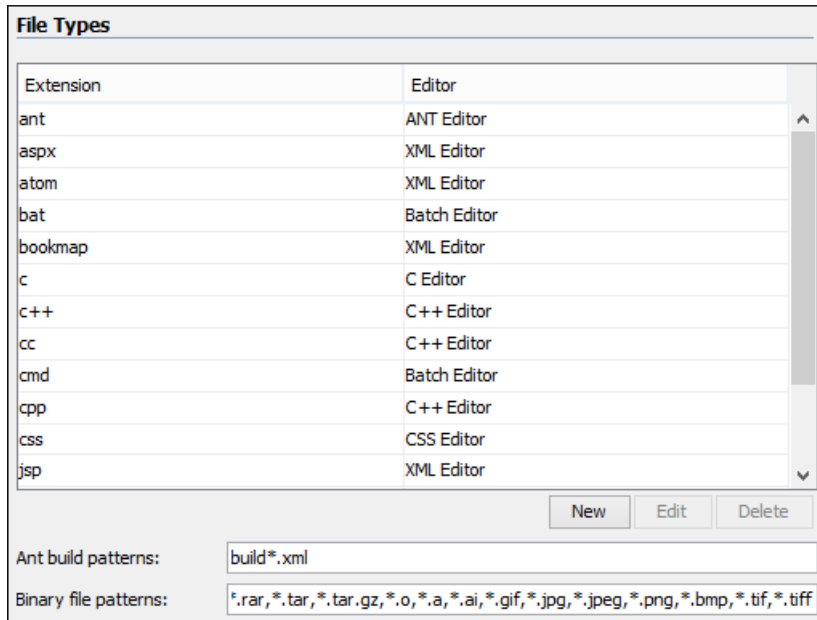


Figure 35: File Types Preferences Panel

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 preference Encoding for non XML files](#) 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 enabled 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 disabled, 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 <i>file</i>) 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, enable this option.  Note: Enabling this option may lead to delays when the indexing is computed.
Content search options section	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 Developer 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 enabled, you can perform XML-specific searches for XML elements and attributes.  Note: Enabling 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 <i>stop words</i>, 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 180

[Open/Find Resource Dialog Box](#) on page 225

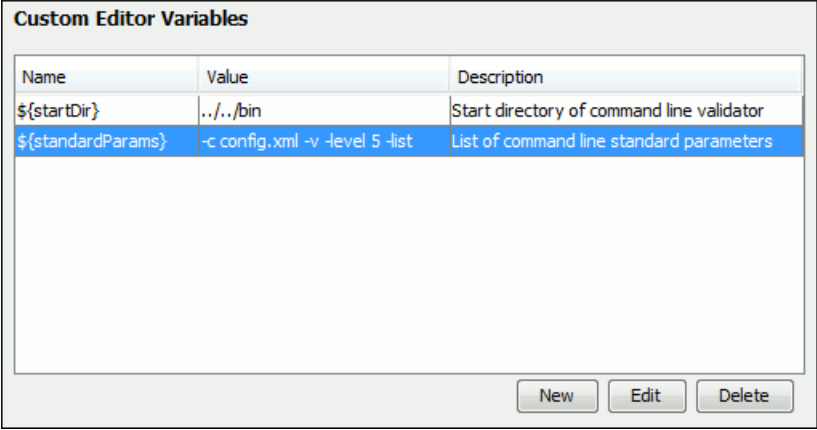
Custom Editor Variables Preferences

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**. To 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.



Name	Value	Description
\${startDir}	../../bin	Start directory of command line validator
\${standardParams}	-c config.xml -v -level 5 -list	List of command line standard parameters

New Edit Delete

Figure 36: 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, <code>myhost.example.com</code>) or as a domain name (for example, <code>example.com</code>). 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 an automatic proxy configuration script (PAC)

If you have set up the path to an automatic proxy configuration script in your system (IE for Windows) Oxygen XML Developer 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 Developer uses the Apache HttpClient to establish connections to HTTP servers. For Oxygen XML Developer 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 <i>Expect: 100-Continue</i> handshake. The purpose of the <i>Expect: 100-Continue</i> 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 <i>Expect: 100-continue</i> handshake can result in noticeable performance improvement when working with databases. The <i>Expect: 100-continue</i> 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 checked, Oxygen XML Developer tries to determine a resource type using the Content-Type header field. This header indicates the <i>Internet media type</i> of the message content, consisting of a type and subtype. For example:

```
Content-Type: text/xml
```

Automatic retry on recoverable error

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.

When enabled, if an HTTP error occurs when Oxygen XML Developer 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 Developer tries to re-send the request to the server.

Automatically accept a security certificate, even if invalid

When enabled, the HTTPS connections that Oxygen XML Developer 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 checked, 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:

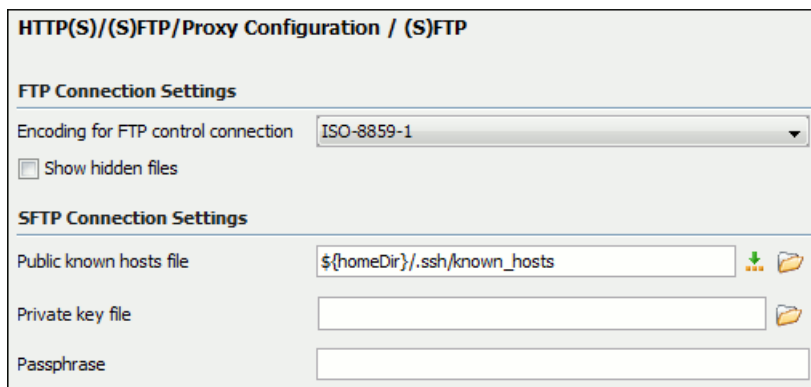


Figure 37: (S)FTP Configuration Preferences Panel

- **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 Developer 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 Developer 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.
Tree Editor section	Format and indent on save	If enabled, Oxygen XML Developer will perform a <i>Format and Indent</i> (pretty-print) action when saving a document that was edited in the Tree Editor .
Information view section	Maximum number of lines	Specifies the maximum number of lines that can be written in the Information view .

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 , 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 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 Developer, 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 Developer includes several distinct layers of option values.

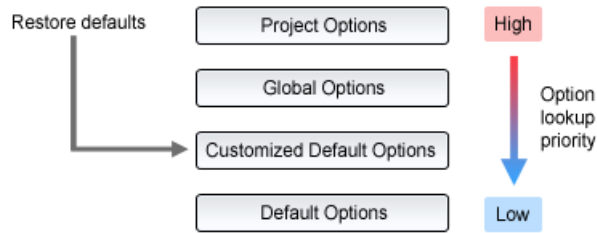


Figure 38: Option Lookup Priority

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 Developer 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 Developer 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 Developer 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 Developer has an extensive set of options that you can configure. When Oxygen XML Developer is 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 Developer 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 Developer 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"/>
```

Sharing and Storing Options

When you configure the Oxygen XML Developer 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.

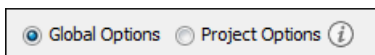


Figure 39: Controlling the Storage of the Preferences

Global Options

By default, **Global Options** is selected in the preferences pages, meaning that the options are stored 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 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`

Project Options

If you select **Project Options**, the preferences are stored in the project file and can 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 the same option configuration that you stored in the project file.



Notice: Some pages do not have the **Project Options** switch, since the options they host might contain sensitive data (passwords, for example), unsuitable for sharing with other users.



Note: 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.

For more information about sharing project files, see the [Sharing a Project - Team Collaboration](#) on page 240 section.

Related information

[Sharing Validation Scenarios](#) on page 292

[Sharing the Transformation Scenarios](#) on page 600

Importing / Exporting 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 options file and the import operation allows you to load the options 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 .
Import Global Options	Allows you to import a set of <i>Global Options</i> from an exported XML options file. You can also select a project-level options file (.xpr) to import all the <i>Global Options</i> 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 enable 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 <i>Global Options</i> to an XML options file. Some user-specific options that are private are not included. For example, passwords and the name of the <i>Review Author</i> is not included in the export operation.

Oxygen XML Developer automatically stores your options in an XML options file. Depending on the platform you are using, this file is located in the following directories:

- [user-home-folder]\AppData\Roaming\com.oxygenxml.developer for Windows Vista/7/8/10
- [user-home-folder]\Application Data\com.oxygenxml.developer for Windows XP
- [user-home-folder]/Library/Preferences/com.oxygenxml.developer for OS X
- [user-home-folder]/.com.oxygenxml.developer for Linux

The name of the options file of Oxygen XML Developer 18.0 is oxyDeveloperOptionsSa18.0.xml.

Reset Global Options

To reset all global preferences to their default values, select **Reset Global Options** from the **Options** menu.

The project level preferences are not changed by this action.

This action also resets the transformation and validation scenarios to the default scenarios and clears recently used file templates.

Associating a File Extension with Oxygen XML Developer

To associate a file extension with Oxygen XML Developer 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 Developer, then click the **Change program** button.
5. In the subsequent dialog box, browse for and choose Oxygen XML Developer.

To associate a file extension with Oxygen XML Developer 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 Developer.
4. Select the **Always Open With** option, then click **Add**.

Configuring the Layout of the Views and Editors

All the Oxygen XML Developer views available in the [Editor Perspective](#), [XSLT Debugger Perspective](#), and [XQuery Debugger Perspective](#) are dockable. To open a view, select it from the **Window > Show View** menu. You can hide a view by closing it with the **X** button at the top-right corner of the view, or with the **Window > Hide current view** action.

You can drag any view to any margin of another view or editor inside the Oxygen XML Developer window. Once you create a layout that suites your needs, you can save it from **Window > Export Layout**. Oxygen XML Developer 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 Developer.

To gain more editing space in the Oxygen XML Developer 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 Developer 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 Developer 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.

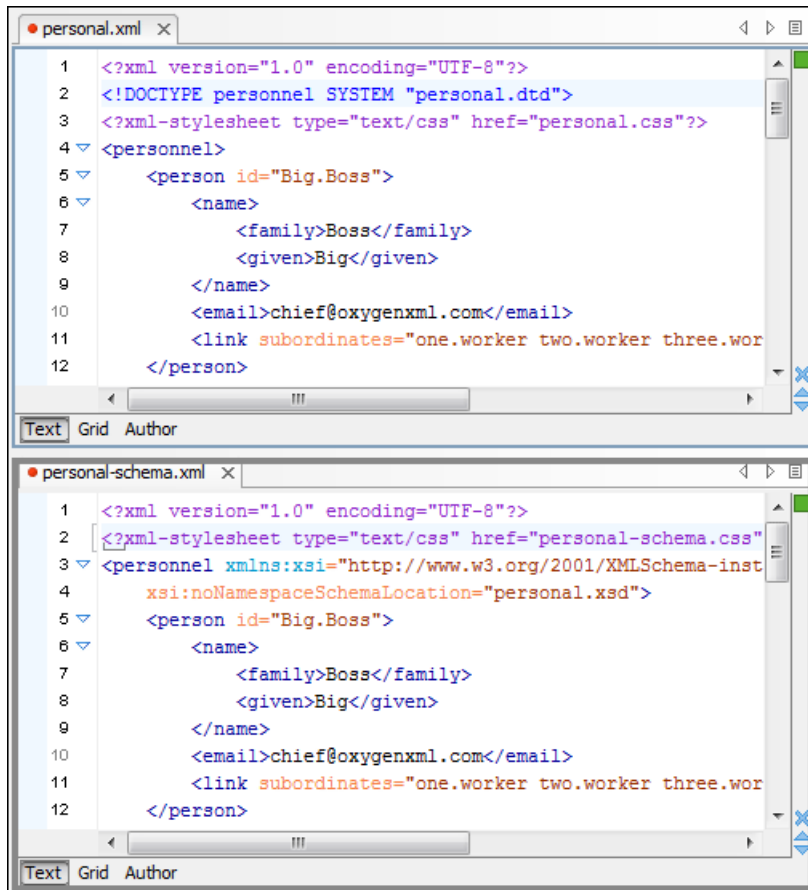


Figure 40: Drag and Drop Editors

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:

 Synchronous scrolling

Select this action to scroll through the tiled editors at the same time.

 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.



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.

You can divide the editing area vertically and horizontally using the following actions available in the **Editor** toolbar and **Window** menu:

-  **Split horizontally** - Splits the editor horizontally. This is useful for comparing and merging content between two documents.
-  **Split vertically** - Splits the editor vertically. This is useful for comparing and merging content between two documents.

-  **Unsplit** - Removes a split action on the editing area.

To maximize or restore the editors, go to **Window > Maximize/Restore Editor Area**.

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 Developer, go to http://oxygenxml.com/demo/Dockable_Views.html.

Configure Toolbars

You can configure the toolbars in Oxygen XML Developer to personalize the interface for your specific needs. You can choose which toolbars to show or hide in the current editor mode (**Text**, **Design**, or **Grid**) and in the current perspective (**Editor**, **XSLT Debugger**, **XQuery Debugger**, or **Database**). 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.

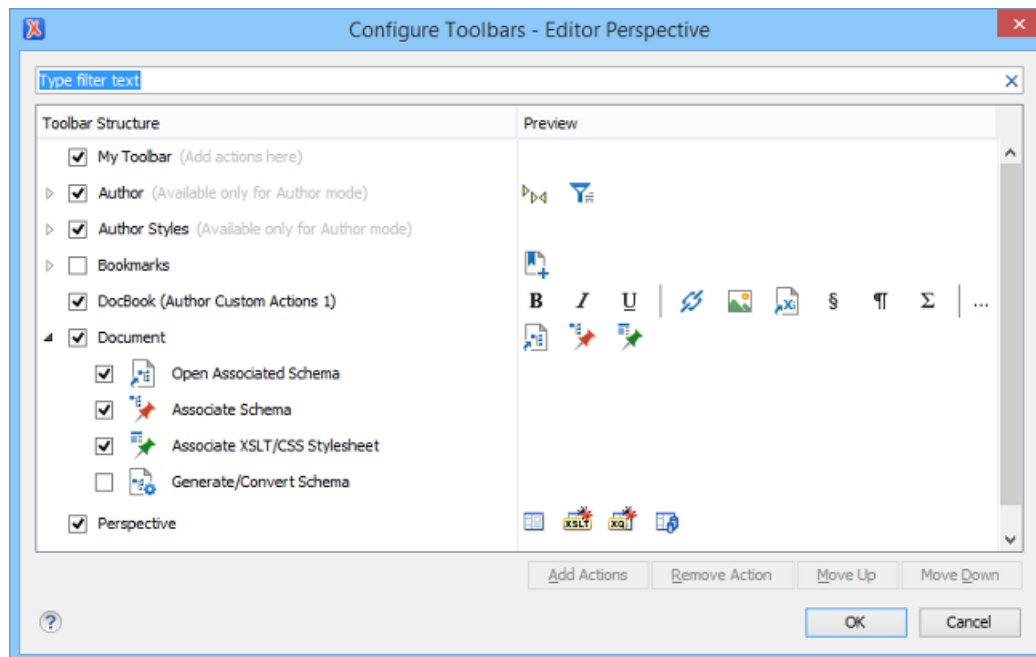


Figure 41: Configure Toolbars Dialog Box

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.

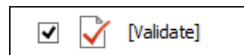
Show or Hide Toolbars

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 (document type associations) 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 Developer.

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.

Scenarios Management

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 (not project-level) transformation scenarios in a specialized <i>scenarios</i> 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 (not project-level) Validation scenarios in a specialized <i>scenarios</i> 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 Developer 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 Developer commands of external engines or other external tools, in transformation scenarios, and in validation scenarios:

- `${oxygenHome}` - Oxygen XML Developer installation folder as URL.
- `${oxygenInstallDir}` - Oxygen XML Developer installation folder as file path.
- `${framework}` - The path (as URL) of the current framework, as part of the `[OXYGEN_INSTALL_DIR]/frameworks` directory.
- `${framework(fr_name)}` - The path (as URL) of the `fr_name` framework.
- `${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 Developer 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 set 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 Developer options.
 - Additional frameworks added in the [Locations preferences page](#).
 - Frameworks installed using the add-ons support.
 - Frameworks found in the [main frameworks location](#) (**Default** or **Custom**).
- `${frameworks}` - The path (as URL) of the `[OXYGEN_INSTALL_DIR]` directory.
- `${frameworkDir}` - The path (as file path) of the current framework, as part of the `[OXYGEN_INSTALL_DIR]/frameworks` directory.
- `${frameworksDir}` - The path (as file path) of the `[OXYGEN_INSTALL_DIR]/frameworks` directory.
- `${home}` - The path (as URL) of the user home folder.
- `${homeDir}` - The path (as file path) of the user home folder.
- `${pdu}` - Current project folder as URL. Usually the current folder selected in the **Project View**.
- `${pd}` - Current project folder as file path. Usually the current folder selected in the **Project View**.

- `${pn}` - Current project name.
- `${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.
- `${cfd}` - Current file folder as file path, that is the path of the current edited document up to the name of the parent folder.
- `${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.
- `${af}` - The local file path of the ZIP archive that includes the current edited document.
- `${afu}` - The URL 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.
- `${currentFileURL}` - Current file as URL, that is the absolute file path of the current edited document represented as URL.
- `${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.
- `${timeStamp}` - Time stamp, that is the current time in Unix format. For example, it can be used to save transformation results in multiple output files on each transformation.
- `${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.
- `${selection}` - The current selected text content in the current edited document. This variable can be used in a code template, in **Author** mode operations, or in a selection plugin.
- `${id}` - Application-level unique identifier. It is a short sequence of 10-12 letters and digits that is not guaranteed to be universally unique.
- `${uuid}` - Universally unique identifier, a unique sequence of 32 hexadecimal digits generated by the Java [UUID](#) class.
- `${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 `${env(VAR_NAME)}` editor variable instead.
- `${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)}
```

- `${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:

Parameter	
url	Format: <code>\${ask('message', url, 'default_value')}</code>
	Description: Input is considered a URL. Oxygen XML Developer checks that the provided URL is valid.
	Example: • <code>\${ask('Input URL', url)}</code> - The displayed dialog box has the name Input URL. The expected input type is URL. • <code>\${ask('Input URL', url, 'http://www.example.com')}</code> - The displayed dialog box has the name Input URL. The expected input type is URL. The input field displays the default value <code>http://www.example.com</code>.
password	Format: <code>\${ask('message', password, 'default')}</code>
	Description: The input is hidden with bullet characters.
	Example: • <code>\${ask('Input password', password)}</code> - The displayed dialog box has the name 'Input password' and the input is hidden with bullet symbols. • <code>\${ask('Input password', password, 'abcd')}</code> - The displayed dialog box has the name 'Input password' and the input hidden with bullet symbols. The input field already contains the default <code>abcd</code> value.
generic	Format: <code>\${ask('message', generic, 'default')}</code>
	Description: The input is considered to be generic text that requires no special handling.
	Example: • <code>\${ask('Hello world!')}</code> - The dialog box has a Hello world! message displayed. • <code>\${ask('Hello world!', generic, 'Hello again!')}</code> - The dialog box has a Hello world! message displayed and the value displayed in the input box is 'Hello again!'.
relative_url	Format: <code>\${ask('message', relative_url, 'default')}</code>
	Description: Input is considered a URL. Oxygen XML Developer tries to make the URL relative to that of the document you are editing.  Note: If the <code>\$ask</code> editor variable is expanded in content that is not yet saved (such as an <i>untitled</i> file, whose path cannot be determined), then Oxygen XML Developer will transform it into an absolute URL.
	Example: • <code>\${ask('File location', relative_url, 'C:/example.txt')}</code> - The dialog box has the name 'File location'. The URL inserted in the input box is made relative to the current edited document location.

Parameter	
combobox	Format: <code>\${ask('message', combobox, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default')}</code> Description: Displays a dialog box that offers a drop-down menu. The drop-down menu is populated with the given <code>rendered_value</code> values. Choosing such a value will return its associated value (<code>real_value</code>).  Note: The 'default' parameter specifies the default selected value and can match either a key or a value. Example: • <code>\${ask('Operating System', combobox, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'osx')}</code> - 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 <code>osx</code> key. However, the same result could be obtained if the default value is indicated by <code>Mac OS X</code>, as in the following example: <code>\${ask('Operating System', combobox, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'Mac OS X')}</code> • <code>\${ask('Mobile OS', combobox, ('win':'Windows Mobile';'ios':'iOS';'and':'Android'), 'Android')}</code>
editable_combobox	Format: <code>\${ask('message', editable_combobox, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default')}</code> Description: Displays a dialog box that offers a drop-down menu with editable elements. The drop-down menu is populated with the given <code>rendered_value</code> values. Choosing such a value will return its associated real value (<code>real_value</code>) 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: • <code>\${ask('Operating System', editable_combobox, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'osx')}</code> - 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.

Parameter	
radio	Format: <code>\${ask('message', radio, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default')}</code> 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: <code>\${ask('Operating System', radio, ('win':'Microsoft Windows';'osx':'Mac OS X';'lnx':'Linux/UNIX'), 'osx')}</code> - 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.
- `${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).
- `${tsf}` - The transformation result file path. If the current opened file has an associated scenario that specifies a transformation output file, this variable expands to it.
- `${dsu}` - The path of the detected schema as a URL for the current validated XML document.
- `${ds}` - The path of the detected schema as a local file path for the current validated XML document.
- `${cp}` - Current page number. Used to display the current page number on each printed page in the **Editor / Print** Preferences page.
- `${tp}` - Total number of pages in the document. Used to display the total number of pages on each printed page in the **Editor / Print** Preferences page.
- `${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)}')}
```

- `$(i18n(key))` - Editor variable used only at framework level to allow translating names and descriptions of **Author** mode actions in multiple actions.

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 154

Custom System Properties

A number of Java system properties can be set in the application to influence its behavior. For information about how to do this, see the [Setting a system property](#) on page 163 section.

Table 3: Custom System Properties

Property	Allowed values	Default value	Purpose
<code>com.oxygenxml.disable.http.protocol.handlers</code>	true or false	false	By default, Oxygen XML Developer 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.
<code>com.oxygenxml.default.options</code>	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 147.
<code>com.oxygenxml.customOptionsDir</code>	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 Global Options on page 149.

Property	Allowed values	Default value	Purpose
<i>com.oxygenxml.ApplicationDataFolder</i> (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 <i>APPDATA</i> folder is located.
<i>com.oxygenxml.editor.frameworks.url</i>	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 Locations Preferences on page 60 page.
<i>com.oxygenxml.MultipleInstances</i>	true or false	false	If set to true, multiple instances of the application are allowed to be started.
<i>com.oxygenxml.xep.location</i>	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 on page 115 page.
<i>com.oxygenxml.additional.classpath</i>	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 <i>lib</i> folder.
<i>com.oxygenxml.user.home</i> (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.
<i>com.oxygenxml.use.late.delegation.for.author.extensions</i>	true or false	true	All Java extensions in a document type configuration are instantiated in a separate class loader. When true , the JAR libraries used in a certain document type configuration will have priority to resolve classes before delegating to the parent class loader. When false , the parent class loader will take precedence.
<i>com.oxygenxml.stack.size.validation.threads</i>	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 <i>Thread</i> stack size to parse complex schemas. The default value should be more than enough.

Property	Allowed values	Default value	Purpose
<code>com.oxygenxml.jing.skip.validation.xhtml.data.attrs</code>	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.
<code>com.oxygenxml.report.problems.url</code>	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 <code>report</code> parameter with the POST HTTP method.

Related information

[Setting a system property](#) on page 163

Localizing the User Interface

You can select the language of the Oxygen XML Developer user interface. Oxygen XML Developer 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 Developer user interface by creating an interface localization file. The example in this procedure is based on the Spanish language, and a standard Oxygen XML Developer 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 Developer support team and ask them to send you the `translation.xml` sample file.
3. Open `translation.xml` in Oxygen XML Developer. 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 enable the **Other language** option. Browse for the *translation.xml* file.
7. Restart the application.

Setting a Java Virtual Machine Parameter in the Launcher Configuration File / Start-up Script

You can set Java Virtual Machine parameters (for example, if you want to increase the maximum amount of memory available) for the Oxygen XML Developer [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 Developer uses on Windows

To increase the memory available to Oxygen XML Developer on Windows:

- Browse to the installation directory of Oxygen XML Developer.
- Locate the `-Xmx` parameter in the `oxygenDeveloper18.0.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 Developer. Go to **Help > About** and verify the amount of memory that is actually available (see the last row in the **About** dialog box). If Oxygen XML Developer does not start and you receive an 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 `oxygenDeveloper18.0.vmoptions` to your desktop (or to any other folder with write access), modify it there, then copy it back to the Oxygen XML Developer installation folder.



Note: The parameters from the `.vmoptions` file are used when you start Oxygen XML Developer 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 Developer uses on OS X

To increase the memory available to Oxygen XML Developer on OS X:

- **Ctrl + Single-Click (Command + Single-Click on OS X)** (or right-click) the Oxygen XML Developer 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 7, 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:

```
oxygenDeveloper18.0.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 Developer system properties, go to [Custom System Properties](#) on page 159.

Disabling DPI Scaling

Some users may prefer the look of smaller icons in a 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 Developer with the `oxygenDeveloper.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=Developer"\  
-Dcom.oxygenxml.editor.plugins.dir="$DEVELOPER_HOME/plugins\  
-Xmx600m\  
...
```

On **Linux**, the Java command should look like this:

```
java -Xmx600m\  
"-Dcom.oxygenxml.editor.plugins.dir="$DEVELOPER_HOME/plugins"
```

Creating Custom Startup Parameters File

The startup launchers for Oxygen XML Developer 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 Developer.

Chapter

5

Perspectives

Topics:

- [Editor Perspective](#)
- [XSLT Debugger Perspective](#)
- [XQuery Debugger Perspective](#)
- [Database Perspective](#)

The detailed descriptions of the **Editor**, **XSLT Debugger**, **XQuery Debugger**, and **Database** perspectives.

An Oxygen XML Developer perspective is an interface layout geared towards a specific use. The Oxygen XML Developer 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 Developer. You can change the perspective by selecting the respective icons in the top-right corner of Oxygen XML Developer 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 Developer 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 Developer



(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 Developer. Most of the menus are common for all types of documents. However, Oxygen XML Developer 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. 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 Developer includes a large variety of 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 controls located on the top-right corner of the view ().

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.
- **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.
- **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.
- **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

[XSLT Debugger Perspective](#) on page 167

[XQuery Debugger Perspective](#) on page 168

[Database Perspective](#) on page 169

Supported Document Types

You can use the main editing pane in Oxygen XML Developer 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

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**.

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 <code>xml:result</code> 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 Developer, go to http://www.oxygenxml.com/demo/XSLT_Debugger.html.

Related information

[Editor Perspective](#) on page 166

[XQuery Debugger Perspective](#) on page 168

[Database Perspective](#) on page 169

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 Developer, go to http://www.oxygenxml.com/demo/XQuery_Debugger.html.

Related information

[Editor Perspective](#) on page 166

[XSLT Debugger Perspective](#) on page 167

[Database Perspective](#) on page 169

Database Perspective

The **Database** perspective allows you to manage databases. To switch the focus to this perspective, select the **Database**

Debugger button in the top-right corner of Oxygen XML Developer  or select **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 Developer 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 Developer. For a detailed feature matrix that compares the Academic, Professional, and Enterprise editions of Oxygen XML Developer [go to the Oxygen XML Developer 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 Developer by registering the database driver as a *Generic JDBC* type driver when defining the data source for accessing the database. For more information, see the [Database Connection Support](#) on page 705 section.

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 Developer editor panel.
- Handling the values from **XML Type** columns as XML instance documents that can be opened and edited in an Oxygen XML Developer 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

[Editor Perspective](#) on page 166

[XSLT Debugger Perspective](#) on page 167

[XQuery Debugger Perspective](#) on page 168

Chapter

6

Editing Modes

Topics:

- [Text Editing Mode](#)
- [Grid Editing Mode](#)
- [Design Editing Mode](#)

Oxygen XML Developer includes several editing modes to suit your type of editing.

The main editing area in Oxygen XML Developer 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 Developer 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.
- **Design** - This mode is found in the schema editor and represents the schema as a diagram.

Text Editing Mode

The **Text** mode of Oxygen XML Developer provides the usual functionality of a plain text editor. It also includes a variety of advanced features that are unique to Oxygen XML Developer.

Related information

[Editing XML Documents in Text Mode](#) on page 245

Text Mode Editor

The **Text** mode **Author** editor in Oxygen XML Developer is designed to be a simple, yet powerful, XML editor. It provides support to help you edit, transform, and debug XML-based documents. It also includes a specialized [Content Completion Assistant](#), an [Outline view](#), and many other helpful features.

Related information

[Changing the colors displayed in the Text Mode Editor](#) on page 87

Navigating the Document Content in Text Mode

Oxygen XML Developer includes some useful features to help you navigate XML documents in **Text** mode.

Using the Keyboard

Oxygen XML Developer 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 Developer 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. Enabled after at least one press of the **Back** button.

Navigating with the Outline View

Oxygen XML Developer 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.

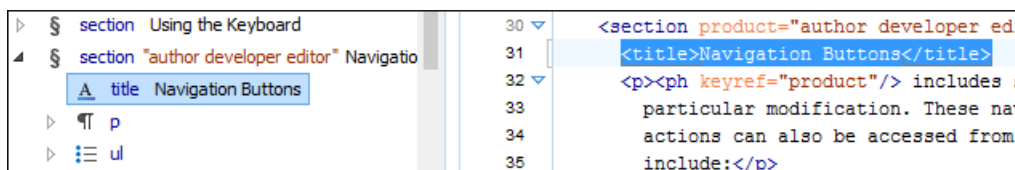


Figure 42: Outline View Navigation in Text Mode

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.

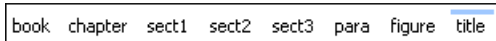


Figure 43: Breadcrumb in Text Mode

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))**.

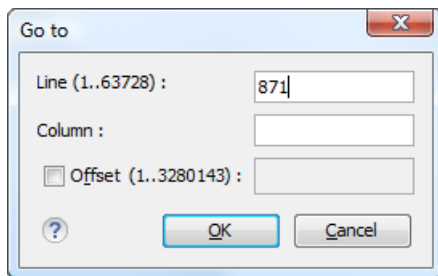


Figure 44: Go to Dialog Box

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.

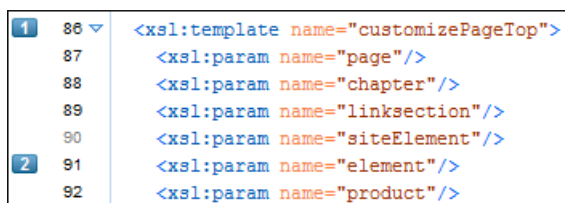


Figure 45: Editor Bookmarks

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 large selection of useful 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 on the contextual menu and toolbar associated to this panel enable the creation of XML projects and shortcuts to various operations on the project documents.

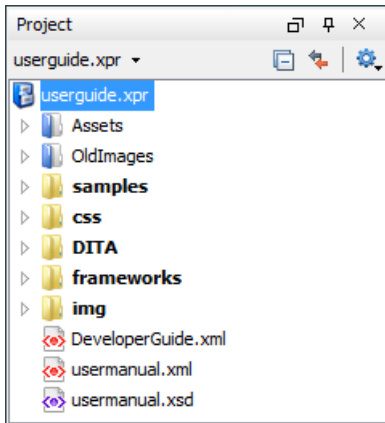


Figure 46: Project View

By default, the view is positioned on the left side of the Oxygen XML Developer 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 Developer XPR project file from the file explorer into the **Project** panel.



Note: 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 Developer 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.

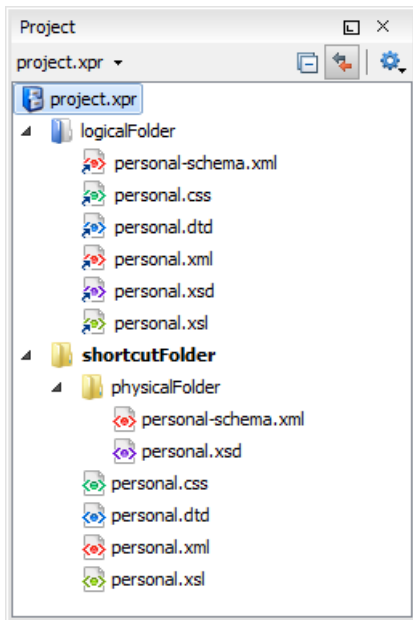


Figure 47: Project View with Examples of all Three Types of Resources

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 <i>project root</i> , 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 <i>project root</i> , 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.

Add Content to a Logical Folder

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.

Managing Project Content

Creating/Adding Files and 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 enabled 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.



Note: The linked files presented in the **Project** view are marked with a special icon. Linked folders are displayed in bold text.

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 Developer 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 Oxygen consoles** option must be enabled 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.

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 Developer 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 are used 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: Archive Browser , 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

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 Developer XPR project file from the file explorer into the **Project** panel.



Note: 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.

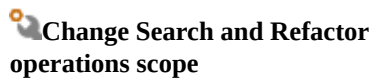


Filters

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.

Related information

[Working with EPUB](#) on page 697

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 Developer layout, next to the **Project** and **DITA Maps Manager** views. If the view is not displayed, it can be opened from the **Window > Show View** menu.

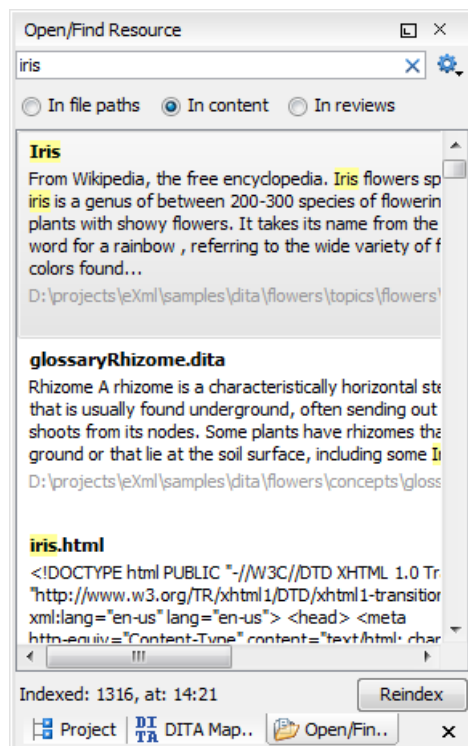


Figure 48: Open/Find Resource View

You can use this view to find a file in the current Oxygen XML Developer project 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).



Note: Full support for searching in document edits is available only in the Enterprise edition of Oxygen XML Developer. The Professional edition offers 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 *samp les* 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](#).
- **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 Developer 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 is 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 Developer 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 http://oxygenxml.com/demo/Open_Find_Resource.html.

Related information

[Open/Find Resource Dialog Box](#) on page 225

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 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.

The drag and drop actions in the **Outline** view can be *disabled and enabled from a Preferences page*.

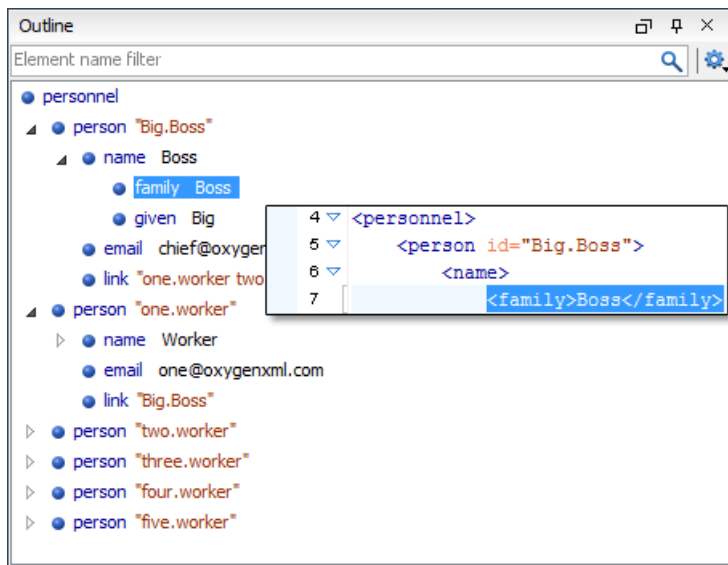


Figure 49: Outline View in Text Mode

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 with a specified value are rendered with a bold font, and their values with a plain font.



Note: The names of the attributes with an empty string value are also rendered bold.

- 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.

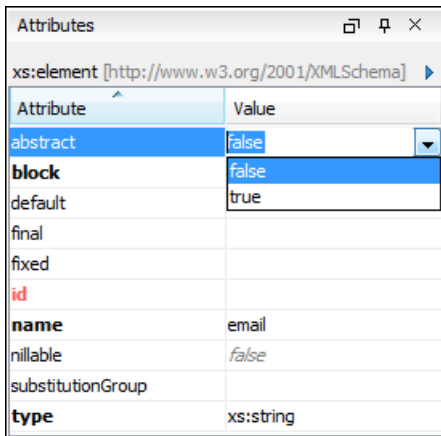


Figure 50: Attributes View

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.
- 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 from the **Window > Show View** menu.

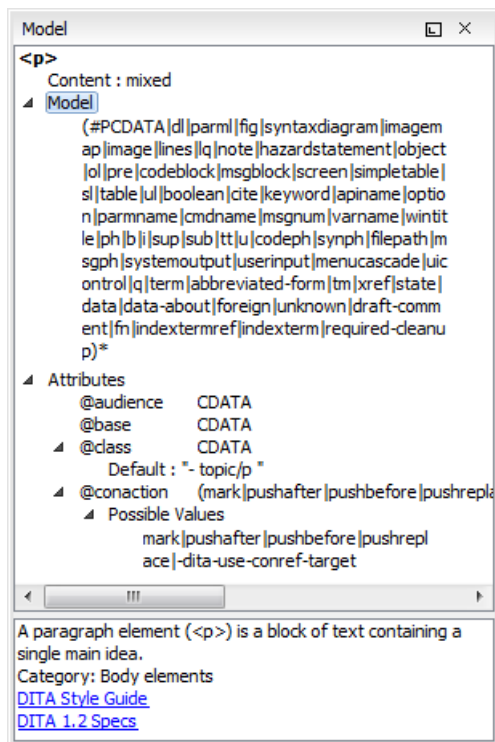


Figure 51: Model View

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.

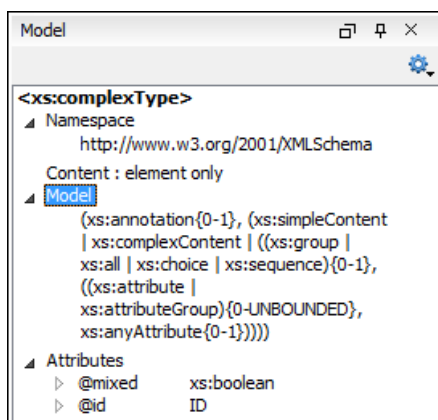


Figure 52: Element Structure Panel

Annotation Panel

The **Annotation** panel displays the annotation information for the currently selected element. This information is collected from the XML schema.

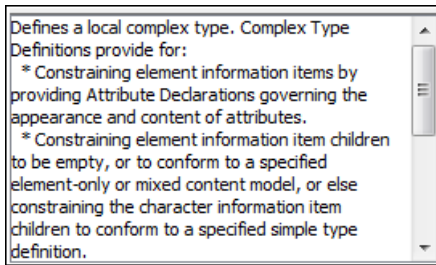


Figure 53: Annotation panel

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 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 54: 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:

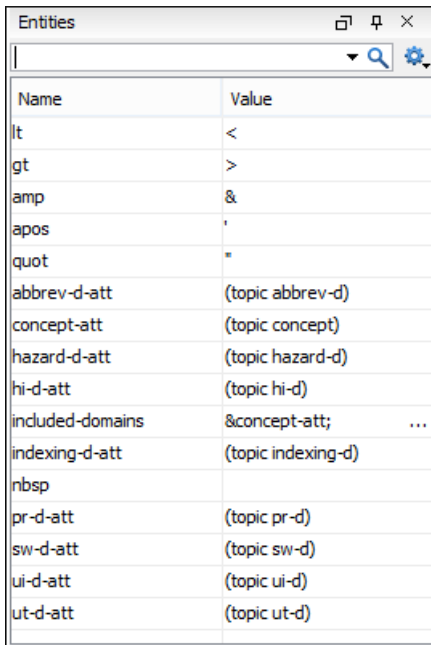
- *Built-in or Predefined* - Entities that are part of the predefined XML markup (<, >, &, ', " ;).
- *Internal* - Defined in the DOCTYPE declaration header of the current XML.
- *External* - Defined in an external DTD module included in the DTD referenced in the XML DOCTYPE declaration.



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 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.



Name	Value
lt	<
gt	>
amp	&
apos	'
quot	"
abbrev-d-att	(topic abbrev-d)
concept-att	(topic concept)
hazard-d-att	(topic hazard-d)
hi-d-att	(topic hi-d)
included-domains	&concept-att; ...
indexing-d-att	(topic indexing-d)
nbsp	
pr-d-att	(topic pr-d)
sw-d-att	(topic sw-d)
ui-d-att	(topic ui-d)
ut-d-att	(topic ut-d)

Figure 55: 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.

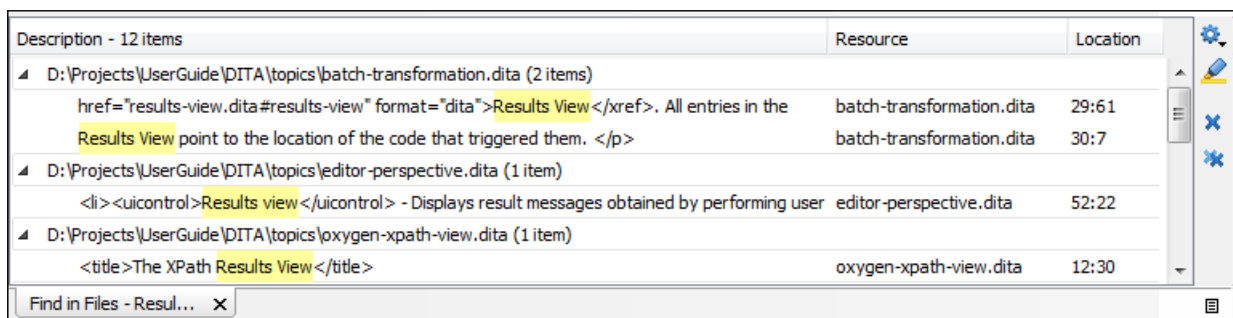


Note: When entering filters, you can use the ? and * wildcards. Also, you can enter multiple filters by separating them with a comma.

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:

- [Validate action](#)
- [Transform action](#)
- [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](#)



Description - 12 items	Resource	Location
D:\Projects\UserGuide\DITA\topics\batch-transformation.dita (2 items) href="results-view.dita#results-view" format="dita">Results View</xref>. All entries in the Results View point to the location of the code that triggered them. </p>	batch-transformation.dita	29:61
D:\Projects\UserGuide\DITA\topics\editor-perspective.dita (1 item) <i><uicontrol>Results view</uicontrol> - Displays result messages obtained by performing user	editor-perspective.dita	52:22
D:\Projects\UserGuide\DITA\topics\oxygen-xpath-view.dita (1 item) <title>The XPath Results View</title>	oxygen-xpath-view.dita	12:30

Figure 56: 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 Developer 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 the table grid:

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 the information associated with the selected messages:

- The file path of the document that triggered the output message.

- The path of the main 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 .
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.

Window > Results Menu

The following actions are also available from the **Window > Results** menu:

Save Results	Saves the complete list of messages in a file in text format.
Print Results	Sends the complete list of messages to a printer.
Close	Closes the current tab.
Close All Tabs	Closes all opened tabs.
Close Other Tabs	Closes all other opened tabs, other than the current one.

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-end0">
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-start0">
9       <xsl:apply-templates select=""/>
10    </fo:list-item-body>
11  </fo:list-item>
12</xsl:template>

```

Figure 57: Example of Coloring XML Tags by Prefix

Related information

Changing the colors displayed in the Text Mode Editor on page 87

Presenting Validation Errors in Text Mode

Oxygen XML Developer can be configured to *automatically validate documents* while editing in the **Text** mode, and actions are also available to *manually validate documents* on-request.

A line with a validation error or warning is marked in the editor panel by underlining the error region with a red line, while validation warnings are underlined in yellow. Hovering over a validation error presents a tooltip message with more details about the error and *possible quick fixes* (if available for that error or warning).

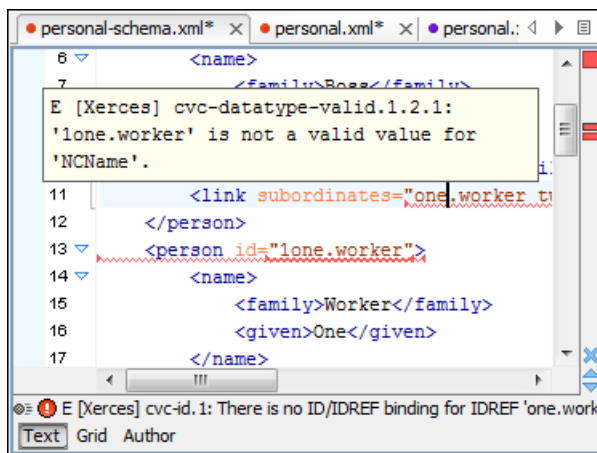


Figure 58: Presenting Validation Errors in Text Mode

Also, the ruler on the right side of the editor panel is designed to display the errors found during the validation process and to help you locate them in the document. The ruler contains the following:

Upper Part of the Ruler

A success indicator square will turn green if the validation is successful, red if validation errors are found, or yellow if validation warnings are found. More details about the errors or warnings are displayed in a tool tip when you hover over indicator square. If there are numerous errors, only the first three are presented in the tool tip.

Middle Part of the Ruler

Errors are depicted with red markers, and warnings with yellow 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 error or warning 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 Ruler

Two navigation arrows () allow you to skip to the next or previous error. 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 error markers.

Status messages from every validation action are logged in the [Information view](#).

If you want to see all the validation error 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 282

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 Developer 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 195

[Inserting Symbols](#) on page 206

Grid Editing Mode

The **Grid** editing mode in Oxygen XML Developer displays XML documents in a structured spreadsheet-like grid. It allows you to easily edit XML documents consisting of repetitive patterns.

Grid Mode Editor

The Oxygen XML Developer **Grid** editor displays the XML document as a structured grid of nested tables. To activate this mode, select **Grid** at the bottom of the editing area.

Modify Content Without Working with XML Tags

If you are a non-technical user, you can modify the text content of the edited document without working with the XML tags directly. You can expand and collapse the tables using the mouse cursor and also display or hide the elements of the document as nested. The document structure can also be changed easily with drag and drop operations on the grid components.

Changing the Font Size

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 (Command + MouseWheelForward on OS X)) Increases the font size with one point for each execution of the action.

Decrease editor font (Ctrl + NumPad- (Command + NumPad- on OS X) or Ctrl + MouseWheelBackwards (Command + MouseWheelBackwards on OS X)) Decreases the font size with one point for each execution of the action.

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.*

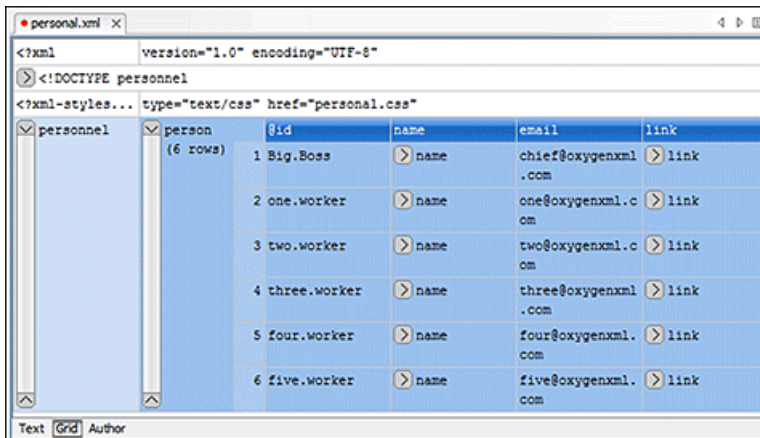


Figure 59: Grid Editing Mode

Switch Editing Modes

To switch back from the **Grid** mode to the **Text** or **Author** mode, use the **Text** and **Grid** buttons from the bottom of the editor. You are also able to perform this switch from **Document > Edit Mode > Grid** and **Document > Edit Mode > Text**.

Content Completion Assistant

If the edited document is associated with a schema (DTD, XML Schema, Relax NG, etc.), the editor offers a **Content Completion Assistant** for the elements and attributes names and values. 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, you have to start editing (for example, double-click a cell). Pressing **Ctrl + Space (Command + Space on OS X)** on your keyboard also displays the pop-up menu.

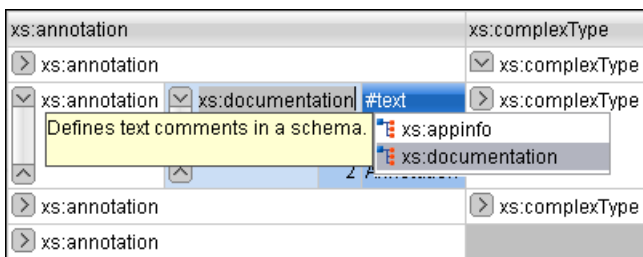


Figure 60: Content Completion in Grid Editing Mode

To watch our video demonstration about some of the features available in the **Grid** editor, go to http://oxygenxml.com/demo/Grid_Editor.html.

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.

<?xml	version="1.0" encoding="UTF-8"				
test	table	tr	@id	first	last
		(3 rows)	1	10001	Jhon Doe
			2	10002	Mark Ewing
			3	10003	Dave Flint

Figure 61: Grid Layout

The other layout mode is tree-like. It does not create any tables and it only presents the structure of the document.

<?xml	version="1.0" encoding="UTF-8"				
test	table	tr	@id	10001	
			first	Jhon	
			last	Doe	
		tr	@id	10002	
			first	Mark	
			last	Ewing	
		tr	@id	10003	
			first	Dave	
			last	Flint	

Figure 62: 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 4: 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.

<?xml	version="1.0" encoding="UTF-8"
sample (9 rows)	#text
1	عندما يريد العالم أن يتكلم، فهو يتحدث بلغة يونيكود.
2	Quan el món vol conversar, parla Unicode
3	כאשר העולם רוצה לדבר, הוא מדבר ב-Unicode
4	Ha a világ beszélni akar, azt Unicode-ul mondja
5	Quando il mondo vuole comunicare, parla Unicode
6	世界的に話すなら、Unicode です。
7	세계를 향한 대화, 유니코드로 하십시오
8	Når verden vil snakke, snakker den Unicode
9	Når verda ønskjer å snakke, talar ho Unicode

Figure 63: Default left to right text orientation

<?xml	"version="1.0" encoding="UTF-8"
sample (9 rows)	#text
1	عندما يريد العالم أن يتكلم، فهو يتحدث بلغة يونيكود.
2	Quan el món vol conversar, parla Unicode
3	כאשר העולם רוצה לדבר, הוא מדבר ב-Unicode
4	Ha a világ beszélni akar, azt Unicode-ul mondja
5	Quando il mondo vuole comunicare, parla Unicode
6	。世界的に話すなら、Unicode です
7	세계를 향한 대화, 유니코드로 하십시오
8	Når verden vil snakke, snakker den Unicode
9	Når verda ønskjer å snakke, talar ho Unicode

Figure 64: Right to left text orientation

Related information

[Bidirectional Text Support in Text Mode](#) on page 192

[Inserting Symbols](#) on page 206

Design Editing Mode

This section presents the **Design** mode that allows you to edit XML Schemas in a visual schema diagram editor.

XML Schema Diagram Editor (Design Mode)

XML Schemas enable document designers to specify the allowed structure and content of an XML document and to check if an XML document is valid.

Oxygen XML Developer 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 Developer 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.

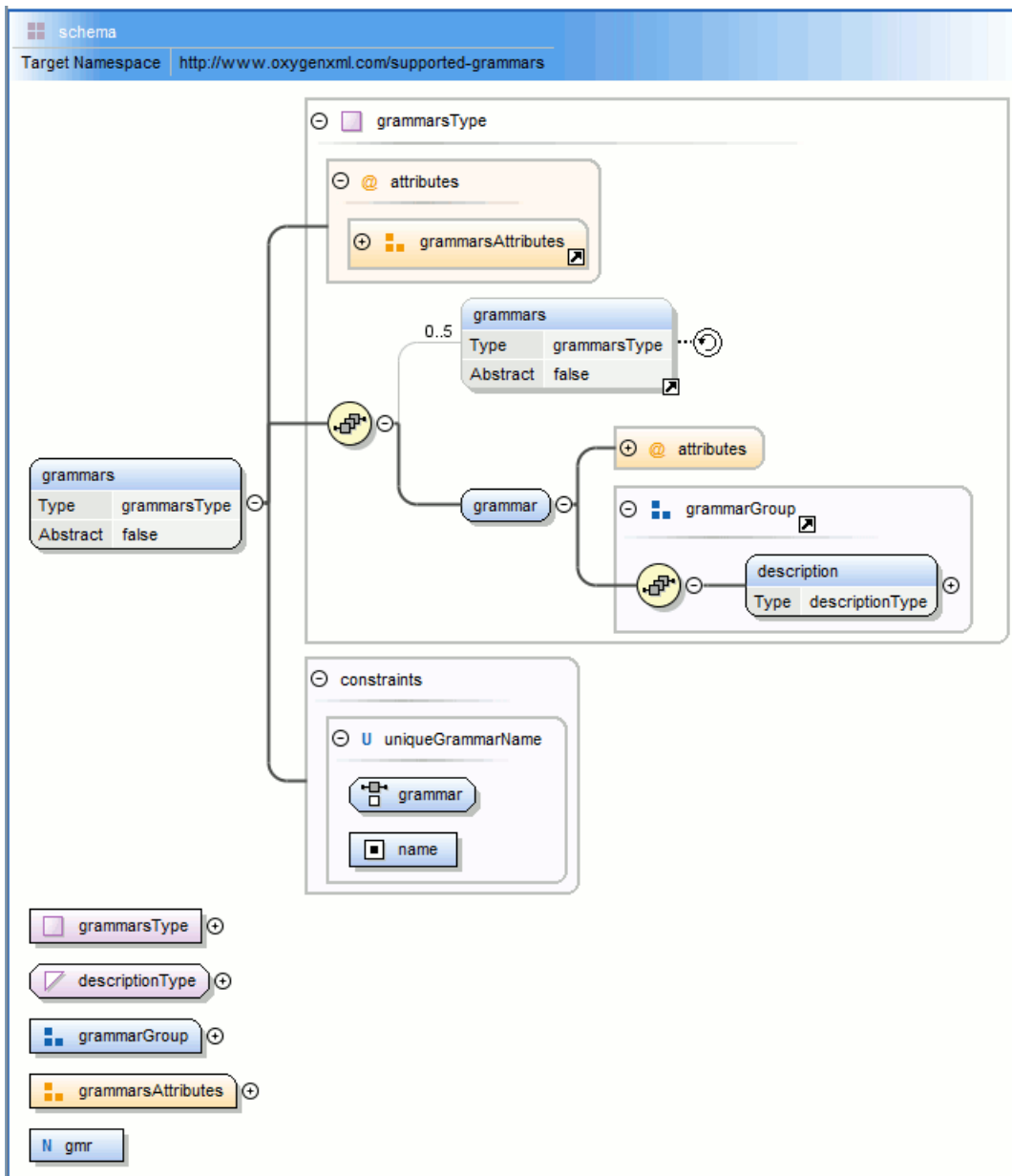


Figure 65: XML Schema Diagram

To watch our video demonstration about the basic aspects of designing an XML Schema using the new Schema Editor, go to http://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 Developer.
-  **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.

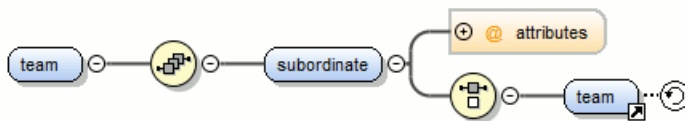


Figure 66: Recursive Reference

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 from the **Window > Show View** menu.

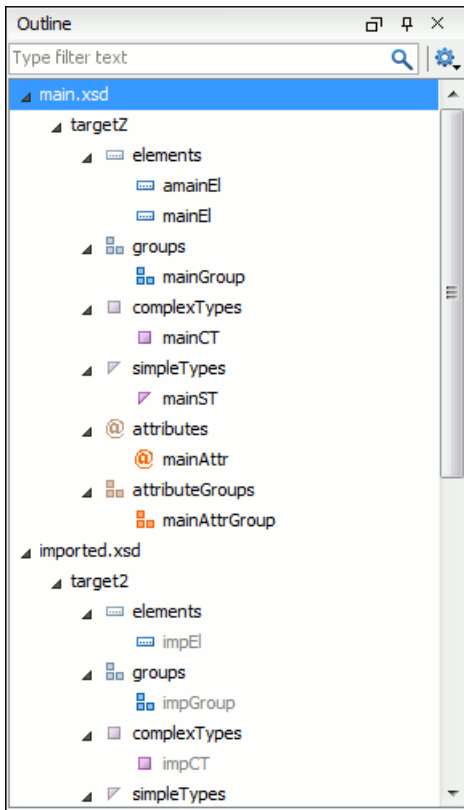


Figure 67: Outline View for XML Schema

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 main 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 <code>componentNS</code> namespace.
Show referable components	Displays all components (collected starting from the main 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)**

Allows you to see the dependencies for the current selected component.

Resource Hierarchy (F4)

Allows you to see the hierarchy for the current selected resource.

Resource Dependencies (Shift + F4)

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 413

[Component Dependencies View for XML Schema](#) on page 411

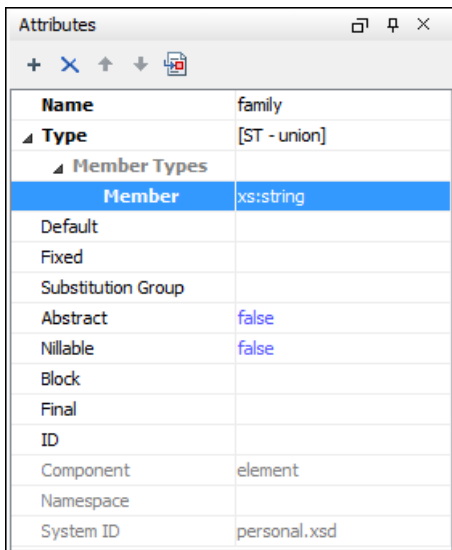
[XML Schema Resource Hierarchy / Dependencies View](#) on page 409

[Generating Sample XML Files](#) on page 415

[Editing Relax NG Schema in the Master Files Context](#) on page 464

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 from the **Window > Show View** menu.



Attributes	
Name	family
Type	[ST - union]
Member Types	
Member	xs:string
Default	
Fixed	
Substitution Group	
Abstract	false
Nullable	false
Block	
Final	
ID	
Component	element
Namespace	
System ID	personal.xsd

Figure 68: 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 from the **Window > Show View** menu.

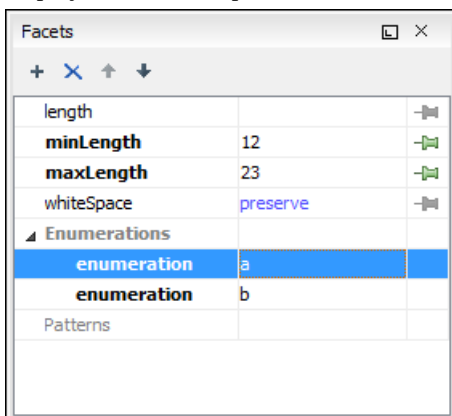


Figure 69: Facets View

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 from the **Window > Show View** menu.

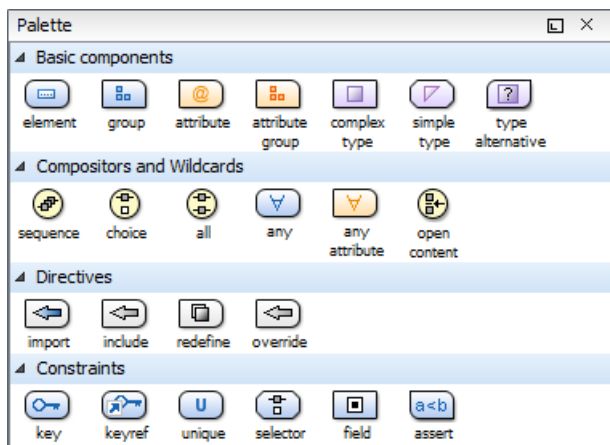


Figure 70: Palette View

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 http://oxygenxml.com/demo/Schema_Palette.html and http://oxygenxml.com/demo/Developing_XML_Schemas.html respectively.

Chapter

7

Editing Documents

Topics:

- [Working with Unicode](#)
- [Creating and Working with 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](#)
- [Spell Checking](#)
- [Loading Large Documents](#)
- [Scratch Buffer](#)
- [Handling Read-Only Files](#)
- [Editing Documents with Long Lines](#)
- [XML Digital Signatures](#)
- [Compare Files or Directories](#)

This chapter explains the editor types available in Oxygen XML Developer and how to work with them for editing various types of documents.

This chapter explains the editor types available in Oxygen XML Developer and how to work with them for editing various types of documents.

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 Developer 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 Developer XML Editor uses 16 bit characters covering the Unicode Character set.



Note: Oxygen XML Developer 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 Unicode Documents

When loading documents, Oxygen XML Developer 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 Developer 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 Developer applies [the policy specified for handling such errors](#). If the policy is set to REPORT, Oxygen XML Developer 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 Developer detects the problem and signals it to the user. The user is responsible to resolve the conflict before saving the document.

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 *Internet Explorer* or *Word* to examine XML documents.

When a document with a UTF-16 encoding is edited and saved in Oxygen XML Developer, 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.

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 [enabled the Symbols toolbar](#), 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.

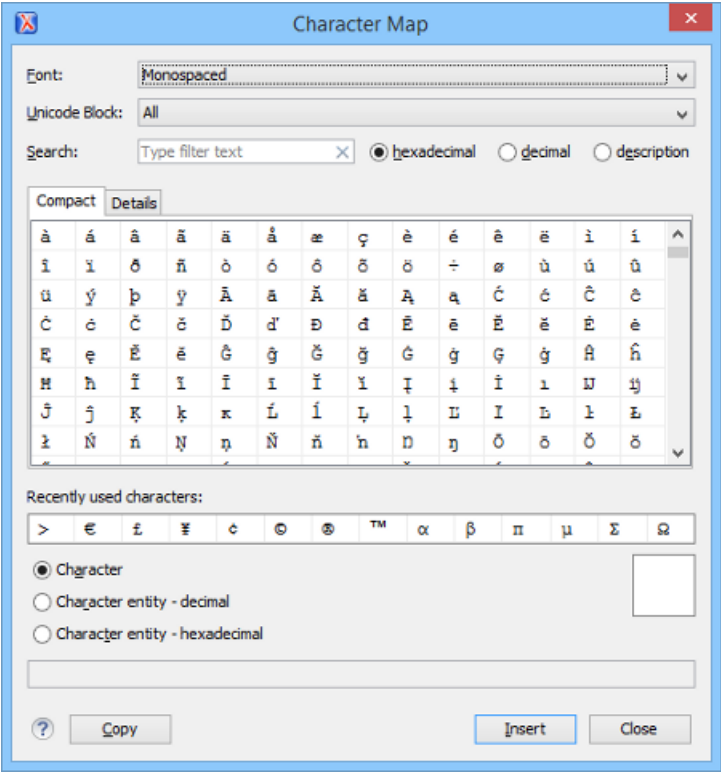


Figure 71: Character Map Dialog Box

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, **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 your document. 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 your document next to the selections in this section. 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 Developer 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 Developer 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 Developer 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 Developer. To allow Oxygen XML Developer 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 Developer 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 Developer for the changes to take full effect.

Result: Whenever Oxygen XML Developer 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 Developer, *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 Developer for the font changes to take full effect.

Creating and Working with Documents

Oxygen XML Developer 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 Developer 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 Developer 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 Developer, follow these steps:

1. Click the  **New** button on the toolbar or select **File > New**.

Result: The **New Document** wizard is displayed and it groups the supported document types 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 Developer 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.
3. 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.

4. 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

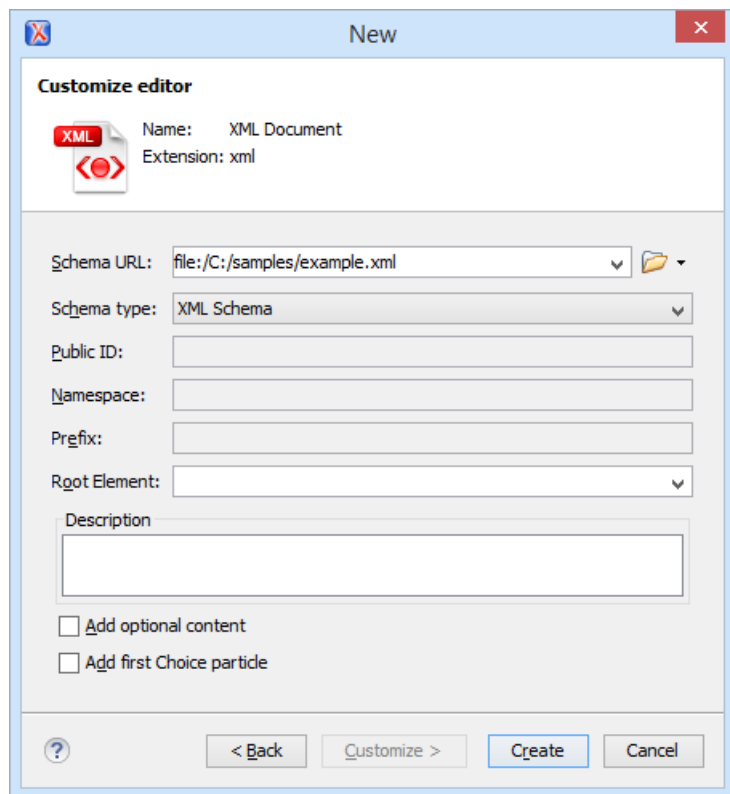


Figure 72: New XML Document Configuration Dialog Box

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 Developer 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 Developer generates the first element of an `xs:choice` schema element in the skeleton XML document. Oxygen XML Developer creates this document in a new editor panel when you click **OK**.

XSLT Stylesheet File Type

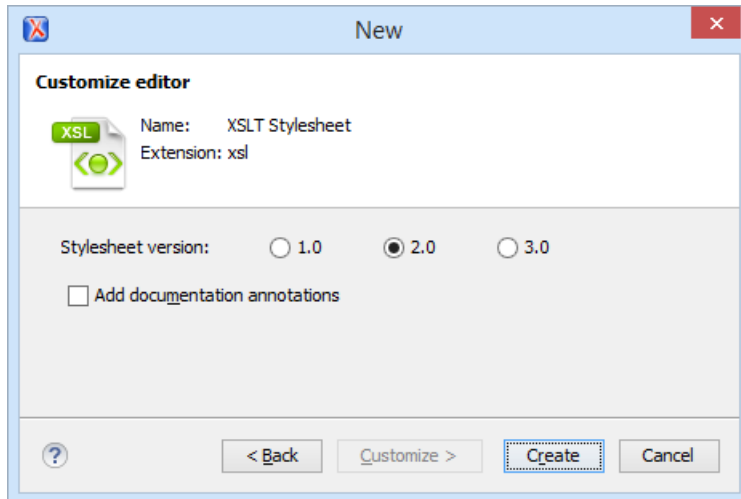


Figure 73: New XSLT Stylesheet Configuration Dialog Box

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** - Enable this option to generate the stylesheet annotation documentation.

XML Schema File Type

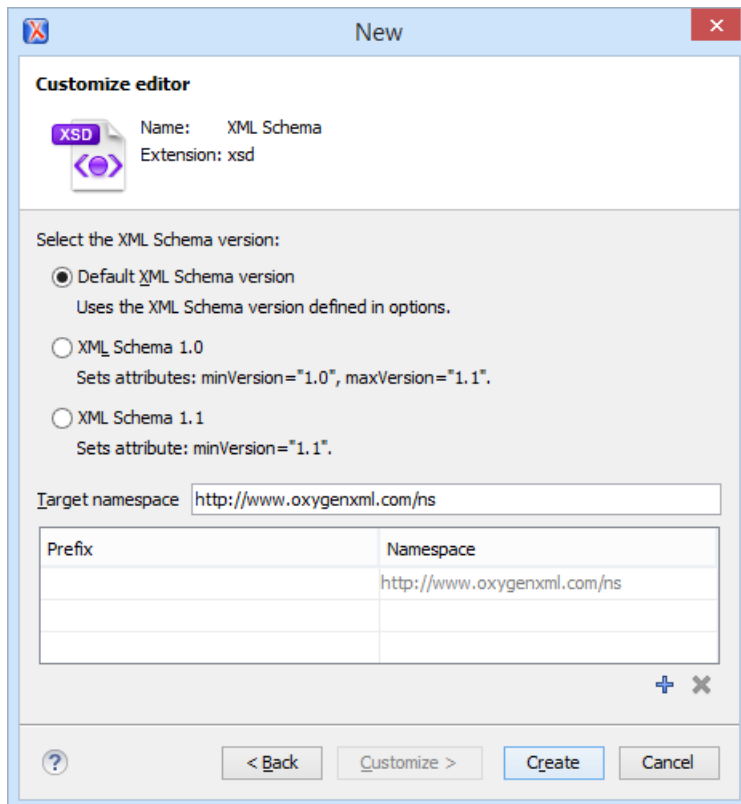


Figure 74: New XML Schema Configuration Dialog Box

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

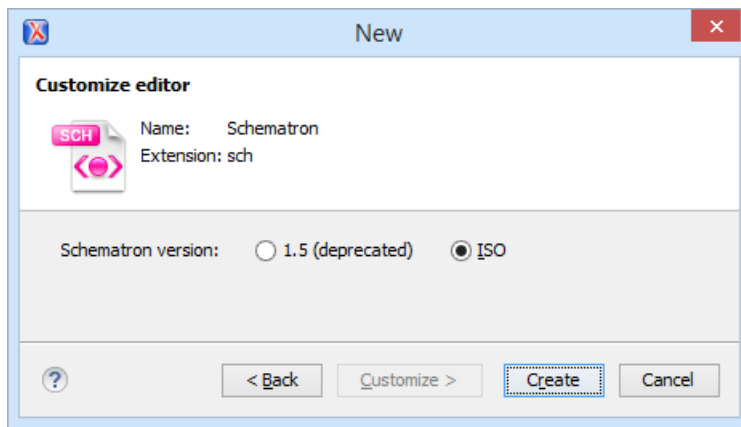


Figure 75: New Schematron Configuration Dialog Box

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 Developer, the support for Schematron 1.5 is deprecated. It is recommended to use ISO Schematron instead.

Creating New Document Templates

Oxygen XML Developer 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 Developer 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, 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 213

Customizing Document Templates

Oxygen XML Developer 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 Developer 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. 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.

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.

Related information

[Creating New Document Templates](#) on page 212

Opening Documents

To open a document in Oxygen XML Developer, 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** 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 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 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 using FTP/SFTP dialog box](#) that allows you to open a remotely stored document.
 - ** Browse for archived file** - Displays an **Archive Browser** dialog box that allows you to browse the content of an archive and choose a file to open in Oxygen XML Developer.
 - ** 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 **Open URL** dialog box.
- ** Search for file** - Displays the [Open/Find Resource dialog box](#).
 - Click the ** Open/Find Resource** toolbar button to run the same action.

- 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** action from the contextual menu of the **Project view**. This opens the selected file from the **Project** view.

Related information

[Opening Local Files at Start-up from Command Line](#) on page 215

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 from Command Line

To open a local file at start-up when you open Oxygen XML Developer from the command line, add the paths for one or more local files as parameters in the command line:

- `scriptName [pathToXMLFile1] [pathToXMLFile2]` where `scriptName` is the name of the startup script for your platform (`oxygenDeveloper.bat` on Windows, `oxygenDeveloper.sh` on Unix/Linux, `oxygenDeveloperMac.sh` on Mac OS) and `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
```

where `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). The next parameter, `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.

Opening a File at a Specific Position Using the Command Line Interface

Oxygen XML Developer offers support for opening a file at a specific position using the command line interface, by transmitting parameters to the Oxygen XML Developer 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 Developer 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 Developer:

```
developer.bat file:samples/personal.xml#line=4
developer.bat file:samples/personal.xml#line=4column=5
```

```
developer.bat file:samples/personal.xml#line=4;column=5
developer.bat file:samples/personal.xml#char=334
```

2. Simplified XPath index path

Oxygen XML Developer 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 Developer 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 Developer and select the third child of the root element:

```
developer.bat file:samples/personal.xml#element(1/3)
```

3. Anchors identified by ID attribute values

Oxygen XML Developer 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 Developer and select the element that has the `id` element set to `titleID`:

```
developer.bat file:samples/personal.xml#titleID
```

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** dialog box 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 via FTP/SFTP/WebDAV/SharePoint

Oxygen XML Developer 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 a project, or make them subject of 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 Developer by checking 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 Developer 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 Developer.

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 Developer and the HTTP / WebDAV server is compressed using the GZIP algorithm.

The current [WebDAV Connection](#) details can be saved using the  **Database Perspective** button and then used in the *Data Source Explorer* view.

Open URL Dialog Box

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.

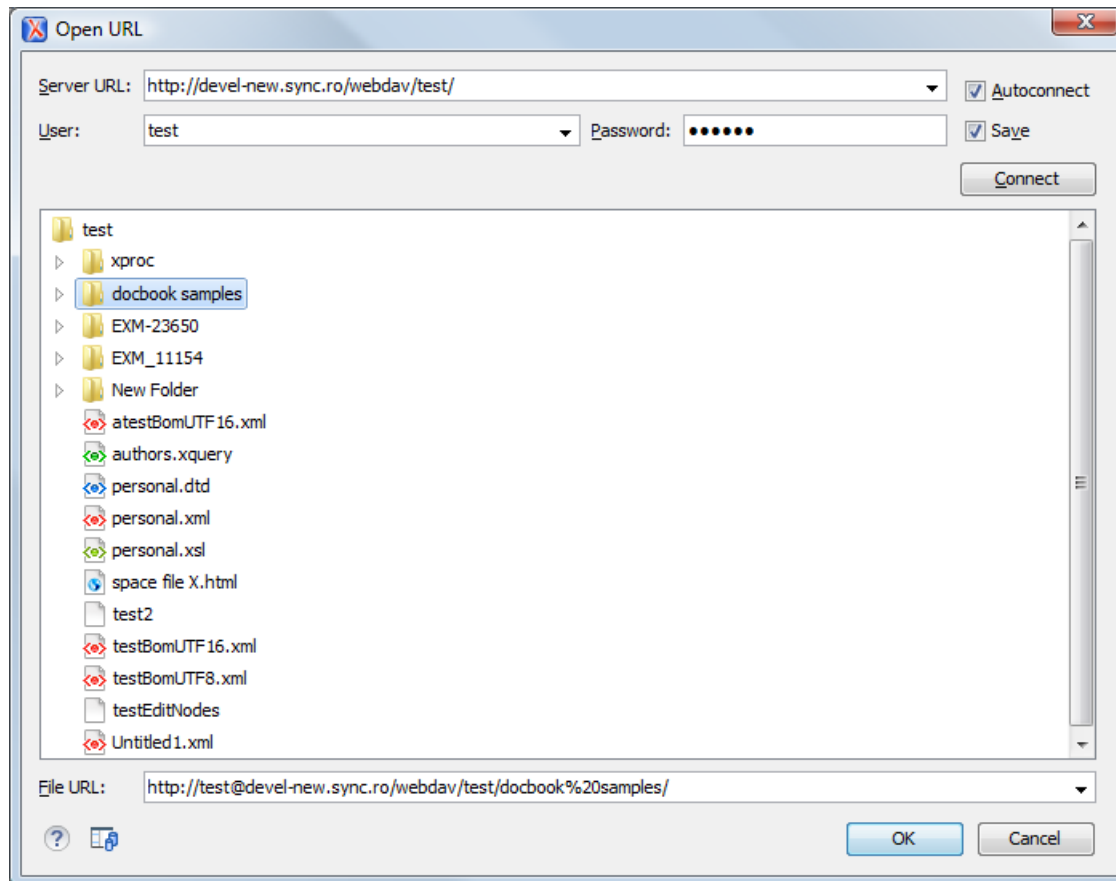


Figure 76: Open URL Dialog Box

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.

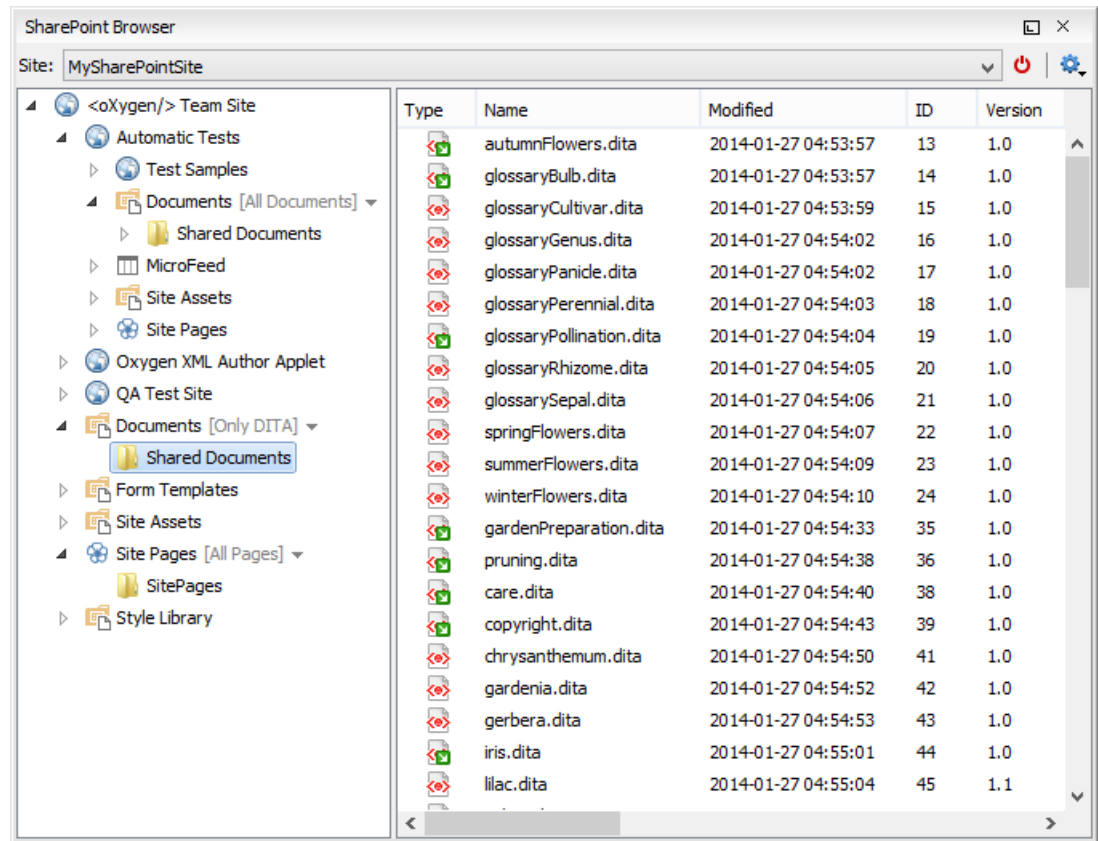


Figure 77: Browsing a SharePoint Repository

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.

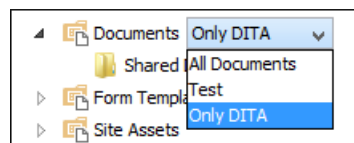


Figure 78: Drop-Down Menu to Select Which Items to Display

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.

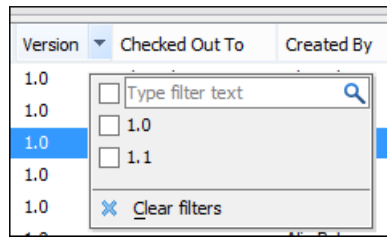


Figure 79: Column Filter

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 check boxes.

WebDAV over HTTPS

If you want to access a WebDAV repository across an unsecure network, Oxygen XML Developer 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 Developer 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 Developer. This means that Oxygen XML Developer 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 Developer cannot connect to an HTTPS-capable server, most likely there is no certificate set in the *Java Runtime Environment (JRE)* that Oxygen XML Developer 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 Developer.

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.

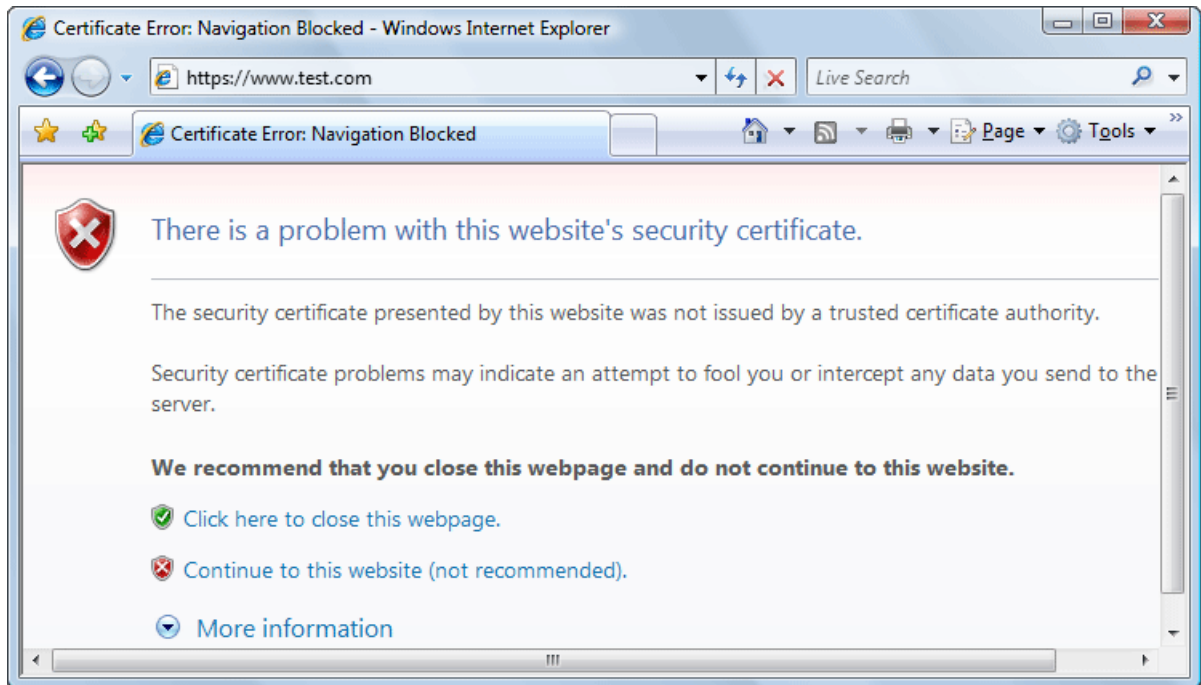


Figure 80: Security alert - untrusted certificate

- 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 Developer.
- a) Open a text-mode console with administrative rights.
If Oxygen XML Developer 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 Developer. 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 Developer 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 Developer 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.



Note: To make Oxygen XML Developer accept a certificate even if it is invalid, [open the Preferences dialog box \(Options > Preferences\)](#), go to **Connection settings > HTTP(S)/WebDAV**, and enable the **Automatically accept a security certificate, even if invalid** option.



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 Developer.

Related information

[HTTP\(S\)/WebDAV Preferences](#) on page 143

HTTP Authentication Schemes

Oxygen XML Developer 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 Developer 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 Developer 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 in the Launcher Configuration File / Start-up Script](#) on page 162

Switching Between Opened Tabs

There are two actions 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.

Searching Documents

Oxygen XML Developer 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 Developer layout, next to the **Project** and **DITA Maps Manager** views. If the view is not displayed, it can be opened from the **Window > Show View** menu.

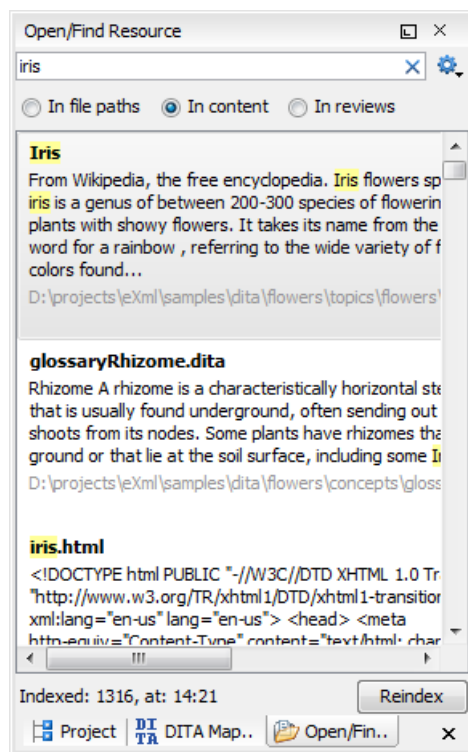


Figure 81: Open/Find Resource View

You can use this view to find a file in the current Oxygen XML Developer project 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).



Note: Full support for searching in document edits is available only in the Enterprise edition of Oxygen XML Developer. The Professional edition offers 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 *samp les* 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](#).
- **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 Developer 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 is 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 Developer 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 http://oxygenxml.com/demo/Open_Find_Resource.html.

Related information

[Open/Find Resource Dialog Box](#) on page 225

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.

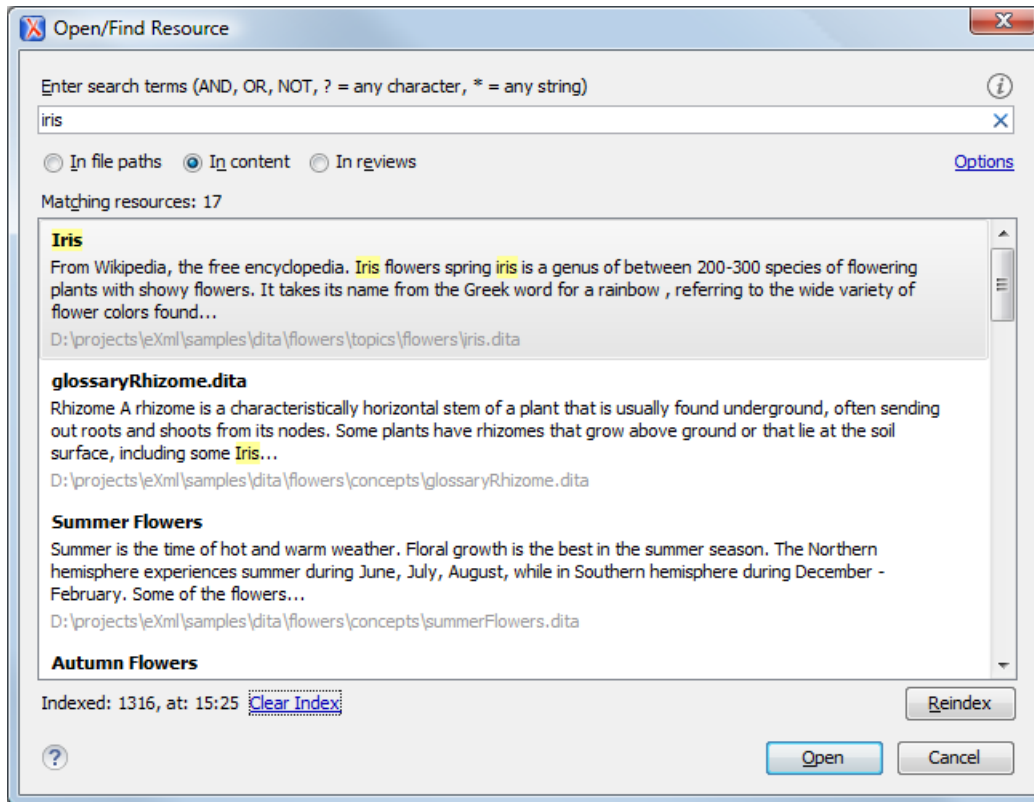


Figure 82: Open/Find Resource Dialog Box

You can use this dialog box to find a file in the current Oxygen XML Developer project 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 is available only in the Enterprise edition of Oxygen XML Developer. The Professional edition offers 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](#).
- **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 Developer 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 is 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 Developer 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 http://oxygenxml.com/demo/Open_Find_Resource.html.

Related information

[Open/Find Resource View](#) on page 180

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:

Term Searches

Use the **Open/Find Resource** view or dialog box to search for plain text:

```
Garden Preparation
```

Element-Specific Searches

Use the **Open/Find Resource** view or dialog box to search for content that belongs to a specific element:

```
title:"Garden Preparation"
```

Wildcard Searches

Use 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 what you introduce in the **Open/Find Resource** view or dialog box. 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 prioritise the word *Autumn* since the word *Flowers* occurs more often. To prioritise 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 disable the **Show Description** option.

Technical Aspects

When Oxygen XML Developer 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.developer\lucene` folder.

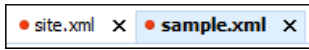
Closing Documents

To close open documents, use one of the following methods:

- Click **Close (Ctrl + W (Command + W on OS X))** in the contextual menu of an open tab (or from the **File** menu) to close it.
- Click **Close Other Files** in the contextual menu of an open tab (or from the **File** menu) to close all the open tabs except the selected one.
- Click **Close All (Ctrl + Shift + F4 (Command + Shift + F4 on OS X))** in the contextual menu of an open tab (or from the **File** menu) to close all open 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	Closes the currently selected editor.
Close Other Files	If multiple files are opened, this action is available to close all opened editors 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.
Reopen last closed editor	Reopens the last closed editor.
Maximize/Restore Editor Area	Collapses all the side views and spans the editing area 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 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, you can quickly access 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.

Using Projects to Group Documents

This section explains how to create and work with Projects.

Creating a New Project

Oxygen XML Developer 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 much more information on managing projects and their content, see the [Project View](#) on page 174 section.

For more details about how you can share projects with other users, see the [Sharing a Project - Team Collaboration](#) on page 240 section.

Related information

[Using Projects to Group Documents](#) on page 231

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 on the contextual menu and toolbar associated to this panel enable the creation of XML projects and shortcuts to various operations on the project documents.

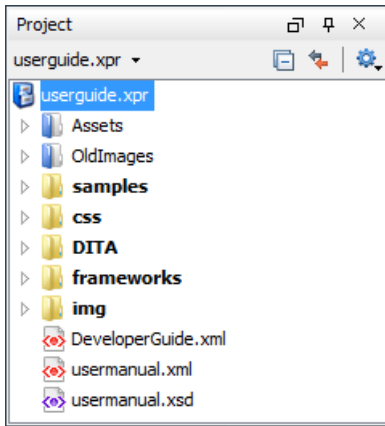


Figure 83: Project View

By default, the view is positioned on the left side of the Oxygen XML Developer 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 Developer XPR project file from the file explorer into the **Project** panel.



Note: 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 Developer 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.

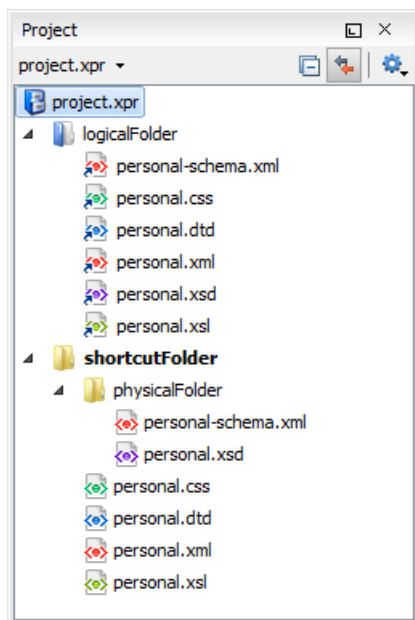


Figure 84: Project View with Examples of all Three Types of Resources

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.

Add Content to a Logical Folder

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.

Managing Project Content

Creating/Adding Files and 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 enabled 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.



Note: The linked files presented in the **Project** view are marked with a special icon. Linked folders are displayed in bold text.

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 Developer 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 oXygen consoles** option must be enabled 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.

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 Developer 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 are used 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: *Archive Browser*, *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** 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 Developer XPR project file from the file explorer into the **Project** panel.



Note: 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.

Related information

[Working with EPUB](#) on page 697

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 or use the  **Cut** and  **Paste** actions from the contextual menu or **Edit** menu (you can also use the  **Copy** and  **Paste** actions from the contextual menu or **Edit** menu to copy resources to a new location).

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)** - Enable 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** - Enable 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 case 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 Developer displays a warning dialog box.

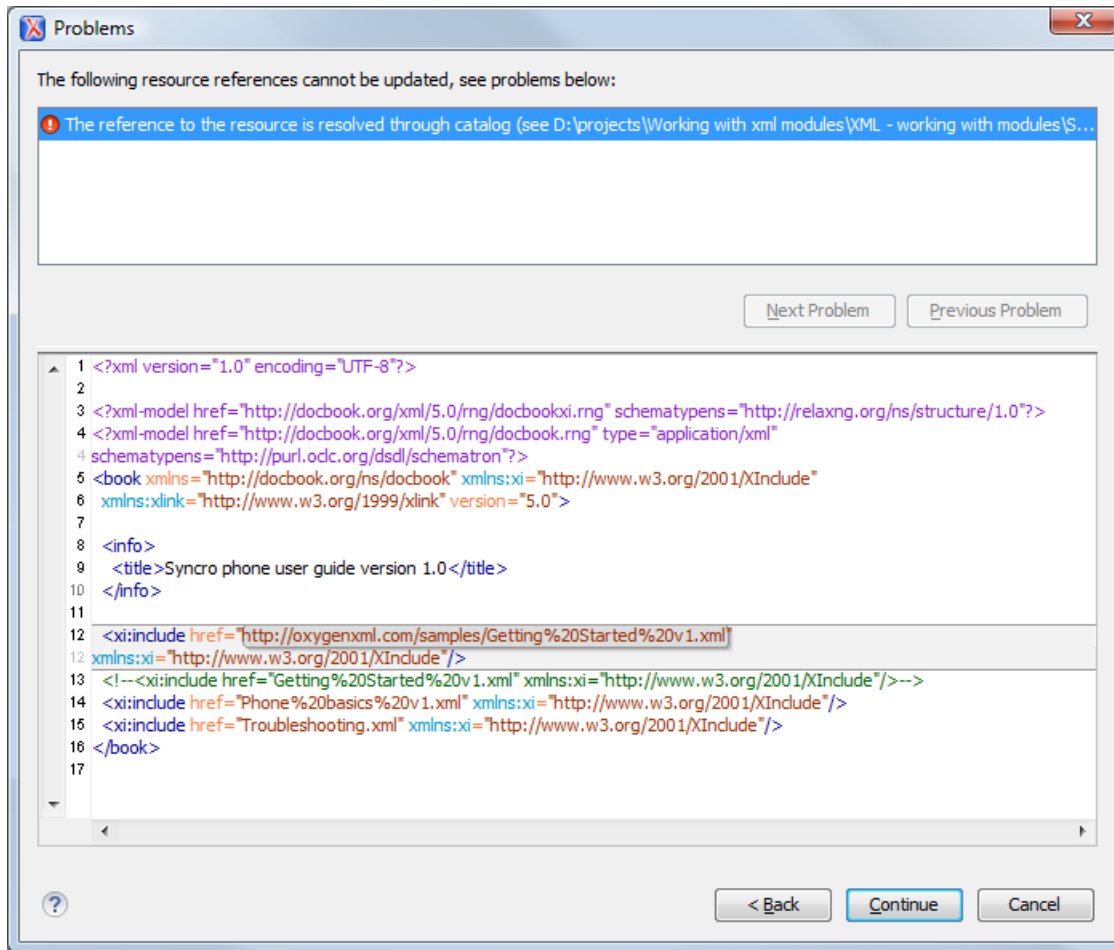


Figure 85: Problems Dialog Box

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 **Project** view's contextual menu. 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 Developer also allows you to store preferences, transformation scenarios, and validation scenarios 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 create a project-level options file by storing them in a *project file* (.xpr file extension). To do so, follow these steps:

1. You may 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** switch, 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.



Note: 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.



Note: 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.



Note: 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 Developer 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 and Storing Options](#) on page 148

[Sharing the Transformation Scenarios](#) on page 600

[Sharing Validation Scenarios](#) on page 292

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 Developer project file 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.

Defining Master Files at Project Level

This chapter details the *Master Files Support* available in Oxygen XML Developer. 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.

Oxygen XML Developer allows you to define *master files* at project level. These *master files* are automatically used by Oxygen XML Developer to determine the context for operations such as validation, content completion, refactoring, or search for XML, XSD, XSL, WSDL, and RNG modules. Oxygen XML Developer maintains the hierarchy of the *master files*, helping you to determine the editing context.

To watch our video demonstration about the **Master Files Support** for XML documents, XSL documents, and WSDL documents, go to [Working with Modular XML Files](#), [Master Files Support](#), and [Working with Modular WSDL Files](#), respectively.

Master Files Benefits

Using the *Master Files* support in Oxygen XML Developer includes the following benefits:

- When the module is validated, Oxygen XML Developer 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 Developer 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 Developer 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. To enable it, select **Enable Master Files Support** from the  **Settings** menu in the top-right corner of the **Project** view. You can also select **Enable Master Files Support** from the contextual menu of the **Master Files** directory, or from the contextual menu of the project itself. Oxygen XML Developer allows you to enable or disable the *Master Files Support* for each project you are working on.

When the **Master Files Support** is disabled, Oxygen XML Developer displays a window tip located at the bottom of the project view. This window contains an **Enable** button and a **Read more** link. 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** option to detect the *master files* from the current project.

 **Warning:** Once you close this window tip, Oxygen XML Developer hides it for all projects. You can make the window tip reappear by [resetting Oxygen XML Developer to its default settings](#). However, doing so will result in you losing your customized options.

Detecting Master Files

Oxygen XML Developer 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 Developer 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*.
- **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 enable their check-boxes. Oxygen XML Developer 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*.

 **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*.

Adding or Removing a Master File

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 in the **Properties** dialog of the **Project** view and the *master files* for the current editor in the **Properties** and **Information** views.

Project Validation and Transformation

The *Master Files Support* is also useful for project-level validation and transformation. When you hover the cursor over a file in the **Master Files** directory, Oxygen XML Developer 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 Developer executes a batch transformation and validation. If the current node is not selected, Oxygen XML Developer 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 Developer 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 <i>master files</i> 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 <i>Find/Replace dialog box</i> .
//*XPath in Files	Opens the <i>XPath/XQuery Builder view</i> that allows you to compose XPath and XQuery expressions and execute them over the currently edited XML document.
Open/Find Resource	Opens the <i>Open/Find Resource dialog box</i> .
Check Spelling in Files	Opens the <i>Check Spelling in Files dialog box</i> .

 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 Developer performs the validation of the *master files*.

Validate with Schema

Opens the **Validate with** dialog box. Oxygen XML Developer 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 Developer 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 172

[Grid Editing Mode](#) on page 192

Editing XML Documents in Text Mode

This section includes features and actions for editing XML documents in the **Text** mode of Oxygen XML Developer.

Navigating the Document Content in Text Mode

Oxygen XML Developer includes some useful features to help you navigate XML documents in **Text** mode.

Using the Keyboard

Oxygen XML Developer 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 Developer 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. Enabled after at least one press of the **Back** button.

Navigating with the Outline View

Oxygen XML Developer 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.

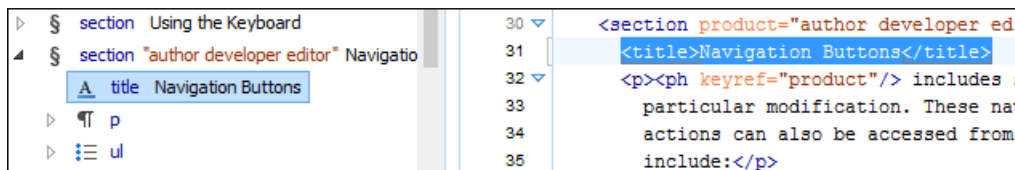


Figure 86: Outline View Navigation in Text Mode

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.

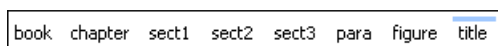


Figure 87: Breadcrumb in Text Mode

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))**.

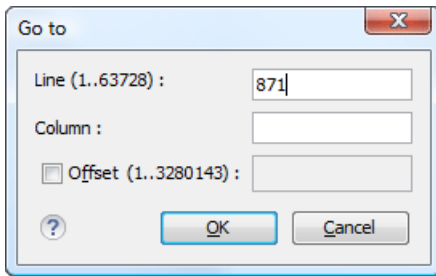


Figure 88: Go to Dialog Box

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.

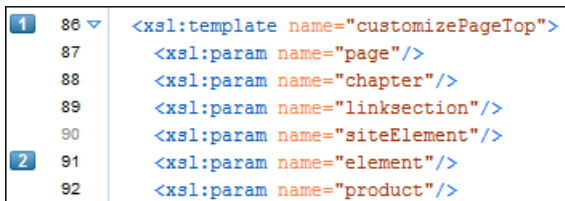


Figure 89: Editor Bookmarks

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 Developer 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 Developer 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.

Editing Content in Text Mode

Oxygen XML Developer includes numerous shortcut actions to help you edit content in the **Text** editing mode.

Changing the Font Size

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 (Command + MouseWheelForward on OS X))	Increases the font size with one point for each execution of the action.
Decrease editor font (Ctrl + NumPad- (Command + NumPad- on OS X) or Ctrl + MouseWheelBackwards (Command + MouseWheelBackwards on OS X))	Decreases the font size with one point for each execution of the action.
Normal editor font (Ctrl + 0 (Command + 0 on OS X))	Resets the font size to <i>the value of the editor font set in the Fonts preferences page.</i>

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.

Miscellaneous Shortcut Actions in Text Mode

Oxygen XML Developer 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.

Editing XML Markup in Text Mode

Oxygen XML Developer 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.

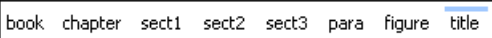


Figure 90: Breadcrumb in Text Mode

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:

<u>Alt + UpArrow</u>	Moves the current node or selected nodes in front of the previous node.
<u>Alt + DownArrow</u>	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. <i>For example</i>, to change the <code>xmlns:p1="ns1"</code> association in the current element to <code>xmlns:p5="ns1"</code>, if the <code>xmlns:p1="ns1"</code> association is applied on the parent element, then Oxygen XML Developer will introduce <code>xmlns:p5="ns1"</code> as a new declaration in the current element and will change the prefix from <code>p1</code> to <code>p5</code>. If <code>p5</code> 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 <code>p1</code> to <code>p5</code> 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, check 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 <i>Position cursor between tags option</i> is enabled in the Content Completion preferences page, the cursor is placed between the start and end tag. • If the <i>Position cursor between tags option</i> is disabled 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 (or from the **Document > Markup** menu):

Attributes submenu Contains predefined XML refactoring operations that pertain to attributes. Oxygen XML Developer considers the editing context to get the names and namespaces of the element or attribute for which the contextual menu was invoked, and uses this information to preconfigure some of the parameter values for the selected refactoring operation.

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.

Elements submenu Contains predefined XML refactoring operations that pertain to elements. Oxygen XML Developer considers the editing context to get the names and namespaces of the element or attribute for which the contextual menu was invoked, and uses this information to preconfigure some of the parameter values for the selected refactoring operation.

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. Oxygen XML Developer considers the editing context to get the names and namespaces of the element or attribute for which the contextual menu was invoked, and uses this information to preconfigure some of the parameter values for the selected refactoring operation.
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.

Related information

[Refactoring XML Documents](#) on page 319

[Contextual Menu Actions in Text Mode](#) on page 270

Folding XML Elements in Text Mode

When working with a large document, the **folding support** in Oxygen XML Developer 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 Developer, but it can be disabled in the **Text** preferences page with the [Enable folding option](#).



Figure 91: Folding of XML Elements in Text Mode

The fact that the folds are persistent is a unique feature of Oxygen XML Developer. 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 (<u>C</u> trl + <u>N</u> umPad/ (<u>C</u> ommand + <u>N</u> umPad/ on OS X))	Folds all the elements except the current element.
 Collapse Child Folds (<u>C</u> trl + <u>N</u> umPad. (<u>C</u> ommand + <u>N</u> umPad. 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 (<u>C</u> trl + <u>N</u> umPad* (<u>C</u> ommand + <u>N</u> umPad* on OS X))	Unfolds all elements in the current document.

To watch our video demonstration about the folding support in Oxygen XML Developer, go to <http://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 Developer 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

<u>Ctrl + A (Meta + A on Mac OS X)</u>	Selects all content in the document.
<u>Shift + Left/Right Arrow Keys</u>	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.
<u>Shift + Up/Down Arrow Keys</u>	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.
<u>Ctrl + Shift + Left/Right Arrow Keys (Meta + Shift + Left/Right Arrow Keys on Mac OS X)</u>	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.
<u>Shift + Home</u>	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).
<u>Shift + End</u>	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).
<u>Ctrl + Shift + Home</u>	Begins a continuous selection at the cursor position and extends it to the beginning of the document.
<u>Ctrl + Shift + End</u>	Begins a continuous selection at the cursor position and extends it to the end of the document.
<u>Shift + PageUp</u>	Begins a continuous selection at the cursor position and extends it up one screen page.
<u>Shift + PageDown</u>	Begins a continuous selection at the cursor position and extends it down one screen page.
<u>Double-Click</u>	Selects certain text, depending on the position of the click in the document. See Smart Editing Double-Click for the specifics.
<u>Triple-Click</u>	Selects entire regions of text, depending on the position of the click in the document. See the Smart Editing Triple-Click for the specifics.

Rectangular Selection Shortcuts

Oxygen XML Developer also includes some keyboard shortcuts that allows 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 enabled in the **Text** preferences page.

The following shortcuts can be used to create a rectangular selection:

<u>Shift + Alt + Left/Right Arrow Keys (Shift + Alt + Meta + Left/Right Arrow Keys on Mac OS X)</u>	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).
<u>Shift + Alt + Up/Down Arrow Keys (Shift + Alt + Meta + Up/Down Arrow Keys on Mac OS X)</u>	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).
<u>Ctrl + Shift + Alt + Left/Right Arrow Keys (Ctrl + Shift + Alt + Meta + Left/Right Arrow Keys on Mac OS X)</u>	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.
<u>Shift + Alt + Home (Shift + Alt + Meta + Home on Mac OS X)</u>	Begins a rectangular selection at the cursor position and extends it to the beginning of the current line.
<u>Shift + Alt + End (Shift + Alt + Meta + End on Mac OS X)</u>	Begins a rectangular selection at the cursor position and extends it to the end of the current line.
<u>Shift + Alt + PageUp (Shift + Alt + Meta + PageUp on Mac OS X)</u>	Begins a rectangular selection at the cursor position and extends it up one screen page.
<u>Shift + Alt + PageDown (Shift + Alt + Meta + PageDown on Mac OS X)</u>	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.

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 Developer includes an intelligent **Content Completion Assistant** that offers rapid, in-line identification and insertion of structured language elements, attributes, and attribute values. Oxygen XML Developer shows the available entries that are valid in the current editing context.

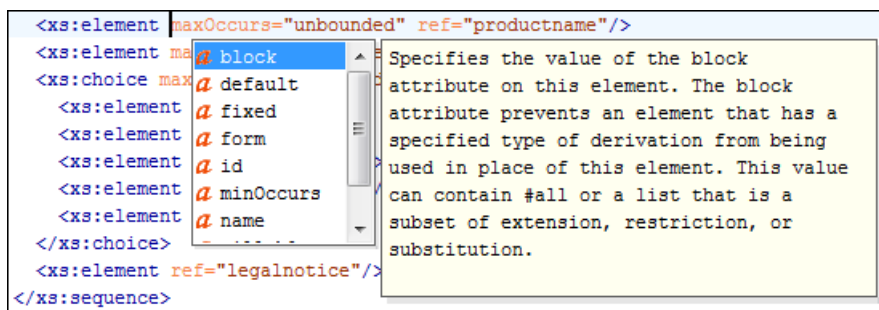


Figure 92: Content Completion Assistant

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 disable 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 $\leq$ 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 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.

When active, the **Content Completion Assistant** displays a list of context-sensitive proposals valid at the current cursor position. Elements can be selected in the list 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 enabled. 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 enable 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 enable 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 items 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 Developer 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.

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 Developer 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.
- For XSLT stylesheets, the schema specified in the Oxygen XML Developer [Content Completion options](#). Oxygen XML Developer 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 Developer [Content Completion options](#). Oxygen XML Developer 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.

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 disabled, 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 Developer 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, enable the [Use all Relax NG annotations as documentation](#) option in the **Annotations** preferences page.

Collecting Annotation from DTDs

For DTD, Oxygen XML Developer defines a custom mechanism for annotations using comments enabled from 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 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.

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 from the **Window > Show View** menu.

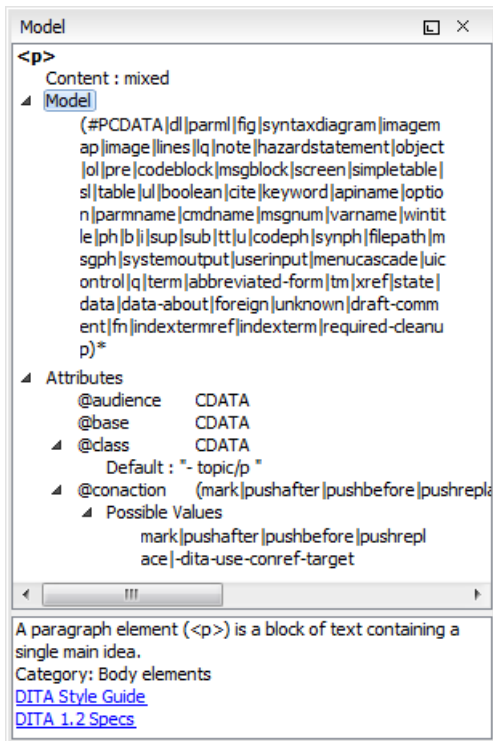


Figure 93: Model View

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.

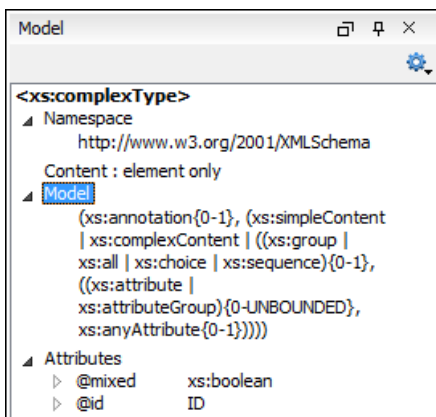


Figure 94: Element Structure Panel

Annotation Panel

The **Annotation** panel displays the annotation information for the currently selected element. This information is collected from the XML schema.

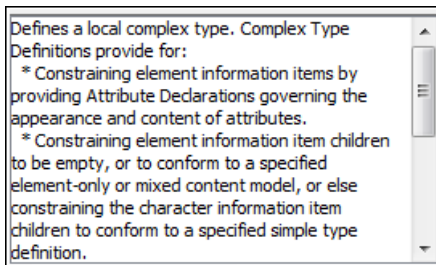


Figure 95: Annotation panel

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 with a specified value are rendered with a bold font, and their values with a plain font.



Note: The names of the attributes with an empty string value are also rendered bold.

- 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.

Attribute	Value
abstract	false
block	false
default	true
final	
fixed	
id	
name	email
nillable	false
substitutionGroup	
type	xs:string

Figure 96: 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.
- 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 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 97: 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 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.

Name	Value
lt	<
gt	>
amp	&
apos	'
quot	"
abbrev-d-att	(topic abbrev-d)
concept-att	(topic concept)
hazard-d-att	(topic hazard-d)
hi-d-att	(topic hi-d)
included-domains	&concept-att; ...
indexing-d-att	(topic indexing-d)
nbsp	
pr-d-att	(topic pr-d)
sw-d-att	(topic sw-d)
ui-d-att	(topic ui-d)
ut-d-att	(topic ut-d)

Figure 98: 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 Developer 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 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 http://oxygenxml.com/demo/Code_Templates.html.

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 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.

The drag and drop actions in the **Outline** view can be [disabled and enabled from a Preferences page](#).

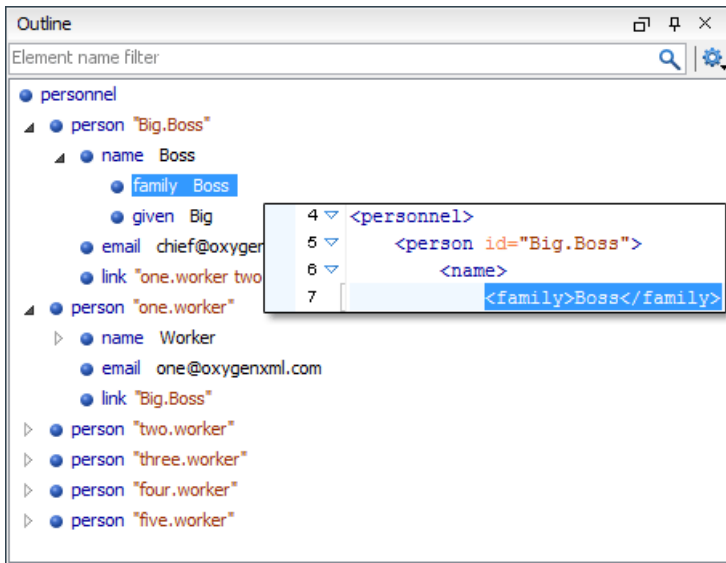


Figure 99: Outline View in Text Mode

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.

Quick Assist Support for IDs and IDREFS

The Quick Assist support is activated automatically when you place the cursor inside an ID or an IDREF. 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 <i>choose the scope of the rename operation</i> . 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 <i>Select the scope for the Search and Refactor operations</i> 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 <i>Select the scope for the Search and Refactor operations</i> dialog box, this scope will be used instead.
 Change scope	Opens the <i>Select the scope for the Search and Refactor operations</i> 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 and 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 308

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 Developer 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.

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 disabled/enabled with the  **Lock / Unlock the XML tags** action available in **Text** editing mode from:

- The **Document > Source** menu.
- The **Source** submenu from the contextual 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 Developer 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 Developer must decide how to format and indent the XML. Oxygen XML Developer 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 Developer 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 or defer 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. Thus, the paragraph above could be written all on one line with no spaces between the start tag and the first word or between the last word and the end tag and the XML parser would see it as exactly the same paragraph. 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 Developer determines when whitespace is significant

When Oxygen XML Developer 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 Developer 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 Developer 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. This generally applies to elements that contain text content only. This content can be normalized by removing insignificant whitespace. Insignificant whitespace may then be added to format and indent the content.
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 s <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 Developer 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 Developer 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 Developer 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.

Schema aware formatting If a schema is available for the XML document, Oxygen XML Developer 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 the **Text** editing mode, [open the Preferences dialog box \(Options > Preferences\)](#), go to **Editor > Format > XML**, and enable/disable 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.

How Oxygen XML Developer formats and indents XML

You can control how Oxygen XML Developer 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](#) on page 79
- [XML Formatting Preferences](#) on page 80
- [Whitespaces Preferences](#) on page 82

When Oxygen XML Developer formats and indents XML

Oxygen XML Developer 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 **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 Developer 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 Developer 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 Developer 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 Developer 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 Developer 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 Developer provides support for formatting and indenting (*pretty printing*) 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.

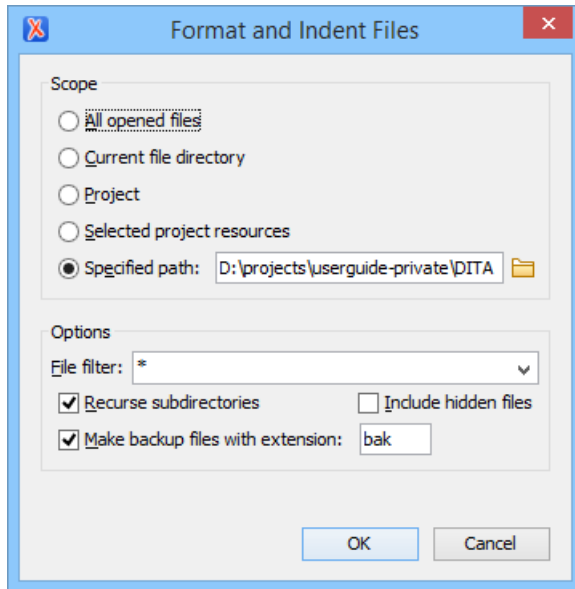


Figure 100: Format and Indent Files Dialog Box

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** - *Pretty prints* 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 enabled, 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 enabled, the *pretty print* is also performed in the hidden files.
- **Make backup files with extension** - When enabled, Oxygen XML Developer 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 Developer 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.

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](#) on page 308

Contextual Menu Actions in Text Mode

When editing XML documents in **Text** mode, Oxygen XML Developer provides the following actions in the contextual menu (many of them also appear in the submenus of the **Document** menu):



Cut,

Executes the typical editing actions on the currently selected content.



Copy,



Paste

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 of the 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 <i>indenting preferences</i> .
 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 <i>base64</i> 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 disabled 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 disabled in the Messages preference page](#).

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 disabled 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 disabled in the Messages preference page](#).

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 disabled 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 disabled in the Messages preference page](#).

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 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.	
 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. <i>For example</i>, to change the <code>xmlns:p1="ns1"</code> association in the current element to <code>xmlns:p5="ns1"</code>, if the <code>xmlns:p1="ns1"</code> association is applied on the parent element, then Oxygen XML Developer will introduce <code>xmlns:p5="ns1"</code> as a new declaration in the current element and will change the prefix from <code>p1</code> to <code>p5</code>. If <code>p5</code> 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, check 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 enabled in the **Content Completion** preferences page, the cursor is placed between the start and end tag.
- If the *Position cursor between tags option* is disabled 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. Oxygen XML Developer considers the editing context to get the names and namespaces of the element or attribute for which the contextual menu was invoked, and uses this information to preconfigure some of the parameter values for the selected refactoring operation.

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.

Elements submenu

Contains predefined XML refactoring operations that pertain to elements. Oxygen XML Developer considers the editing context to get the names

and namespaces of the element or attribute for which the contextual menu was invoked, and uses this information to preconfigure some of the parameter values for the selected refactoring operation.

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. Oxygen XML Developer considers the editing context to get the names and namespaces of the element or attribute for which the contextual menu was invoked, and uses this information to preconfigure some of the parameter values for the selected refactoring operation.

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.

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 <i>choose the scope of the rename operation</i> . 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 <i>Select the scope for the Search and Refactor operations</i> dialog box, this scope will be used instead.
Search References in	Searches for the references of the ID. Selecting this action opens the <i>Select the scope for the Search and Refactor operations</i> .
 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 <i>Select the scope for the Search and Refactor operations</i> dialog box, this scope will be used instead.
Search Declarations in	Searches for the declaration of the ID reference. Selecting this action opens the <i>Select the scope for the Search and Refactor operations</i> .

**Search Occurrences in file**

Searches for the declaration and 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

This section includes features and actions for editing XML documents in the **Grid** mode of Oxygen XML Developer.

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.

Start and Stop 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 Editors

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.

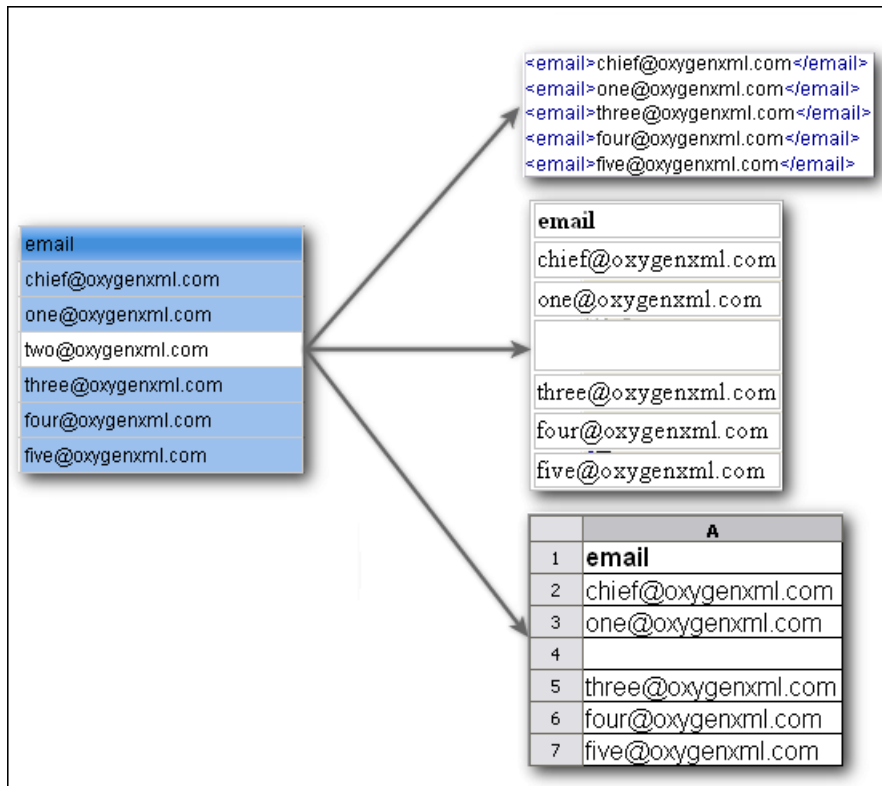


Figure 101: Copying from Grid to Other Editors

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.

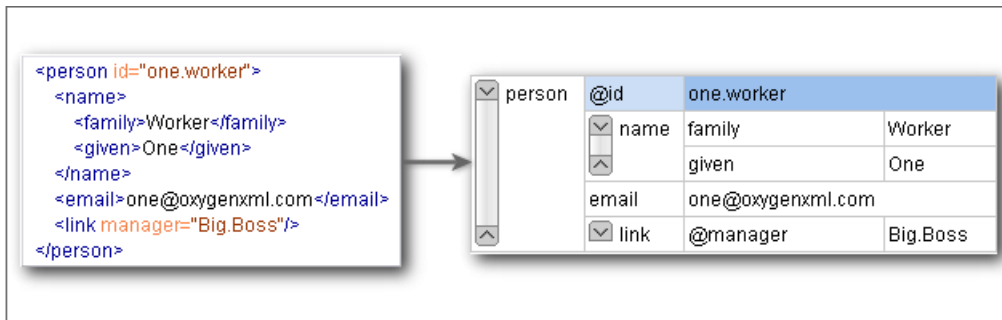


Figure 102: Pasting XML Data into Grid

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 103: Pasting Tab-Separated Values into Grid

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 Developer provides validation functions that enable you to easily identify errors and their location.

Related information

[Working with Modular XML Files in the Master Files Context](#) on page 308

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*, 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.

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>`.

A non Namespace-wellformed Document

```
<x::y></x::y>
```

When **Check document form** is performed the following error is raised:

```
Element or attribute do not match QName production:
QName ::= (NCName ':' )? NCName .
```

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

[Validate an XML Document Against Schematron](#) on page 294

Automatic Validation

Oxygen XML Developer 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. The arrow buttons of the dialog box enable the navigation to other errors issued by the Automatic Validation feature.

Related information

[Manual Validation Actions](#) on page 282

[Presenting Validation Errors in Text Mode](#) on page 191

Manual Validation Actions

You can choose to validate documents at any time by using the manual validation actions that are available in Oxygen XML Developer.

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, the **Document** > **Validate** menu, or from the **Validate** submenu when invoking the contextual menu in the **Project** view

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 Developer custom protocol plugin](#) developed independently and added to Oxygen XML Developer after installation.

Validate with Schema

Available from the **Validate** submenu when invoking contextual menu in the **Project** view

Use this action to validate all selected files with a chosen schema.

Other Validation Options

To open the schema used for validating the current document, select the **Open Associated Schema** action from the **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 Developer 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 282

[Presenting Validation Errors in Text Mode](#) on page 191

Presenting Validation Errors in Text Mode

Oxygen XML Developer can be configured to [automatically validate documents](#) while editing in the **Text** mode, and actions are also available to [manually validate documents](#) on-request.

A line with a validation error or warning is marked in the editor panel by underlining the error region with a red line, while validation warnings are underlined in yellow. Hovering over a validation error presents a tooltip message with more details about the error and [possible quick fixes](#) (if available for that error or warning).

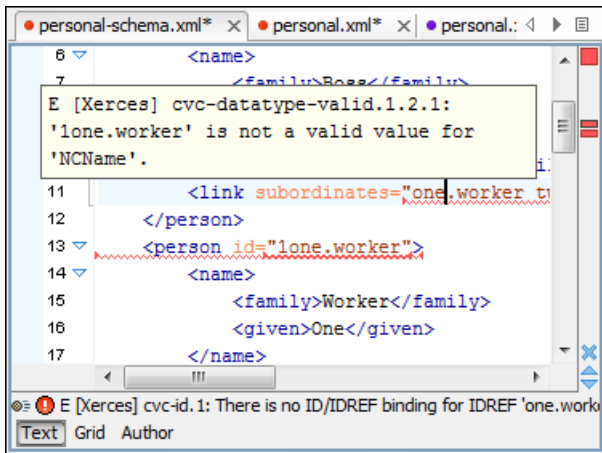


Figure 104: Presenting Validation Errors in Text Mode

Also, the ruler on the right side of the editor panel is designed to display the errors found during the validation process and to help you locate them in the document. The ruler contains the following:

Upper Part of the Ruler	A success indicator square will turn green if the validation is successful, red if validation errors are found, or yellow if validation warnings are found. More details about the errors or warnings are displayed in a tool tip when you hover over indicator square. If there are numerous errors, only the first three are presented in the tool tip.
Middle Part of the Ruler	Errors are depicted with red markers, and warnings with yellow 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 error or warning 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 Ruler	Two navigation arrows (,) allow you to skip to the next or previous error. 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 error markers.

Status messages from every validation action are logged in the [Information view](#).

If you want to see all the validation error 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 282

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 Developer for linked messages:

- **LIBXML** - Included in Oxygen XML Developer (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. XML catalogs support (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 Developer 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 Developer. 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 Developer (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 Developer (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 Developer. 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 Developer](#) 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 Developer. 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 Developer custom protocol plugin](#) developed independently and added to Oxygen XML Developer after installation.

Linked Output Messages of an External Engine

Validation engines display messages in an output view at the bottom of the Oxygen XML Developer 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 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 Scenario

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 Developer 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 Developer (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 are used and take precedence over the default scenarios defined for the particular framework. For more information on master files, see the [Defining Master Files at Project Level](#) on page 242 section or [Working with Modular XML Files in the Master Files Context](#) on page 308.

To watch our video demonstration about how to use a validation scenario in Oxygen XML Developer, go to http://oxygenxml.com/demo/Validation_Scenario.html.

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 are used and take precedence over the default scenarios defined for the particular framework. For more information on master files, see the [Defining Master Files at Project Level](#) on page 242 section or [Working with Modular XML Files in the Master Files Context](#) on page 308.

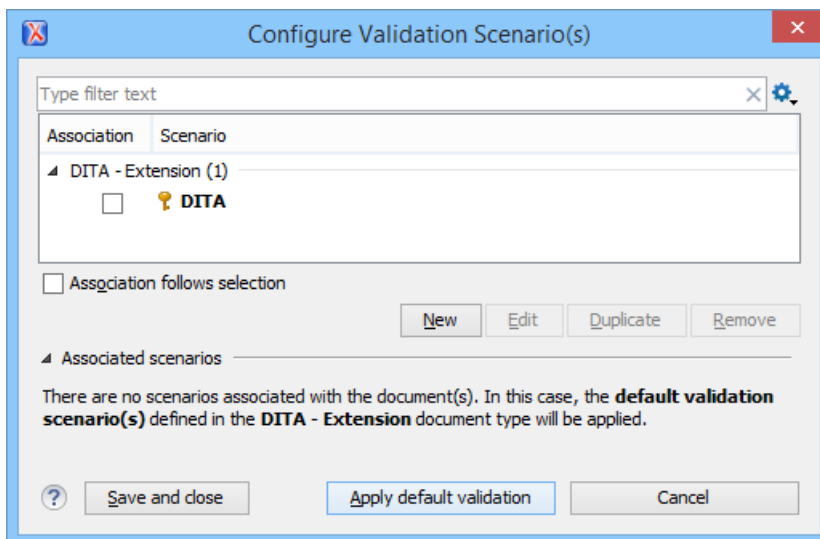


Figure 105: Configure Validation Scenario Dialog Box

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 Developer 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 Developer 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 Developer adds **imported** to the name of the imported scenario.
-  Export selected scenarios

Use this option to export selected scenarios individually. Oxygen XML Developer 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.

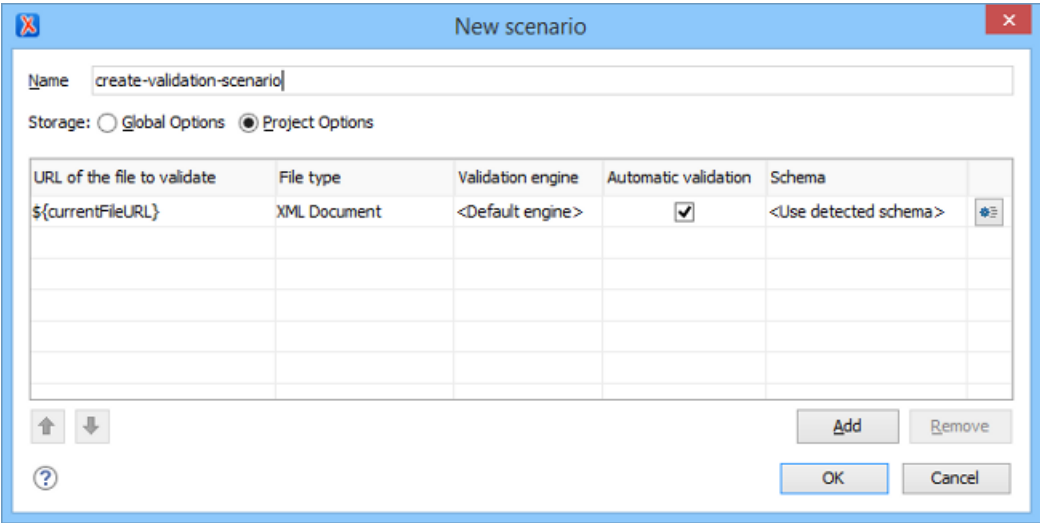


Figure 106: Create New Validation Scenario

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 **Global Options** or **Project 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*.

<code>\${Desktop}</code>	- My Desktop
<code>\${start-dir}</code>	- Start directory of custom validator
<code>\${standard-params}</code>	- List of standard params for command line
<code>\${cfn}</code>	- The current file name without extension
<code>\${currentFileURL}</code>	- The path of the currently edited file (URL)
<code>\${cfdu}</code>	- The path of current file directory (URL)
<code>\${frameworks}</code>	- Oxygen frameworks directory (URL)
<code>\${pdu}</code>	- Project directory (URL)
<code>\${oxygenHome}</code>	- Oxygen installation directory (URL)
<code>\${home}</code>	- The path to user home directory (URL)
<code>\${pn}</code>	- Project name
<code>\${env(VAR_NAME)}</code>	- Value of environment variable VAR_NAME
<code>\${system(var.name)}</code>	- Value of system variable var.name

Figure 107: Insert an Editor Variable

File type	The type of the document that is validated in the current validation unit. Oxygen XML Developer 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 Developer 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. The Table Layout Validation engine looks for table layout problems.
Automatic validation	If this option is checked, 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.
 Settings	Opens the Specify Schema dialog box that allows you to set a schema for validating XML documents, or a list of extensions for validating XSL or XQuery documents. You can also set a default phase for validation with a Schematron schema.
 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.
- 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*.
 - 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 are used and take precedence over the default scenarios defined for the particular framework. For more information on master files, see the [Defining Master Files at Project Level](#) on page 242 section or [Working with Modular XML Files in the Master Files Context](#) on page 308.

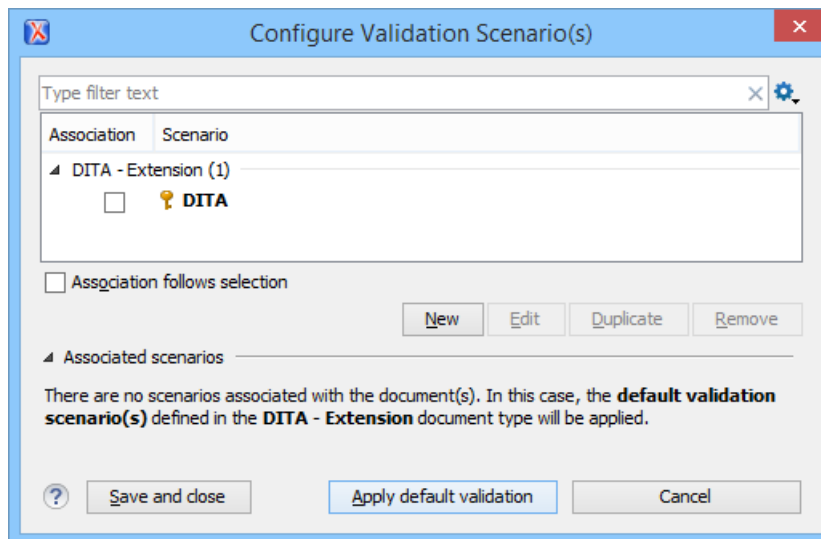


Figure 108: Configure Validation Scenario Dialog Box

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 Developer 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 Developer 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 Developer adds imported to the name of the imported scenario.

**Export selected scenarios**

Use this option to export selected scenarios individually. Oxygen XML Developer 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 Developer needs to create a 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.

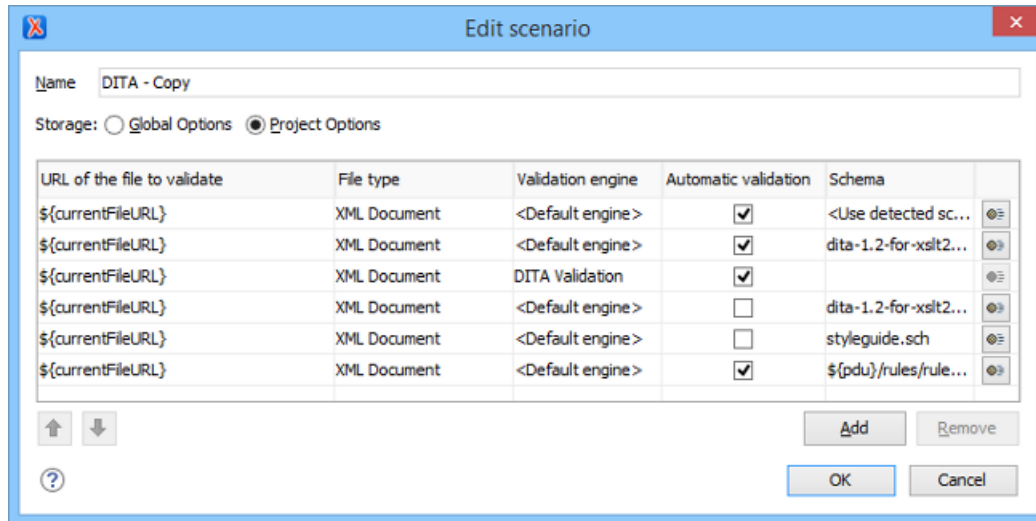


Figure 109: Edit Validation Scenario

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 **Global Options** or **Project 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*.

<code>\${Desktop}</code>	- My Desktop
<code>\${start-dir}</code>	- Start directory of custom validator
<code>\${standard-params}</code>	- List of standard params for command line
<code>\${cfn}</code>	- The current file name without extension
<code>\${currentFileURL}</code>	- The path of the currently edited file (URL)
<code>\${cfdu}</code>	- The path of current file directory (URL)
<code>\${frameworks}</code>	- Oxygen frameworks directory (URL)
<code>\${pdu}</code>	- Project directory (URL)
<code>\${oxygenHome}</code>	- Oxygen installation directory (URL)
<code>\${home}</code>	- The path to user home directory (URL)
<code>\${pn}</code>	- Project name
<code>\${env(VAR_NAME)}</code>	- Value of environment variable VAR_NAME
<code>\${system(var.name)}</code>	- Value of system variable var.name

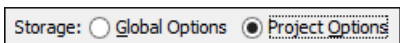
Figure 110: Insert an Editor Variable

File type	The type of the document that is validated in the current validation unit. Oxygen XML Developer 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 Developer 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. The Table Layout Validation engine looks for table layout problems.
Automatic validation	If this option is checked, 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.
 Settings	Opens the Specify Schema dialog box that allows you to set a schema for validating XML documents, or a list of extensions for validating XSL or XQuery documents. You can also set a default phase for validation with a Schematron schema.
 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.

- 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*.
- 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 **Global Options** or **Project Options**.



Selecting **Global Options** stores the scenario in the global options that are stored in the user home directory.

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.

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 and Storing Options](#) on page 148

References to XML Schema Specification

If validation is done against XML Schema, Oxygen XML Developer 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.

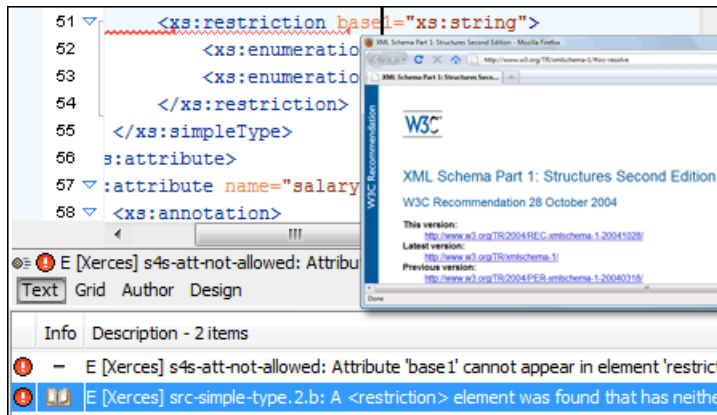


Figure 111: Link to Specification for XML Schema Errors

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"/>
```

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.

Validate an XML Document Against Schematron

To validate an XML document against a Schematron schema, select the  **Validate** action from the  **Validation** toolbar drop-down menu, or the **Document > Validate** menu, or from the **Validate** menu when invoking the contextual menu in the **Project** view.

If you want to add a persistence association between your Schematron rules and the current edited XML document, use the  **Associate Schema** action from the **Document > Schema** menu or the **Document** toolbar. A custom processing instruction is added into the document and the validation process will take into account the Schematron rules:

PI Added by the Associate Schema Action

```

<?xml-model href="percent.sch" type="application/xml"
schematypens="http://purl.oclc.org/dsdl/schematron"?>

```

The possible errors that might occur during the validation process are presented in the **Errors** panel at the bottom area of the Oxygen XML Developer window. Each error is flagged with a severity level that can be one of *warning*, *error*, *fatal* or *info*.

To set a severity level, Oxygen XML Developer looks for the following information:

- The `role` attribute, which can have one of the following values:
 - `warn` or `warning`, to set the severity level to *warning*
 - `error`, to set the severity level to *error*
 - `fatal`, to set the severity level to *fatal*
 - `info` or `information`, to set the severity level to *info*
- The start of the message, after trimming leading white-spaces. Oxygen XML Developer looks to match the following exact string of characters (case sensitive):
 - `Warning:`, to set the severity level to *warning*
 - `Error:`, to set the severity level to *error*
 - `Fatal:`, to set the severity level to *fatal*
 - `Info:`, to set the severity level to *info*



Note: Displayed message does not contain the matched prefix.

- If none of the previous rules match, Oxygen XML Developer sets the severity level to *error*.

Related concepts

[Validation Scenario](#) on page 286

Finding and Replacing Text in the Current File

This section walks you through the find and replace features available in Oxygen XML Developer.

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)**.

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.

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 checked, 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 enabled, 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 enabled, 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 disabled 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 enable 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, disable **Enable XML search options**.

- **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.

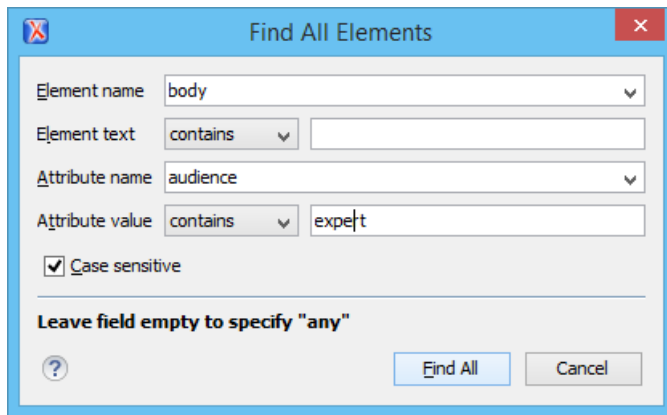


Figure 112: Find All Elements Dialog Box

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 checked, operations are case-sensitive

When you press **Find All**, Oxygen XML Developer 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 Developer 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 or **ESC** key.

All matches are highlighted in the current editor.

The toolbar offers the following controls:

- A search input box where you 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** and **Previous** buttons. They allow you to advance to the next or previous match.
- **All** button. Highlights in the current document all matches of the search string.
- **Incremental** checkbox. The search operation is started every time you type or delete a character in the search input box.
- **Case sensitive** checkbox. When selected, the search operation follows the exact letter case of the search text.
- Shortcuts to the  [Find/Replace](#) and  [Find/Replace in Files](#) dialog boxes.

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 to quickly repeat the last find action performed in the [Find / Replace dialog box](#), taking into account the same find options.

Regular Expressions Syntax

Oxygen XML Developer uses the [Java regular expression syntax](#). It is **similar** to that used in Perl 5, with several exceptions. Thus, Oxygen XML Developer 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 Developer/Java in the text searches from the **Find/Replace** dialog boxes. 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 boxes:

- `/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

<http://docs.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html#jcc>

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.

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(ls+)(.*)>` 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 Developer 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`, `.nvd1`, `.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.

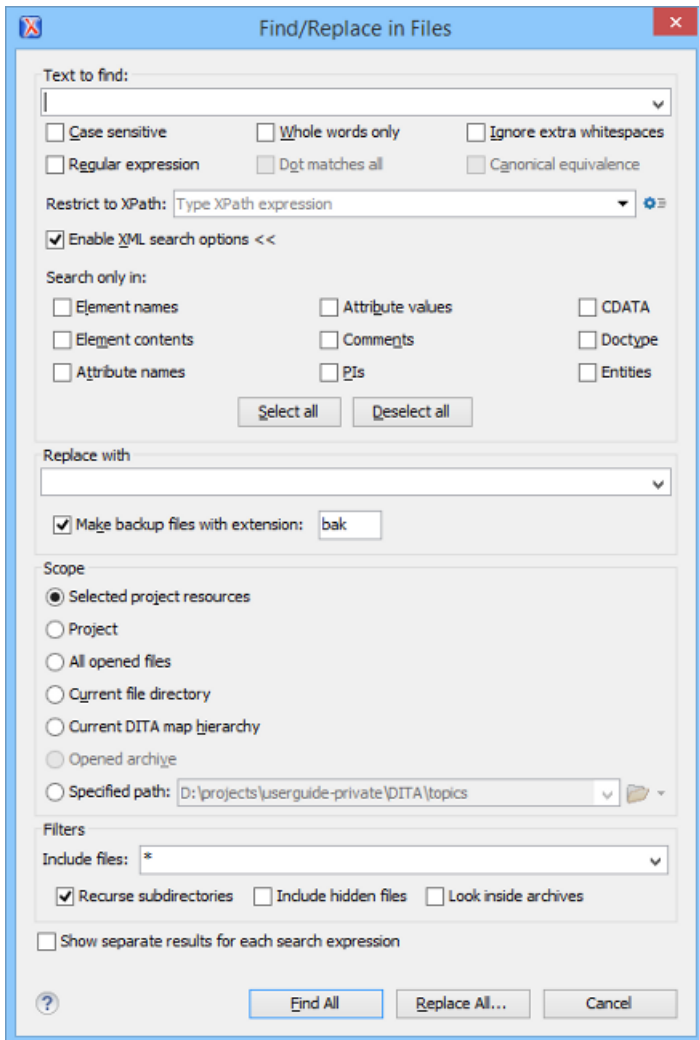


Figure 113: Find / Replace in Files Dialog Box

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 (\ONNNN) 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 checked, 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 enabled, 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 enabled, 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 enabled, 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 disabled 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 enable 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, disable **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 checked.
- **Make backup files with extension** - In the replace process Oxygen XML Developer 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 [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 [Archive Browser](#) view.
- **All opened files** - Searches in all files opened in Oxygen XML Developer. 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 [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 disabled. This option is not displayed when this dialog box is opened from the [Archive Browser](#) 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 enabled, 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 enabled, the search is also performed in the hidden files.
- **Include archives** - When enabled, the search is also done in all individual file entries from all supported ZIP-type archives.
- **Show separate results for each search expression** - When enabled, 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 Developer 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 Developer uses the *Java regular expression syntax*. It is **similar** to that used in Perl 5, with several exceptions. Thus, Oxygen XML Developer 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 Developer/Java in the text searches from the **Find/Replace** dialog boxes. 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 boxes:

- `/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

<http://docs.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html#jcc>

Comparison between the Java and Perl 5 regular expression syntax

Search and Refactor Actions for IDs and IDREFS

Oxygen XML Developer 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 Developer 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.

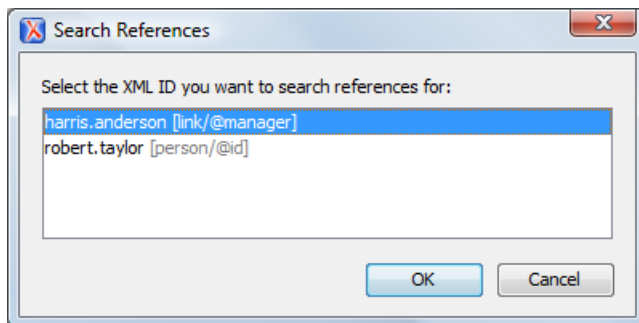


Figure 114: Selecting an ID in the Author Mode

Related information

[Working with Modular XML Files in the Master Files Context](#) on page 308

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](#).

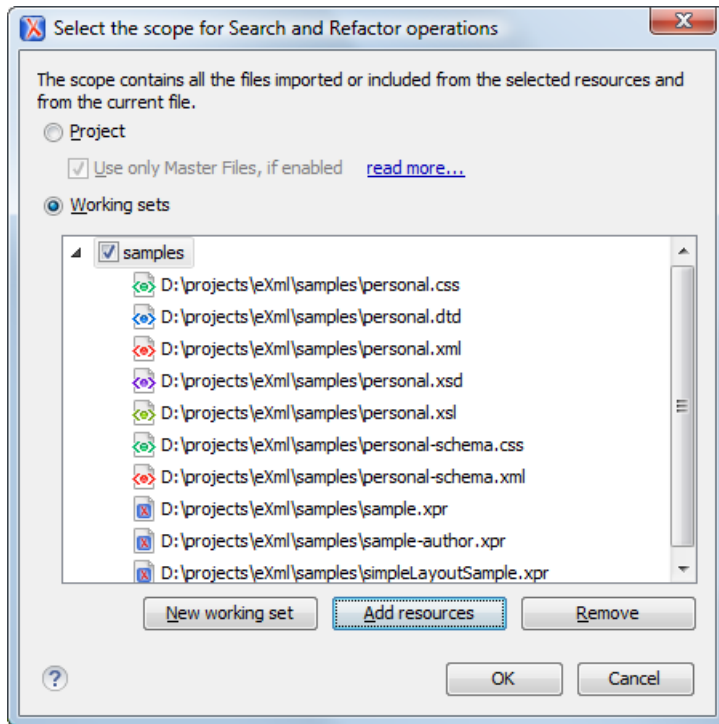


Figure 115: Change Scope Dialog Box

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.

Associate a Schema to a Document

This section explains the methods of associating a schema to a document for validation and content completion purposes.

Related information

[Working with Modular XML Files in the Master Files Context](#) on page 308

Setting a Schema for Content Completion

This section explains the available methods of setting a schema for content completion in an XML document edited in Oxygen XML Developer.

Supported Schema Types for XML Documents

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)

Setting a Default Schema

When trying to detect a schema, Oxygen XML Developer searches in multiple locations, in the exact following order:

- The [validation scenario](#) associated with the document.
- The validation scenario associated with the document type (if defined).

- The document schema declaration.
- The document type schema definition. Each document type available in [Document Type Association preferences page](#) contains a set of rules for associating a schema with the current document.



Note: The locations are sorted by priority, from high to low.

The schema has one of the following types: XML Schema, XML Schema with embedded Schematron rules, Relax NG (XML syntax or compact syntax), Relax NG (XML syntax) with embedded Schematron rules, Schematron, DTD, NVDL.

The rules are applied in the order they appear in the table and take into account the local name of the root element, the default namespace and the file name of the document.



Important:

The editor is creating the content completion lists by analysing the specified schema and the current context (the position in the editor). If you change the schema, then the list of tags to be inserted is updated.

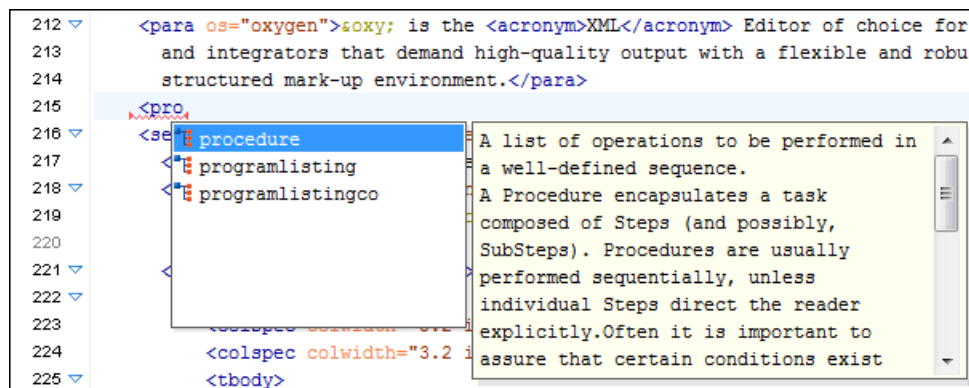


Figure 116: Content Completion Driven by DocBook DTD

Making the Schema Association Explicit in the XML Instance Document

The schema used by the [Content Completion Assistant](#) and [document validation](#) engine can be associated with the document using the **Associate Schema** action. For most of the schema types, it uses [the xml-mode l processing instruction](#), the exceptions being:

- **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 document type level.

Select the **Associate schema** action from the toolbar or **Document > Schema** menu to select the schema that will be associated with the XML document. The **Associate Schema** dialog box is displayed:

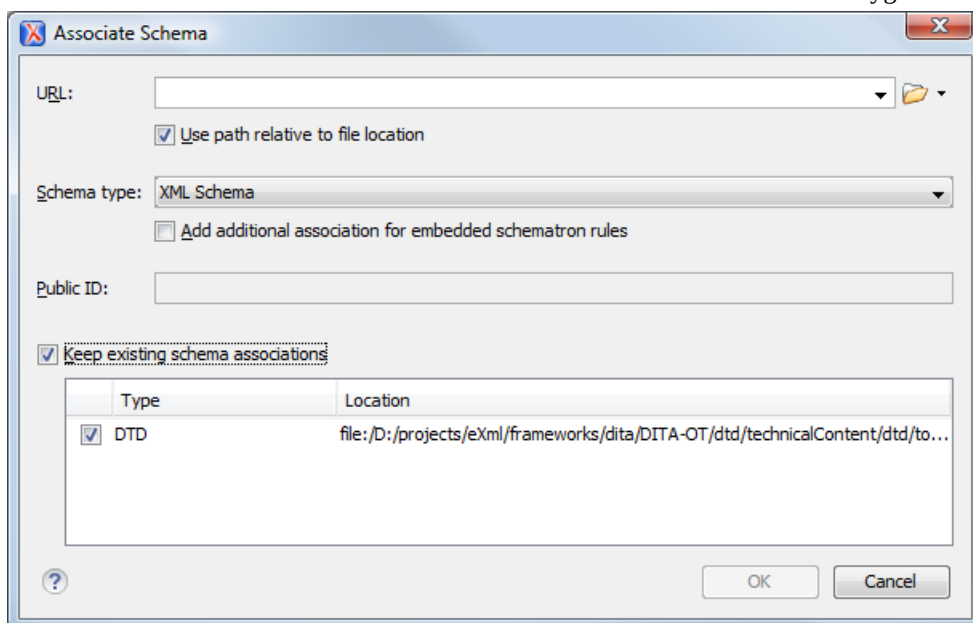


Figure 117: Associate Schema Dialog Box

The available options are:

- **URL** - Contains a predefined set of schemas that are used more often and 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](#).
- **Schema type** - Selected automatically from the list of possible types in the **Schema type** combo box (XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, NVDL) based on the extension of the schema file that was entered in the **URL** field.
- **Public ID** - Specify a public ID if you have selected a DTD.
- **Add additional association for embedded schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules, enable this option.
- **Use path relative to file location** - Enable 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.
- **Keep existing schema associations** - Enable this option to keep the associations of the currently edited document with a Schema when you associate a new one.

The association with an XML Schema is added as an attribute of the root element. The **Associate schema** action adds 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.

The association with a DTD is added as a **DOCTYPE** declaration. The association with a Relax NG, Schematron or NVDL schema is added as [xml-model processing instruction](#).

Associating a Schema With the Namespace of the Root Element

The namespace of the root element of an XML document can be associated with an XML Schema using an [XML catalog](#). If there is no `xsi:schemaLocation` attribute on the root element and the [XML](#) document is not matched with a [document type](#), the namespace of the root element is searched in [the XML catalogs set in Preferences](#).

If the XML catalog contains a `uri` or `rewriteUri` or `delegateUri` element, its schema will be used by the application to drive the [content completion](#) and document [validation](#).

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 prologue. 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 Developer 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`.
- **NVDSLSchema** - 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 Developer.

Learning Document Structure

When working with documents that do not specify a schema, or for which the schema is not known or does not exist, Oxygen XML Developer 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 DTD's for documents containing personal or custom element types.

When you open a document that is not associated with a schema, Oxygen XML Developer automatically learns the document structure and uses it for *content completion*. To disable this feature you have to uncheck the checkbox *Learn on open document in the user preferences*.

Create a DTD from Learned Document Structure

When there is no schema associated with an XML document, Oxygen XML Developer can learn the document structure by parsing the document internally. This feature is enabled with [the option *Learn on open document*](#) that is available in the user preferences.

To create a DTD from the learned structure:

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's 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.

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 Developer 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 XML files.

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 main file using a validation scenario, add validation units that point to the main modules. Oxygen XML Developer 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 Developer 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 determine the [scope of the search and refactoring actions](#) for ID/IDREFS values and for updating references when renaming/moving a resource. Oxygen XML Developer 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 http://oxygenxml.com/demo/Working_With_XML_Modules.html.

Related information

[Defining Master Files at Project Level](#) on page 242

[XML Resource Hierarchy/Dependencies View](#) on page 308

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**.

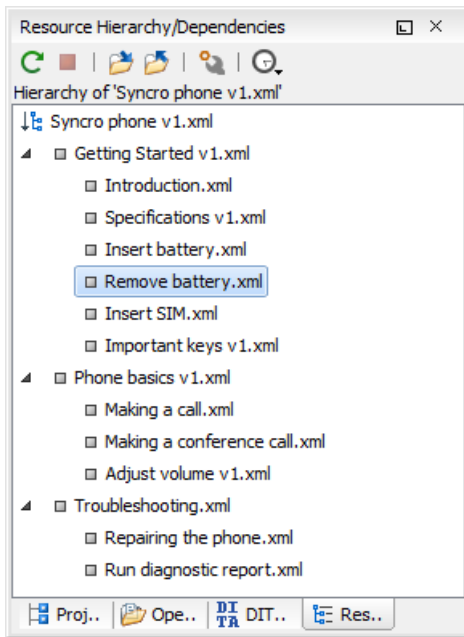


Figure 118: Resource Hierarchy/Dependencies View - Hierarchy for Syncro phone v1.xml

The build process for the dependencies view is started with the **Resource Dependencies** action available on the contextual menu.

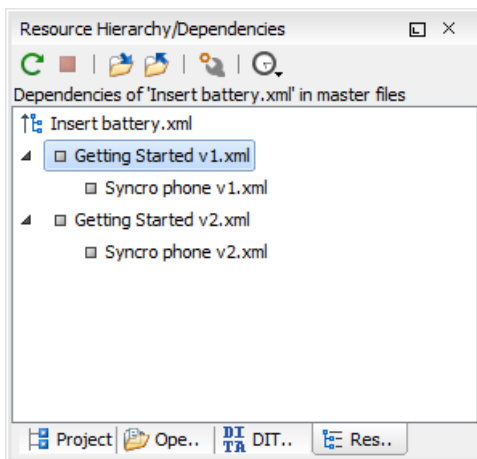


Figure 119: Resource Hierarchy/Dependencies View - Dependencies for Insert battery.xml

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 308

[Search and Refactor Operations Scope](#) on page 303

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** - Enable 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)** - Enable 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 enabled, 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

An *XML Catalog* maps a system ID or a URI reference pointing to a resource (stored either remotely or locally) to a local copy of the same resource. If 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 Internet connection is slow.

Oxygen XML Developer 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 Developer uses different catalog mappings.

Table 5: Catalog Mappings

Document	Referenced Resource	Mappings
XML	DTD	<i>system</i> or <i>public</i> The <i>Prefer option</i> 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):
	Relax NG	1. resolve the schema using <i>URI</i> catalog mappings.
	Schematron	2. resolve the schema using <i>system</i> catalog mappings.
	NVDL	This happens only if the <i>Resolve schema locations also through system mappings option</i> is enabled (it is by default). 3. resolve the root <i>namespace</i> using <i>URI</i> catalog mappings.
XSL	XSL/ANY	<i>URI</i>
CSS	CSS	<i>URI</i>
XML Schema	XML Schema	The following strategy is used (if one step fails to provide a resource, the next is applied):
	Relax NG	1. resolve schema reference using <i>URI</i> catalog mappings.
	Relax NG	2. resolve schema reference using <i>system</i> catalog mappings.
		This happens only if the <i>Resolve schema locations also through system mappings option</i> is enabled (it is by default). 3. resolve schema <i>namespace</i> using <i>uri</i> catalog mappings. This happens only if the <i>Process namespaces through URI mappings for XML Schema option</i> is enabled (it is not by default).

Creating an XML Catalog with a Template

An XML Catalog file can be created quickly in Oxygen XML Developer starting from the two XML Catalog document templates called *OASIS XML Catalog 1.0* and *OASIS XML Catalog 1.1* and 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"/>
  <!-- The "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.xml"
    uri="frameworks/docbook/xml/fo/docbook.xml"/>

</catalog>

```

How Oxygen XML Developer Determines which Catalog to Use

An XML catalog is used to resolve references for document validation and transformations and it maps such references to the built-in local copies of the schemas of the Oxygen XML Developer frameworks (DocBook, DITA, TEI, XHTML, SVG, etc.)

Oxygen XML Developer includes a built-in catalog set as default, but you can also create your own and Oxygen XML Developer 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 at document type level, in the [Catalog tab](#) from the [Document Type configuration dialog box](#).
- Default built-in catalogs.

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 99

Resolve Schemas Through XML Catalogs

Oxygen XML Developer resolves the location of a schema in the following order:

- First, it attempts to resolve the schema location as a URI (`uri`, `uriSuffix`, `rewriteURI` mappings from the XML catalog). If this succeeds, the process ends here.
- If the [Resolve schema locations also through system mappings option](#) is selected, it attempts to resolve the schema location as a systemID (`system`, `systemSuffix`, `rewriteSuffix`, `rewriteSystem` from the XML catalog). If this succeeds, the process ends here.
- If the [Process namespace through URI mappings for XML Schema option](#) is selected, it attempts to resolve the location by processing the schema namespace as a URI (`uri`, `uriSuffix`, `rewriteURI` from the XML catalog). The namespace is taken into account only when the schema specified in the `schemaLocation` attribute was not resolved successfully. If this succeeds, the process ends here.
- If none of these succeeds, the actual schema location is used.

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 (the 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 main 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 frameworks such as manuals and Web pages. Employing XInclude enables authors and content managers 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.

Include a chapter in an article using XInclude

Create a chapter file and an article file in the `samples` folder of the Oxygen XML Developer install folder.

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 the following is of note:

- 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.

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. 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>
  <xi:include href="a.xml" xpointer="a1"
             xmlns:xi="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>
```

The XInclude support in Oxygen XML Developer is enabled by default. To [configure it, open the Preferences dialog box \(Options > Preferences\)](#) and go to **XML > XML Parser > Enable XInclude processing**. When enabled, Oxygen XML Developer will be able to validate and transform documents comprised of parts added using XInclude.

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.

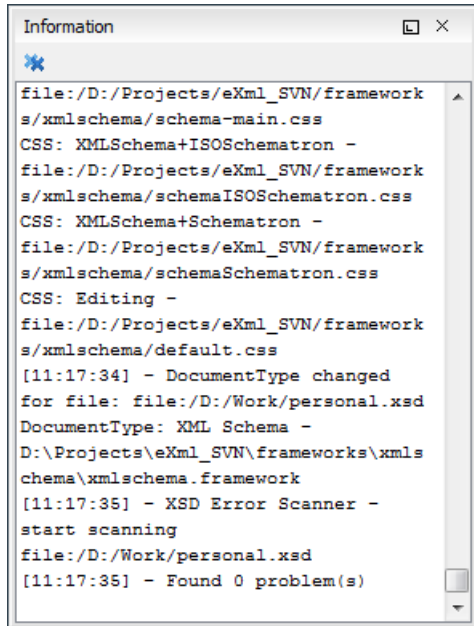


Figure 120: Information view messages

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** panel 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** panel, 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>
  </end>
</Location>
</Incident>
</Report>

```

Editor Highlights

An *editor highlight* is a text fragment emphasized by a colored background.

Highlights are generated when the following actions generate results:

- Find/Replace in Files
- Open/Find Resource
- Find All
- Find All Elements
- XPath in Files
- Search References
- Search Declarations

By default, Oxygen XML Developer 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 from the results panel at the bottom of the editor.



Note: Use the  **Highlight all results in editor** button to either display all the highlights or hide them.

Printing a File

Printing is supported in **Text** 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.

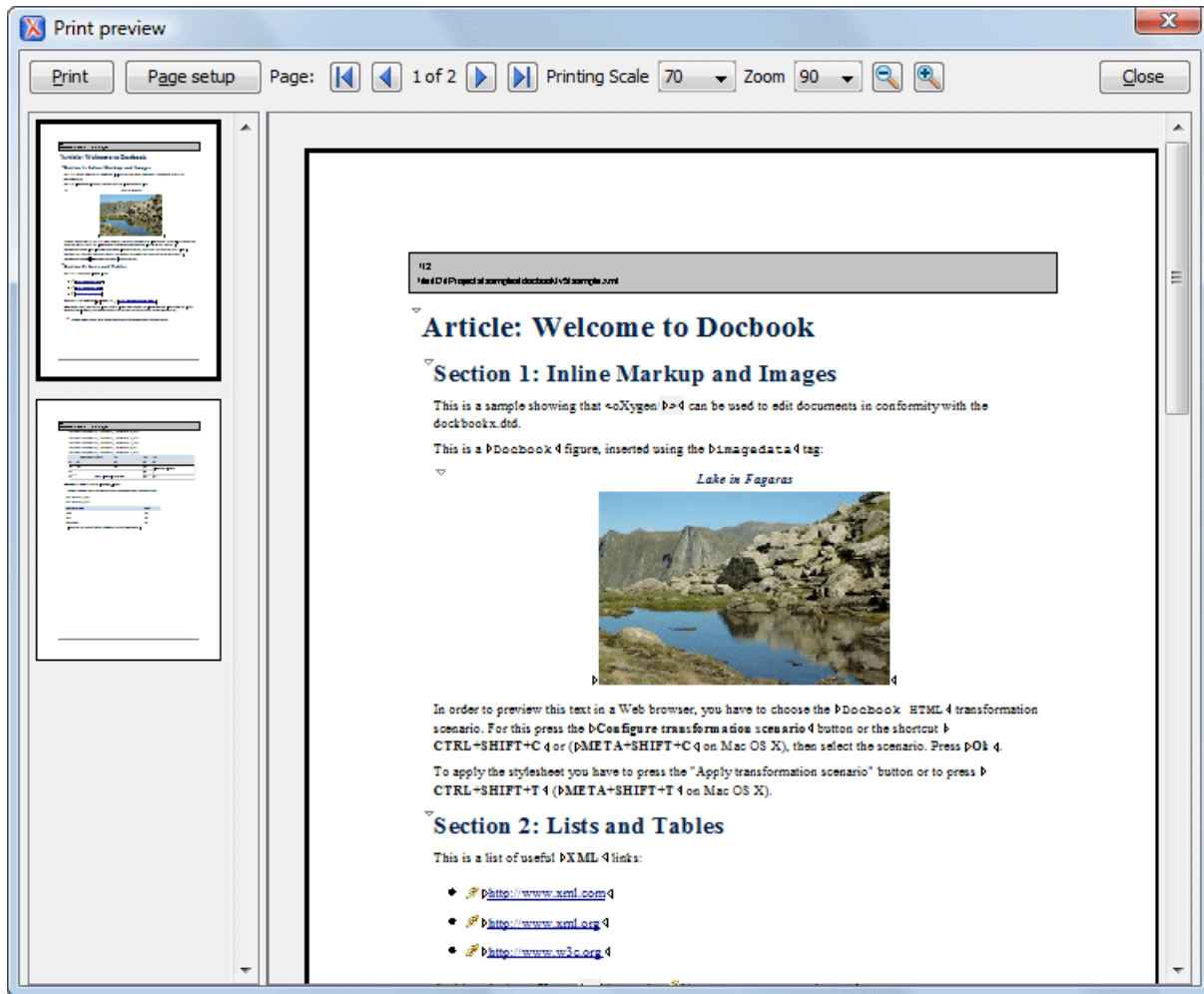


Figure 121: Print Preview Dialog Box

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 **Text** mode and line numbers are displayed (the *Show line numbers option* is enabled), 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 enabled), 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.

XML Quick Fixes

The Oxygen XML Developer 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

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 and the available quick fixes include a link that can be used to perform the fix.



Figure 122: Quick Fix Presented in a Tooltip in Text 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 Developer displays the list of available fixes.

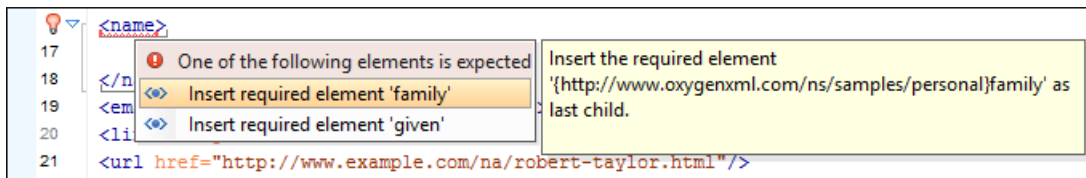


Figure 123: Quick Fix Menu Invoked by Clicking on the 💡 Icon

- 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 Assistant options](#) are configured.

Quick Fixes for XSD and Relax NG Errors

Oxygen XML Developer offers quick fixes for common errors that appear in XML documents that are validated against 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 .

Oxygen XML Developer 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 319

Schematron Quick Fixes (SQF)

Oxygen XML Developer provides support for Schematron Quick Fixes (SQF). They help you resolve errors 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 error 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:

<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.

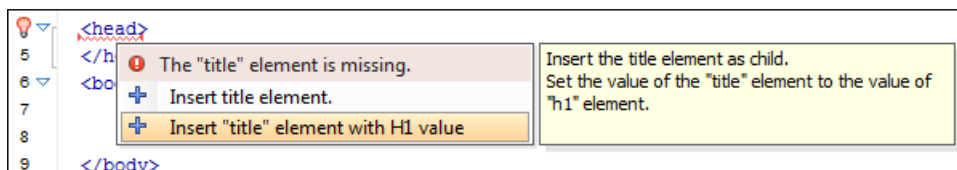


Figure 124: Example of a Schematron Quick Fix

Related information

[Editing Schematron Quick Fixes](#) on page 498

[Schematron Quick Fix Specifications](#)

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 Developer 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.



Note: The predefined refactoring operations are also available from the **Refactoring** submenu in the contextual menu of **Text** mode. This is useful because by selecting the operations from the contextual menu, Oxygen XML Developer 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 are used also appear in the **Refactoring** submenu of the contextual menu in the **Project** view.

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.

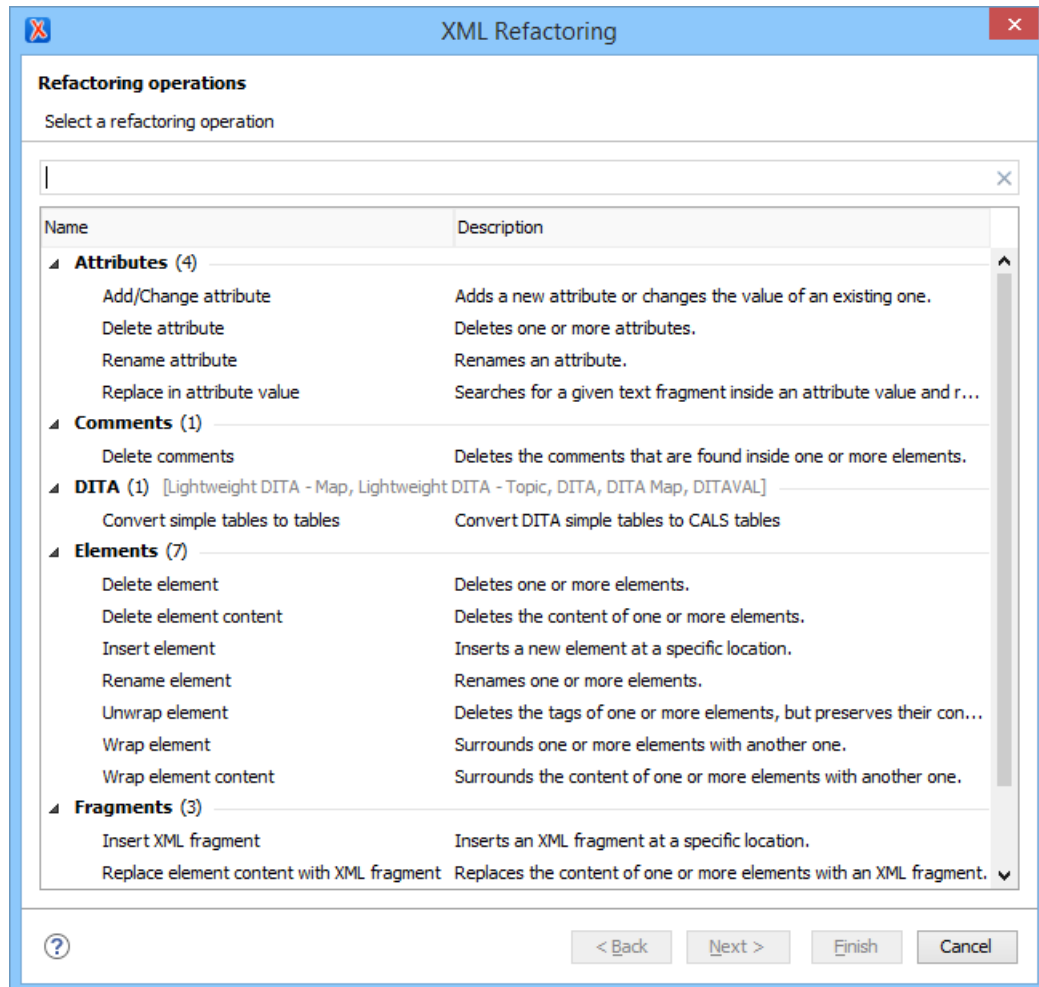


Figure 125: XML Refactoring Wizard

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.

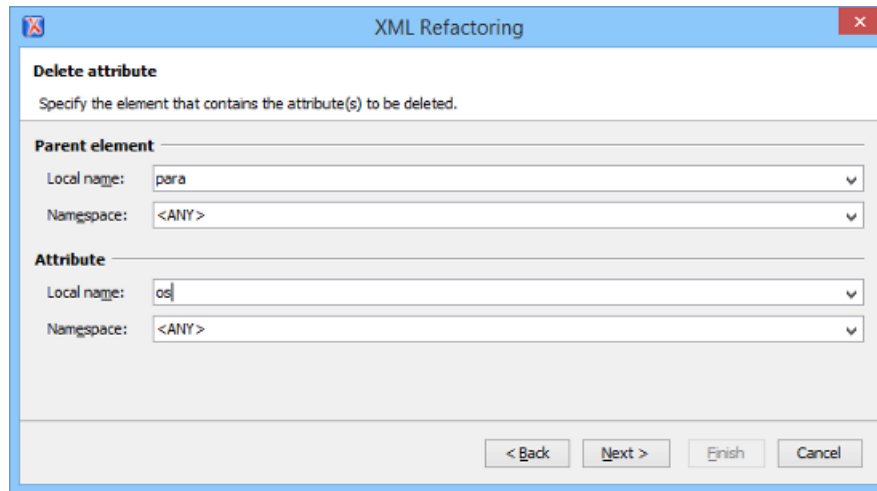


Figure 126: XML Refactoring 2nd Wizard Page (Delete Attribute Operation)

Scope and Filters The last wizard page allows you to select the set of files that represent the input of the operation. You can select from predefined resource sets (such as the current file, your whole project, the current DITA map hierarchy, etc.) or you can define your own set of resources by creating a working set.

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 enabled, only resources with a known XML file type will be affected by the operation.
- **Look inside archives** - When enabled, the resources inside archives will also be affected.

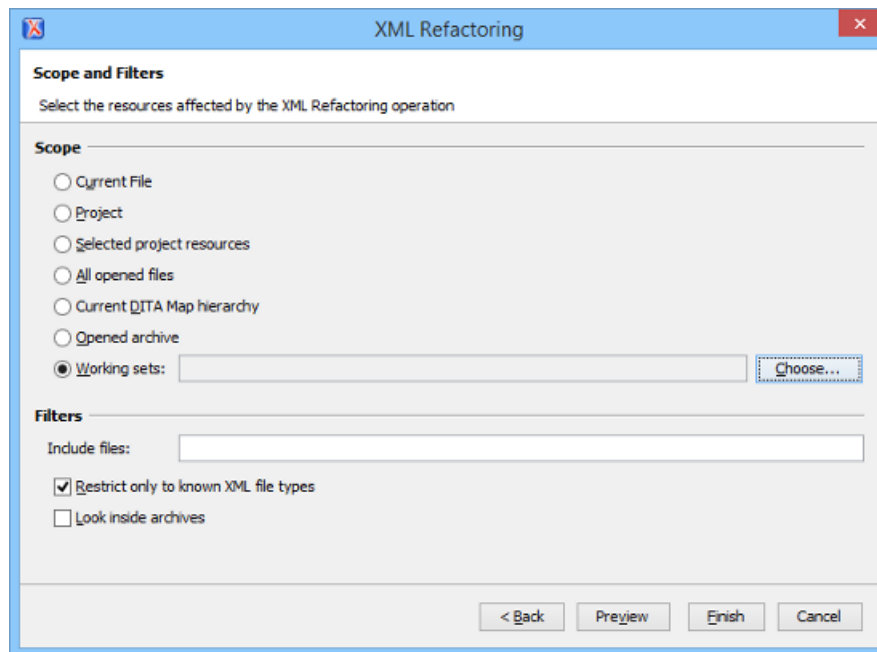


Figure 127: XML Refactoring - Scope and Filters Wizard Page

If an operation takes longer than expected you can use the  **Stop** button in the progress bar to cancel the operation.



Note: It is recommended that you use the **Preview** button to review all the changes that will be made by the refactoring operation before applying the changes.



Warning: After clicking the **Finish** button, the operation will be processed and Oxygen XML Developer provides no automatic means for reverting the operations. Any **Undo** action will only revert changes on the current document.

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 **Text** mode. The operations are also grouped by category in this submenu. When selecting the operations from the contextual menu, Oxygen XML Developer 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 attribute 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.

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 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.

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 Developer 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 Developer 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).

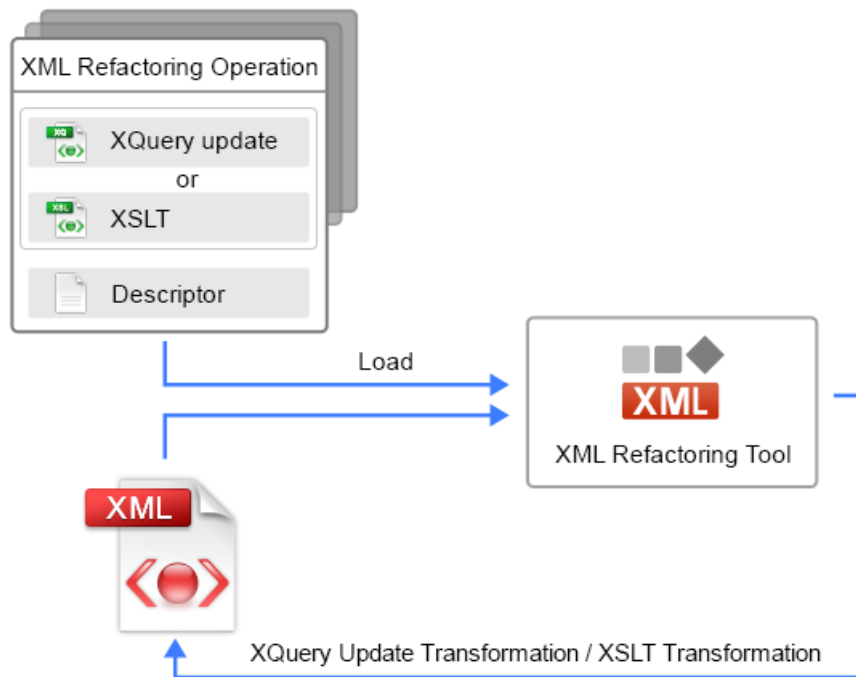


Figure 128: Diagram of an XML Refactoring Operation

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 Developer 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.
 - 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://oxygenxml.com/app/xml_refactory/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 performed on *comments* or *processing instructions* that are inside the root element.

Creating a Custom Refactoring Operation

To create a custom refactoring operation, follow these steps:

1. [Create an XQuery Update script or XSLT file.](#)
2. [Create an XML Refactoring Operation Descriptor file.](#)
3. Store both files in [one of the locations that Oxygen XML Developer](#) 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 333

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 examples](#) 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 Developer. Thus, if you want to create or edit an XQuery Update script you have to enable this facility 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 enabled 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 a custom operation descriptor file](#).

Related information

[Example of an XML Refactoring Operation](#) on page 330

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 Developer 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 Developer 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 Developer 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 Developer 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>The local name of the attribute's parent element.</description>
  </localName>
  <namespace label="Namespace" name="element_namespace" allowsAny="true">
    <description>The local name of the attribute's 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>The 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 XML Refactoring Operation](#) on page 330

Example of an XML Refactoring 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.

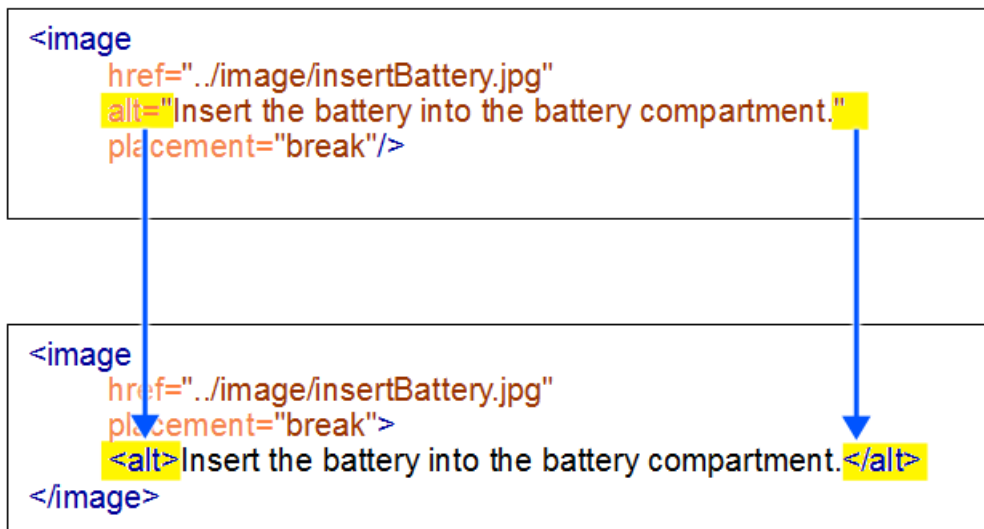


Figure 129: Example: Custom XML Refactoring Operation

A new custom XML refactoring operation requires:

- An [XQuery Update script](#) or [XSLT stylesheet](#).
- An [XML Refactoring operation descriptor file](#) that contains the path to the XQuery Update script or XSLT stylesheet.

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 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 for Creating a Custom Operation to Convert an Attribute to an Element

After you have developed the XQuery script or 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>The local name of the attribute's parent element.</description>
        </localName>
        <namespace label="Namespace" name="element_namespace" allowsAny="true">
          <description>The local name of the attribute's parent element</description>
        </namespace>
      </elementParameter>

```



```

</section>
<section label="Attribute">
  <attributeParameter dependsOn="elemID">
    <localName label="Name" name="attribute_localName">
      <description>The name of the attribute to be converted.</description>
    </localName>
    <namespace label="Namespace" name="attribute_namespace" allowsAny="true">
      <description>The 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 Developer 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 130: Example: XML Refactoring Wizard for a Custom Operation

Storing and Sharing Refactoring Operations

Oxygen XML Developer 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.
- Any folder. In this case, you need to *open the Preferences dialog box (Options > Preferences)*, go to **XML > XML Refactoring**, and specify the same folder in the *Load additional refactoring operations from text box*.



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 Developer installation directory (`[OXYGEN_INSTALL_DIR]/refactoring/`).

Sharing Custom Refactoring Operations

The purpose of Oxygen XML Developer 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 frameworks or projects.

After storing custom operations, you can share them with other users by sharing the resources.

Localizing XML Refactoring Operations

Oxygen XML Developer 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 Developer 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 Developer installation directory (`[OXYGEN_INSTALL_DIR]/refactoring/i18n`).

Example of a Refactoring Operation Descriptor File with *i18n* Support

```
<?xml version="1.0" encoding="UTF-8"?>
<refactoringOperationDescriptor xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://oxygenxml.com/app/xml_refactory
  http://oxygenxml.com/app/xml_refactory/operation_descriptor.xsd"
  xmlns="http://oxygenxml.com/app/xml_refactory" 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 Developer 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).
- **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 Developer makes XSLT files more readable.

To watch our video demonstration about basic XSLT editing and transformation scenarios in Oxygen XML Developer, go to http://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 Developer 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 can set a main XSLT stylesheet 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 modules. Oxygen XML Developer 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 main 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** 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 <http://oxygenxml.com/demo/MasterFilesSupport.html>.

Related information

[XSLT Resource Hierarchy/Dependencies View](#) on page 346

[XSLT Component Dependencies View](#) on page 349

Validating XSLT Stylesheets

Oxygen XML Developer 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 9.6.0.7 and [a JAXP transformer specified by the main Java class](#). For XSLT 2.0, the options are: Saxon 9.6.0.7

and *a JAXP transformer specified by the main Java class*. For XSLT 3.0, the options are Saxon 9.6.0.7 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 Developer, 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 Developer 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.

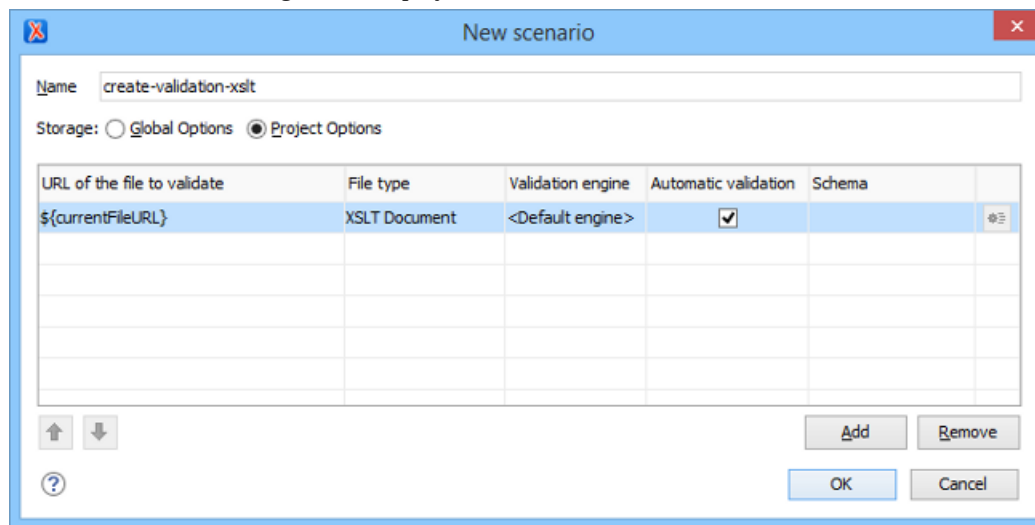


Figure 131: Add / Edit a Validation Unit

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 **Global Options** or **Project 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 checked, the validation operation defined by this row is also used by the automatic validation feature.

- 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.

- 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 Developer 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 Developer (Windows edition). It is associated to the XSL Editor type in [Preferences page](#).
- MSXML.NET** - included in Oxygen XML Developer (Windows edition). It is associated to the XSL Editor type in [Preferences page](#).

Content Completion in XSLT Stylesheets

The items in the list of proposals offered by the **Content Completion Assistant** are context-sensitive and includes proposed items that are valid at the current cursor position. You can enhance the list of proposals by specifying an additional schema. This schema is *defined by the user in the Content Completion / XSL preferences* page and can be: XML Schema, DTD, RELAX NG schema, or NVDL schema.

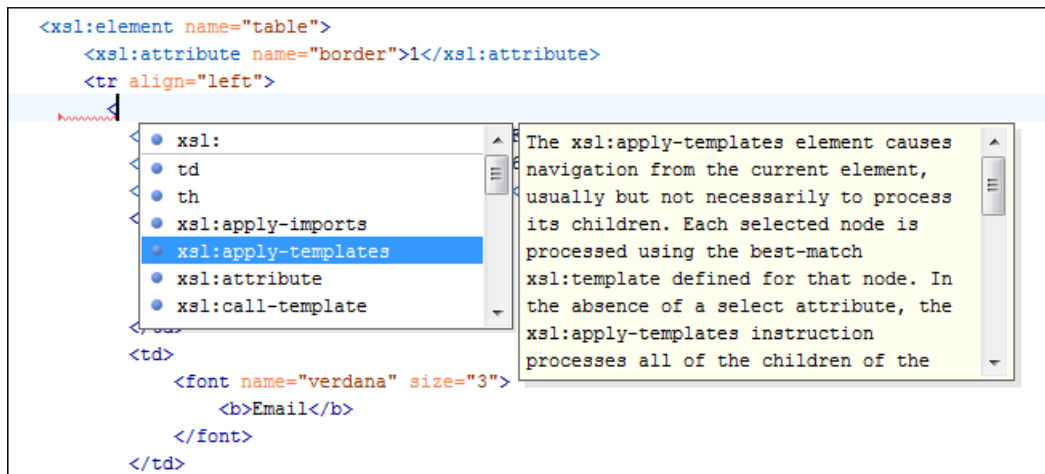


Figure 132: XSLT Content Completion Assistant

The feature is activated in **Text** mode in the following situations:

- After you press the $\leq$ 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 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 9.6.0.7 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 9.6.0.7 PE or Saxon 9.6.0.7 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 9.6.0.7 PE or Saxon 9.6.0.7 EE (for version 2.0 / 3.0).
- The validation engine specified in [Options](#) page is Saxon 6.5.5 (for version 1.0), Saxon 9.6.0.7 PE or Saxon 9.6.0.7 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.

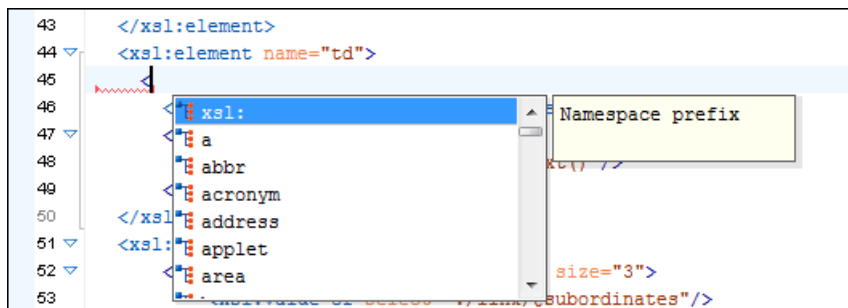


Figure 133: Namespace Prefixes in the Content Completion Assistant

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 Developer 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 items 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.

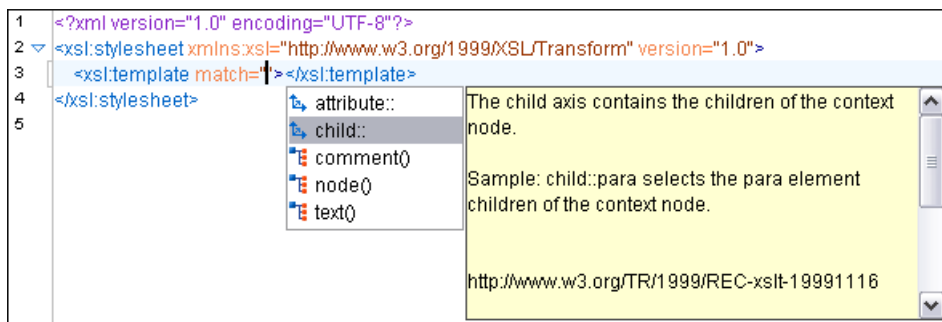


Figure 134: Content Completion in the match Attribute

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:

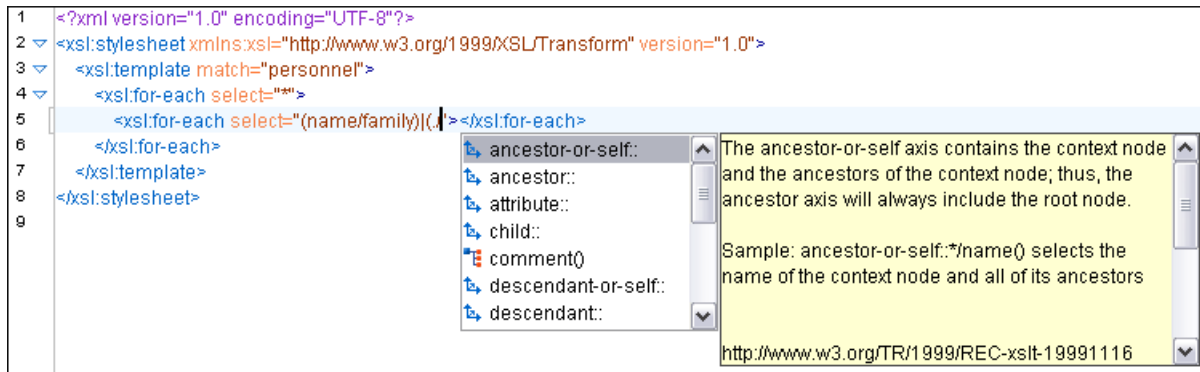


Figure 135: Content Completion in the select Attribute

Also XPath expressions typed in the `test` attribute of an `xsl:if` or `xsl:when` element benefit of the assistance of the content completion.

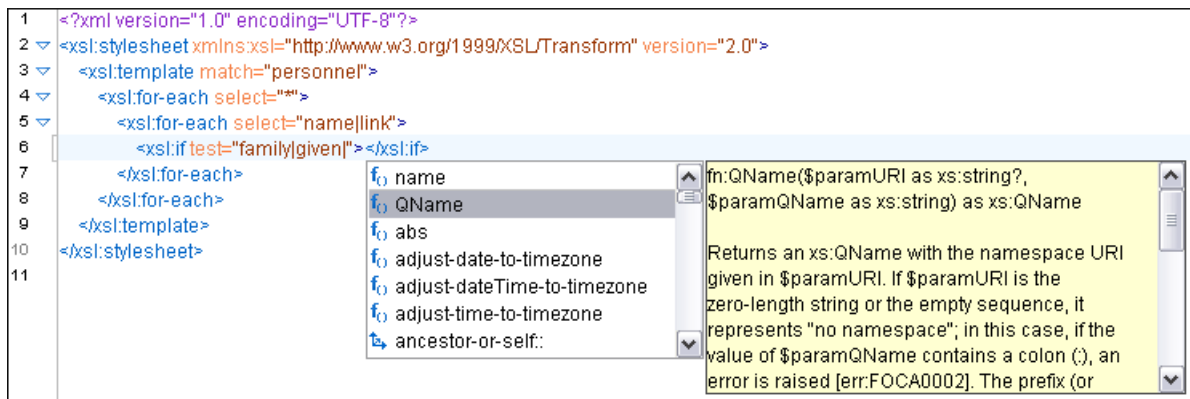


Figure 136: Content Completion in the test Attribute

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.

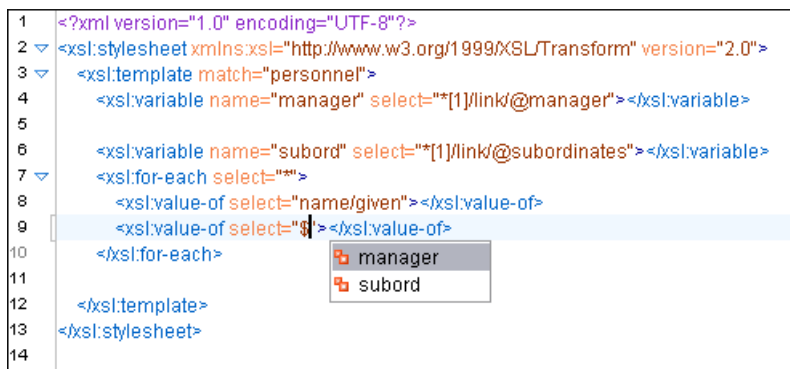


Figure 137: Content Completion in the test Attribute

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.

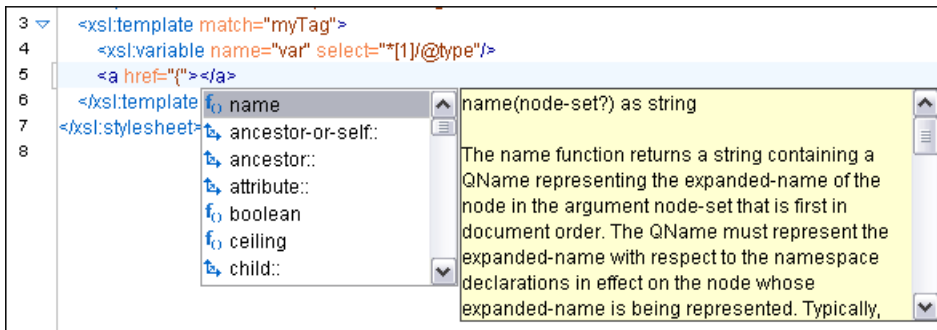


Figure 138: Content Completion in Attribute Value Templates

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.

Tooltip Helper for the XPath Functions Arguments

When editing the arguments of an XPath/XSLT function, Oxygen XML Developer 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.

We 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 Developer 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 takes also other arguments, having the same type, and returns a *xs:string*.

```
name: concat($arg1 as xdt:anyAtomicType?, $arg2 as xdt:anyAtomicType?, ...) as xs:string
match:
select: "concat(abs($v1), abs($v2))" </xsl:value-of>
</xsl:template>
:stylesheet>
```

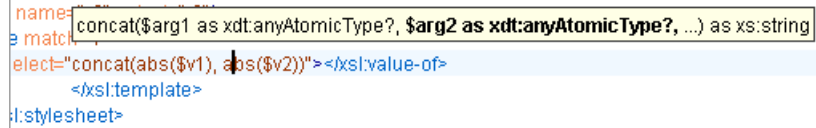
Figure 139: XPath Tooltip Helper - Identify the concat Function's First Argument

Moving the cursor on the first variable *\$v1*, the editor identifies the *abs* as context function and shows its signature:

```
name="v2" se: abs($arg as numeric?) as numeric?
match="/">
select: "concat(abs($v1), abs($v2))" </xsl:value-of>
</xsl:template>
:stylesheet>
```

Figure 140: 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.



```

name: concat($arg1 as xdt:anyAtomicType?, $arg2 as xdt:anyAtomicType?, ...) as xs:string
match:
select="concat(abs($v1), abs($v2))" => </xsl:value-of>
</xsl:template>
<!-- stylesheet -->

```

Figure 141: XPath Tooltip Helper - Identify the concat Function's Second Argument

The tooltip helper is available also in the XPath toolbar and the **XPath Builder** view.

Syntax Highlight

The XSL editor renders the CSS and JS scripts, and XPath expressions with dedicated coloring schemes. To customize the coloring schemes, [open the Preferences dialog box \(Options > Preferences\)](#) and go to **Editor > Syntax Highlight**.

Related information

[Customize Syntax Highlight colors](#) on page 87

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 from the **Window > Show View** menu.

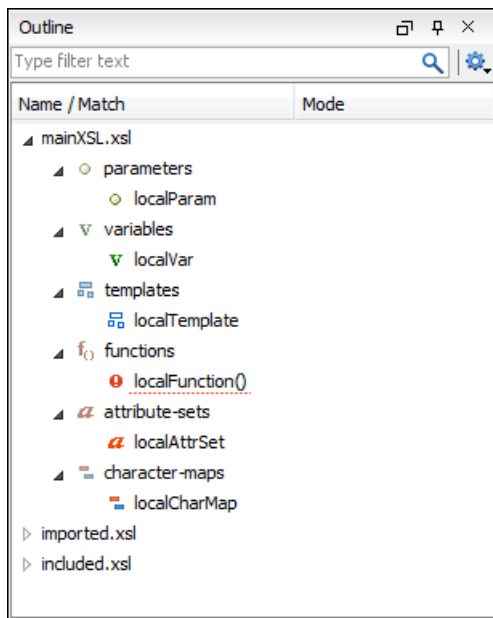


Figure 142: XSLT Outline View

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 main 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 <u>Ctrl + Shift + R</u> (<u>Command + Shift + R on OS X</u>)	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	Allows you to see the dependencies for the current selected component. See Component Dependencies View for more details.
Resource Hierarchy	Displays the hierarchy for the currently selected resource.
Resource Dependencies	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	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 the 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 Developer 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 Developer has a predefined order of the groups in the **Outline** view:

- For location, the names of the files are sorted alphabetically. The main file is the one you are editing and it 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 Developer sorts the components depending on their order in the document. Oxygen XML Developer 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 **XSLT/XQuery Input**. If the view is not displayed, it can be opened from the **Window > Show View** menu. The tree nodes represent the elements of the documents.

XSLT Input View

If you click a node in the **XSLT 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.

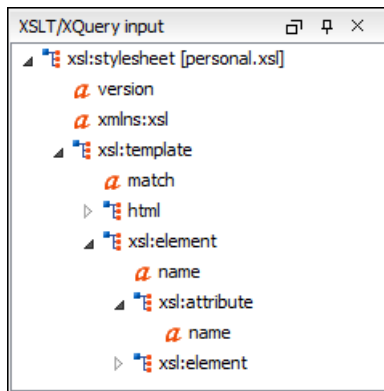


Figure 143: XSLT Input View

For 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:

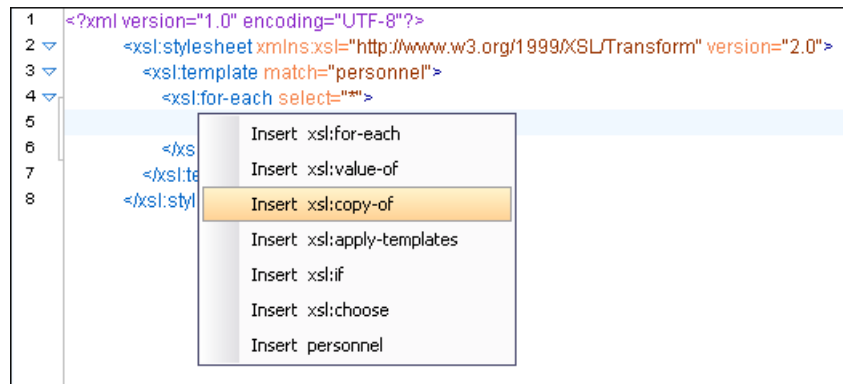


Figure 144: XSLT Input Drag and Drop Pop-up Menu

If you select **Insert xsl:value-of**, for example, the resulting document is:

```
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:value-of select="name/given"/>
6 </xsl:for-each>
7 </xsl:template>
8 </xsl:stylesheet>
```

Figure 145: XSLT 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 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.

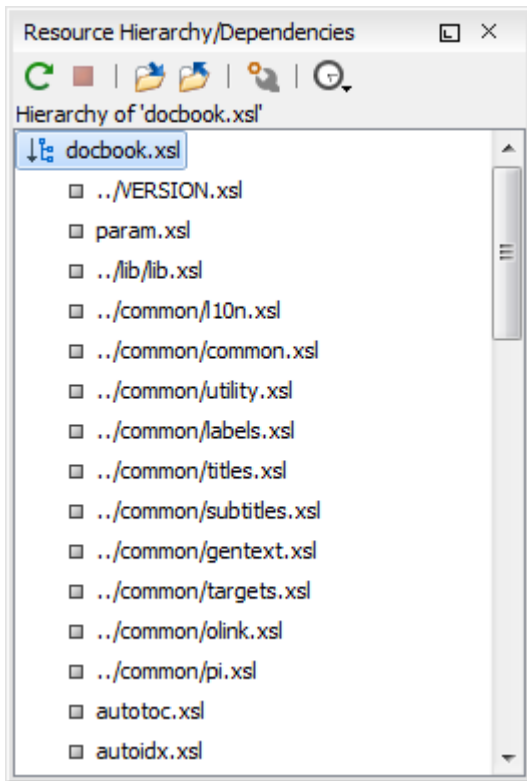


Figure 146: Resource Hierarchy/Dependencies View - Hierarchy for docbook.xsl

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.

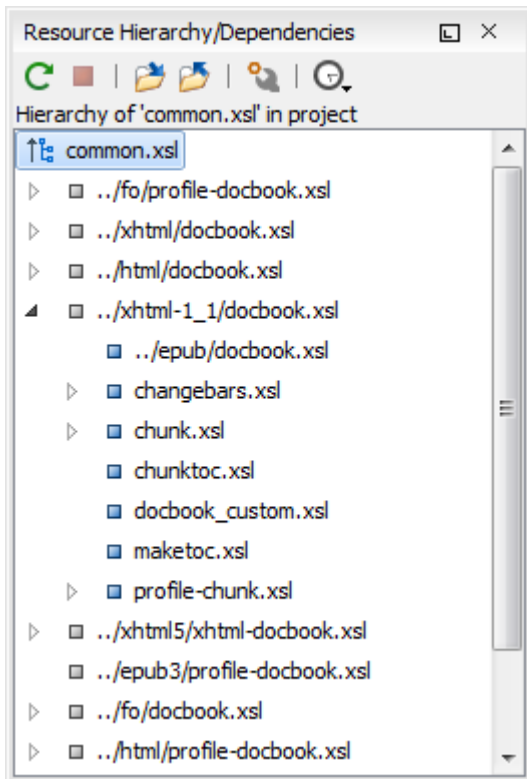


Figure 147: Resource Hierarchy/Dependencies View - Dependencies for common.xsl

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 308

[Search and Refactor Operations Scope](#) on page 303

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** - Enable 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)** - Enable 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 enabled, 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 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).

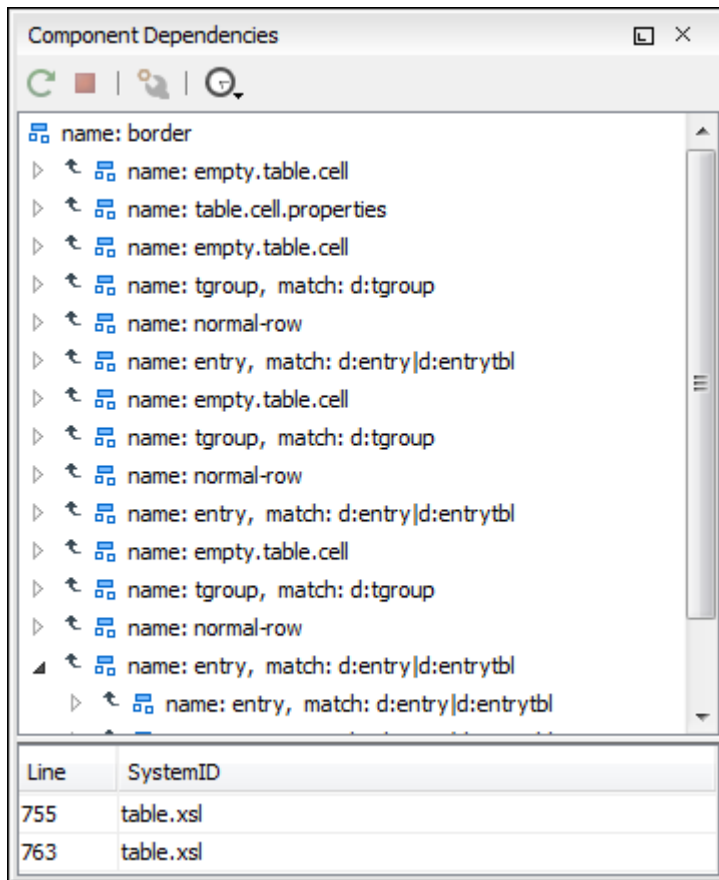


Figure 148: Component Dependencies View - Hierarchy for table.xsl

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 checking 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 303
Search and Refactor Operations Scope

Highlight Component Occurrences

When a component (for example variable or named template) is found at current cursor position, Oxygen XML Developer 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 Developer 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 303
Search and Refactor Operations Scope

XSLT Stylesheet Documentation Support

Oxygen XML Developer 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 Developer 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 while editing use the **Add component documentation** action available in the contextual menu.

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 of a documentation block using Oxygen XML Developer 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 351

Generating Documentation for an XSLT Stylesheet

You can use Oxygen XML Developer 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.

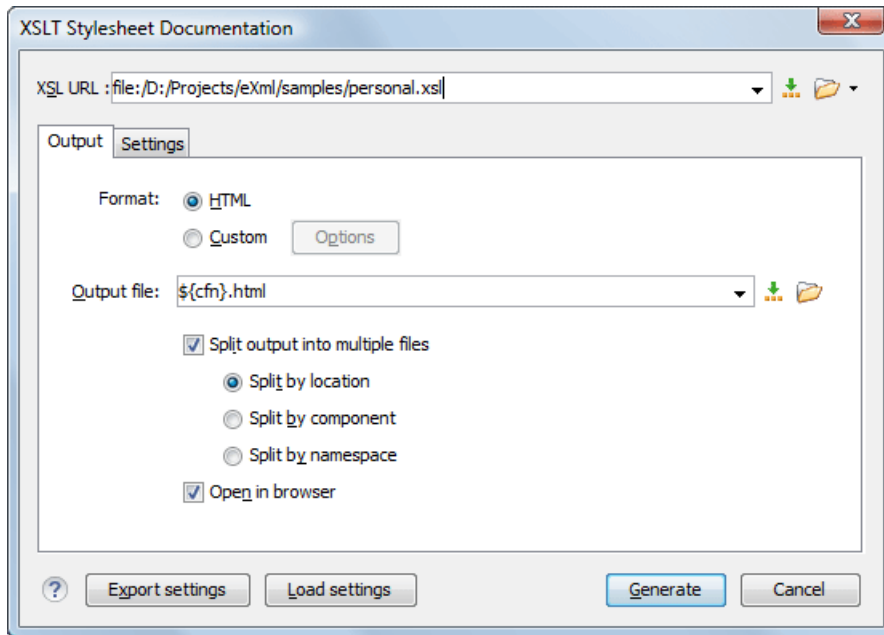


Figure 149: XSLT Stylesheet Documentation Dialog Box

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.

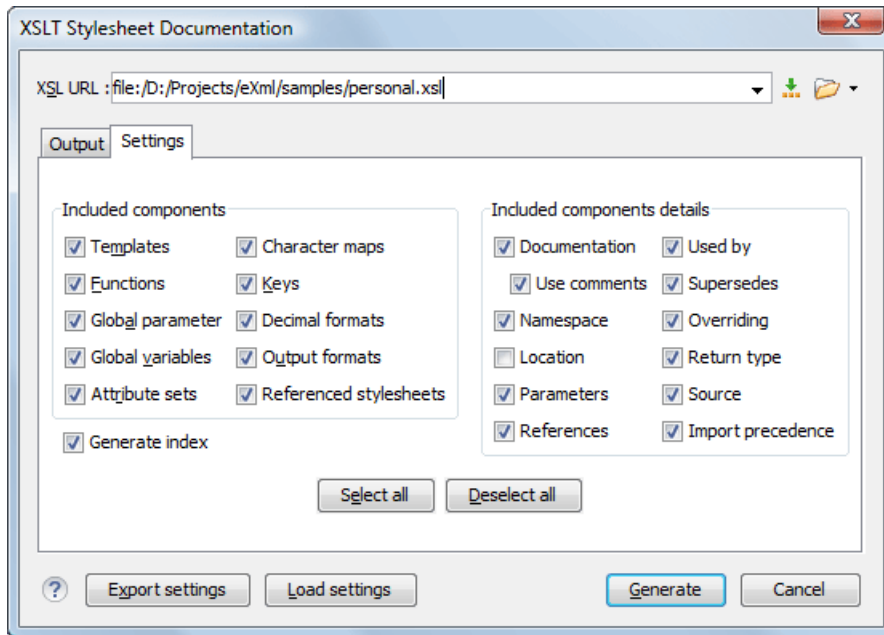


Figure 150: Settings Tab of the XSLT Stylesheet Documentation Dialog Box

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 Developer 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 supersede 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](#).)

Load settings - Reloads the settings from the exported file.

Generate - Use this button to generate the XSLT documentation.

Related information

[XSLT Stylesheet Documentation Support](#) on page 350

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.

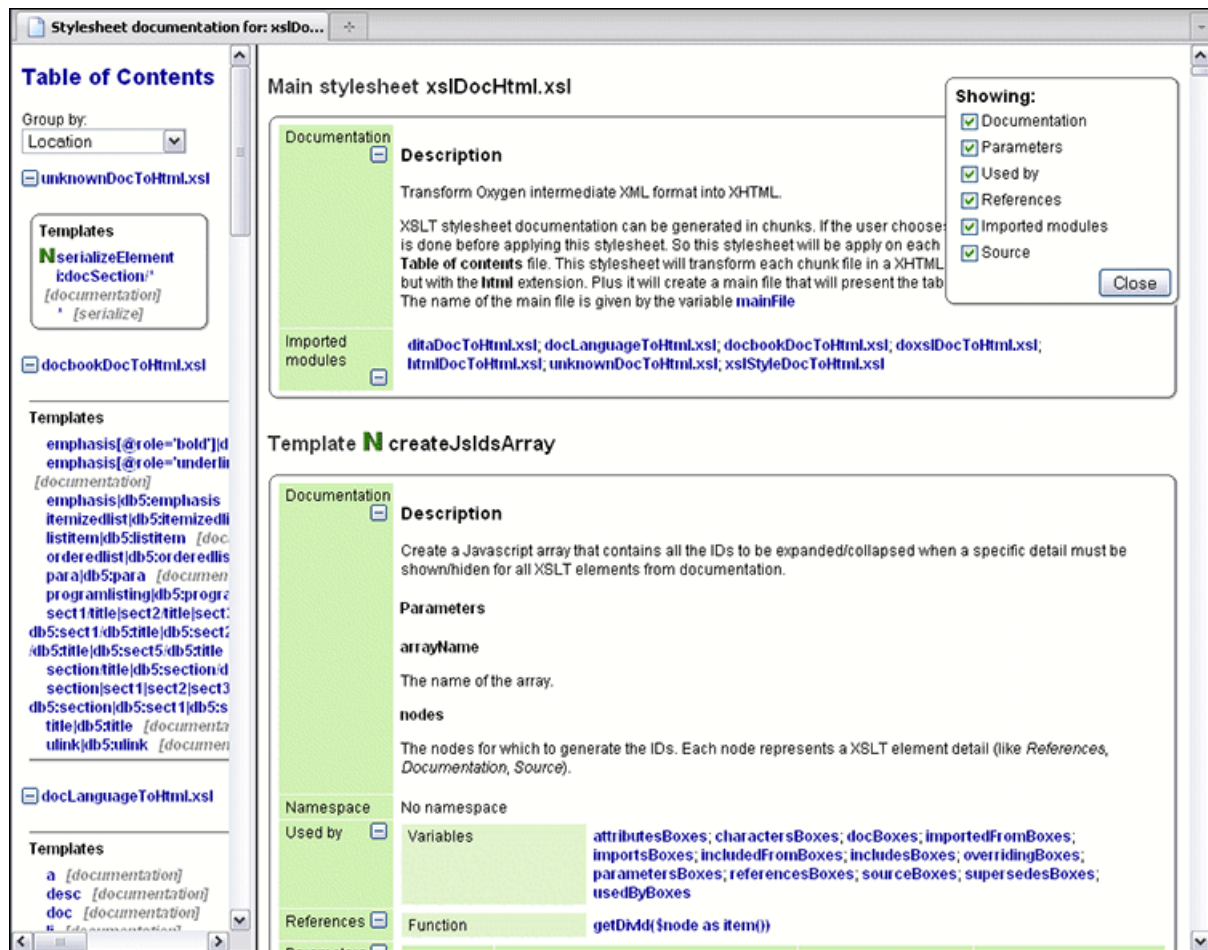


Figure 151: XSLT Stylesheet Documentation Example

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.

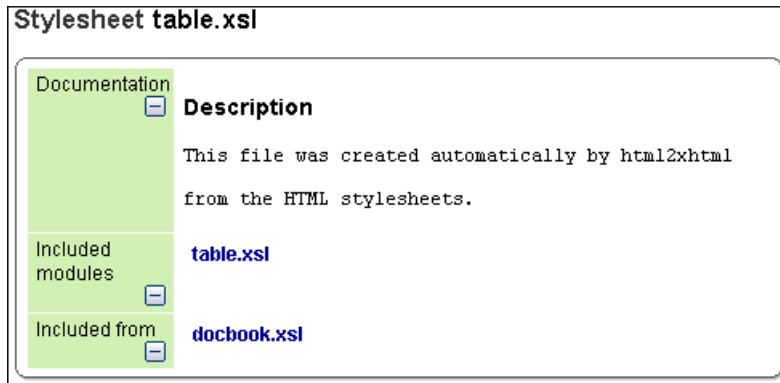


Figure 152: Information About an XSLT Stylesheet

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.

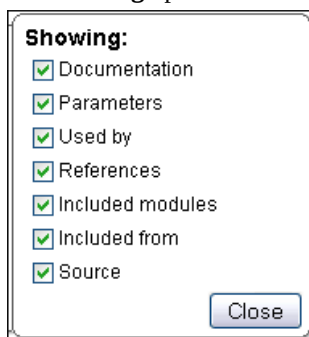


Figure 153: Showing Options

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).

Function func:substring-before-last

Documentation

Description

Get the substring before the last occurrence of the given substring

Parameters

string

The string in which to search

searched

The string to search

Return

The substring starting from the start of the **string** to the index of the last occurrence of **searched**

Namespace

http://www.oxygenxml.com/doc/xsl/functions

Type

xs:string

Used by

Template	Nindex
Function	func:substring-before-last(\$string as item(), \$searched as item())
Variable	indexFile

References

Function	substring-before-last(\$string as item(), \$searched as item())

Parameters

QName	Namespace
searched	No namespace
string	No namespace

Import precedence

7

Source

```
<xsl:function as="xs:string" name="func:substring-before-last">
  <xsl:param name="string"/>
  <xsl:param name="searched"/>
  <xsl:variable name="toReturn">
    <xsl:choose>
      <xsl:when test="contains($string, $searched)">
        <xsl:variable name="before" select="substring-before($string, $searched)"/>
        <xsl:variable name="rec" select="func:substring-before-last(substring-after($string, $searched), $searched)"/>
        <xsl:concat($before, $rec)
      </xsl:when>
      <xsl:otherwise>
        $string
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  $toReturn
</xsl:function>
```

Figure 154: 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`.

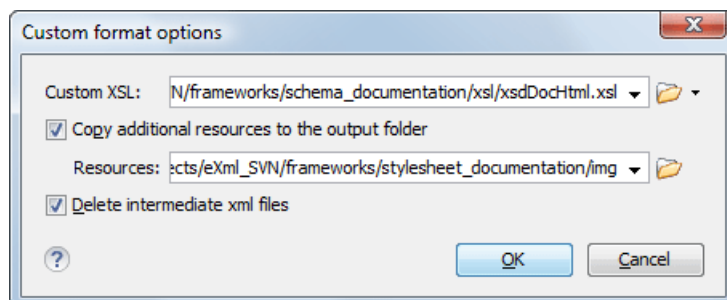


Figure 155: Custom Format Options Dialog Box

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 Developer 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 of an XML Configuration File

```
<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>
      </xsdDocumentationOptions>
    </entry>
  </map>
</serialized>
```

```

<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>

```

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.

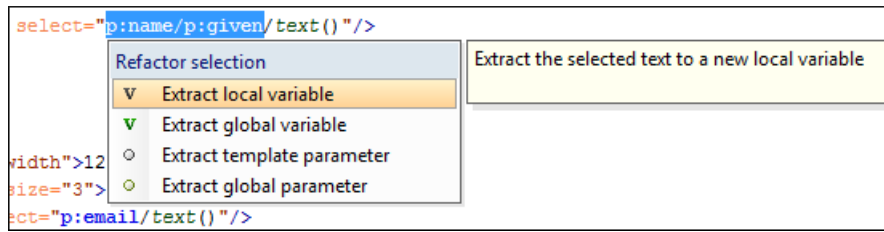


Figure 156: XSLT Quick Assist Support - Refactoring Actions

- 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.

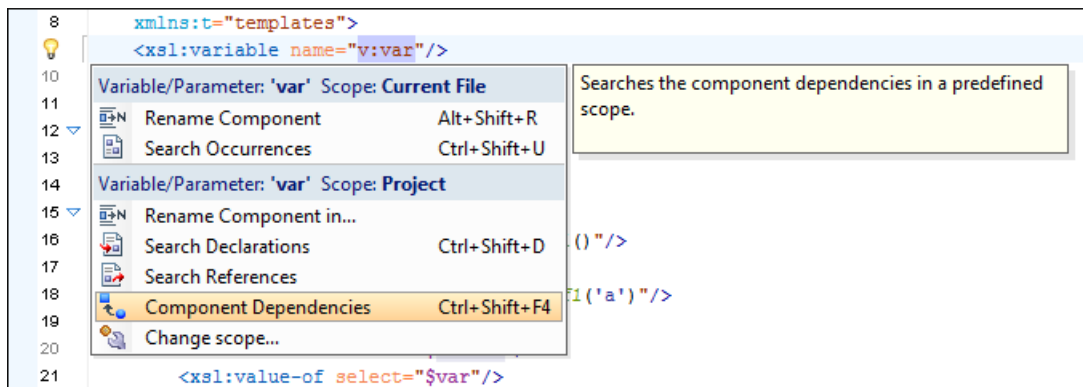


Figure 157: XSLT Quick Assist Support - Component Actions

Related information

[Component Dependencies View](#) on page 349

[XSLT Hierarchy View](#) on page 346

[XSLT Refactoring Actions](#) on page 361

[Search and Refactor Operations Scope](#) on page 303

XSLT Quick Fix Support

The Oxygen XML Developer 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 Developer 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.

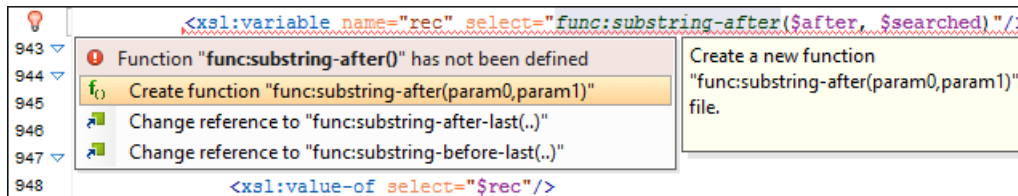


Figure 158: Example of an Undefined XSLT Functions Quick Fix

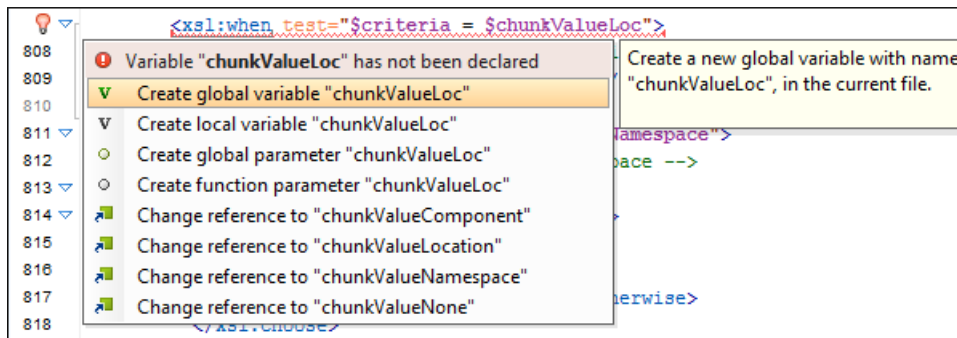


Figure 159: Example of an Undeclared XSLT Variables/Parameters Quick Fix

Oxygen XML Developer 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.

XSLT Refactoring Actions

Oxygen XML Developer 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.

-  **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 Developer 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 an `xsl:if` block to an `xsl:when` block surrounded by an `xsl:choose` element. 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)

-  **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 Developer 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 Developer allows you to edit the name of the variable.



Note: Oxygen XML Developer 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 Developer 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 Developer allows you to edit the name of the parameter.



Note: Oxygen XML Developer 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.

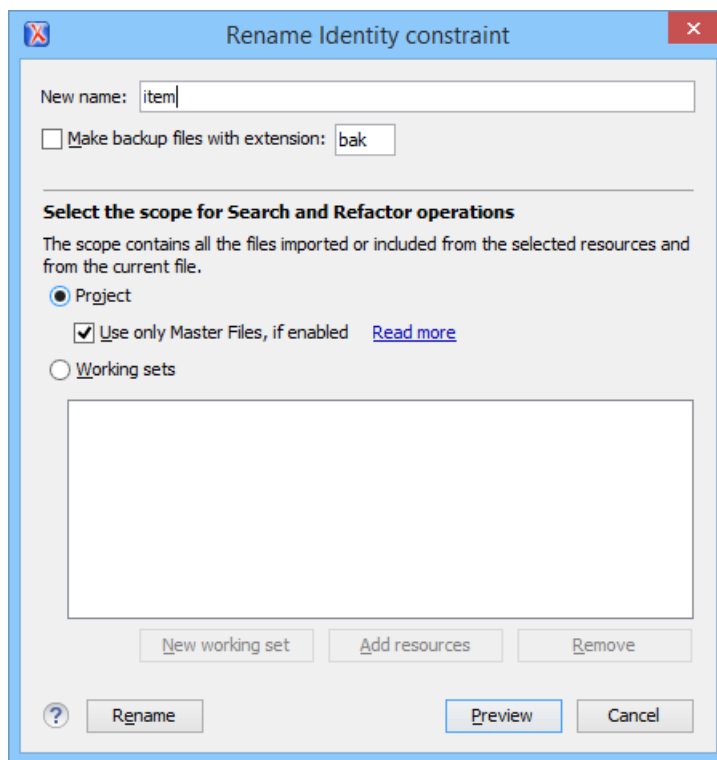


Figure 160: Rename Identity Constraint Dialog Box



Note: Many of these refactoring actions are also proposed by the [Quick Assist support](#).

To watch our video demonstration about XSLT refactoring, go to http://oxygenxml.com/demo/XSL_Refactoring.html.

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.

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 Developer 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.

To run an XSLT Unit Test, open the XSPEC file in an editor and click  **Apply Transformation Scenario(s)** on the main toolbar.

 **Note:** The transformation scenario is defined in the XSPEC *document type*.

When you create an XSpec document based on an XSL document, Oxygen XML Developer uses information from the validation and transformation scenarios associated with the XSL file. From the transformation scenario Oxygen XML Developer uses extensions and properties of Saxon 9.6.0.7, improving the Ant scenario associated with the XSpec document.

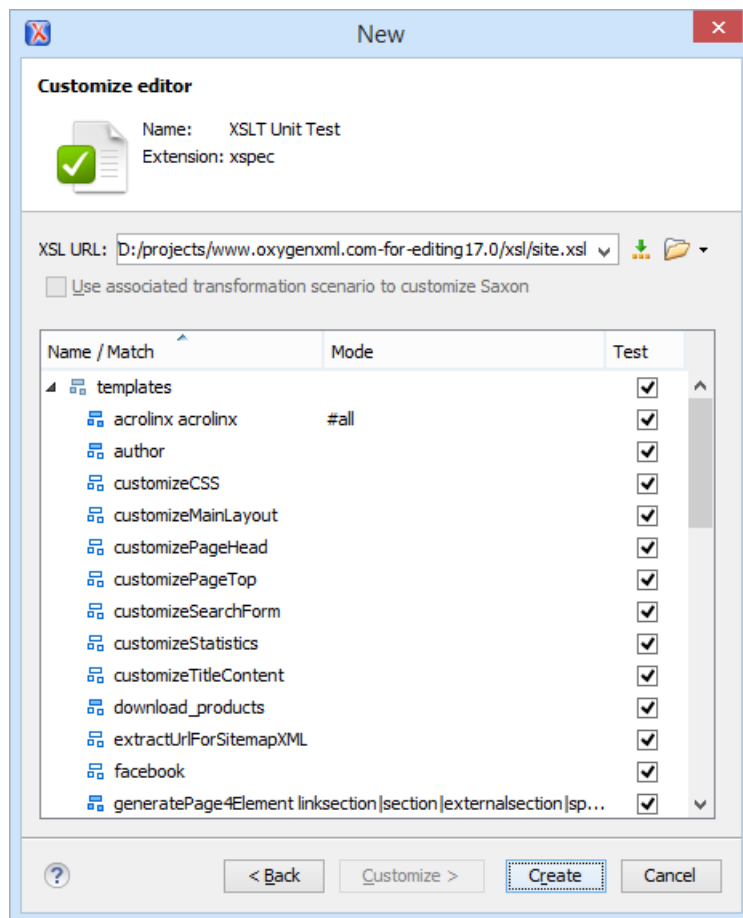


Figure 161: New XSLT Unit Test Wizard

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:call template="createTable">
  <x:param name="nodes">
    <value>A</value>
    <value>B</value>
  </x:param>
  <x:param name="cols" select="2" />
</x:call>
```

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>.

Editing Ant Build Files

This section explains the features of the Ant build file editor.

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 Developer 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 can set a main Ant build file either by using the *master files support from the **Project view***, or *a validation scenario*.

To set a main file using a validation scenario, add validation units that point to the main modules. Oxygen XML Developer 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 main 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.

Validate Ant Build Files

Oxygen XML Developer 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](#) on page 120 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 Developer 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 Developer (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. 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.

Content Completion in Ant Build Files

The items in the list of proposals offered by the **Content Completion Assistant** are context-sensitive.

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 Developer (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

<http://ant.apache.org/manual/properties.html>

Syntax Highlight

To change the syntax highlighting colors for Ant build files, [open the Preferences dialog box \(Options > Preferences\)](#) and go to **Editor > Syntax Highlight**.

Related information

[Syntax Highlight Preferences](#) on page 87

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 from the **Window > Show View** menu.

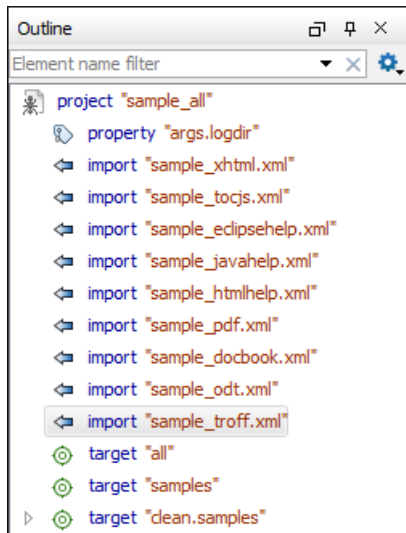


Figure 162: Ant Outline View

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 main file. This option is set by default.

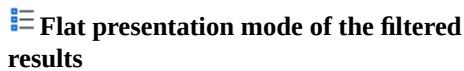
Show only local components

Displays the components defined in the current file only.

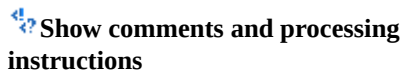
Group by location/type

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**

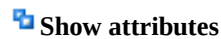
When active, the application flattens the filtered result elements to a single level.

**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**Allows you to see the dependencies for the current selected component. See [Ant Component Dependencies View](#) for more details.**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 Developer has a predefined order for the groups in the **Outline** view:

- For location, the names of the files are sorted alphabetically. The main file is the one you are editing and it is located at the top of the list.
- For type, the order is: properties, targets, references.



Note: When no grouping is available Oxygen XML Developer sorts the components depending on their order in the document. Oxygen XML Developer 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 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 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 checking 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 303

Search and Refactor Operations Scope

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 Developer 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 configure it, *open the **Preferences** dialog box (**Options > Preferences**)* and go to **Editor > Mark Occurrences**. If the automatic feature is disabled for a particular type of file, 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 97

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 303
Search and Refactor Operations Scope

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 370

[Ant resource Hierarchy/Dependencies View](#) on page 369

[Ant Refactoring Actions](#) on page 372

[Search and Refactor Operations Scope](#) on page 303

Ant Quick Fix Support

The Oxygen XML Developer 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 Developer 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 Developer 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.

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.

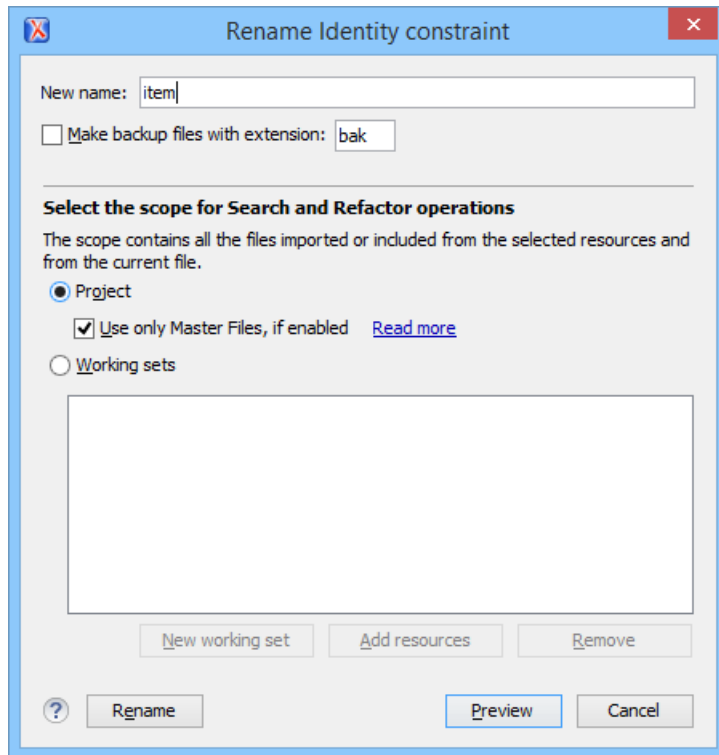


Figure 163: Rename Identity Constraint Dialog Box

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 Developer 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.

XML Schema Diagram Editor (Design Mode)

XML Schemas enable document designers to specify the allowed structure and content of an XML document and to check if an XML document is valid.

Oxygen XML Developer 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 Developer 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.

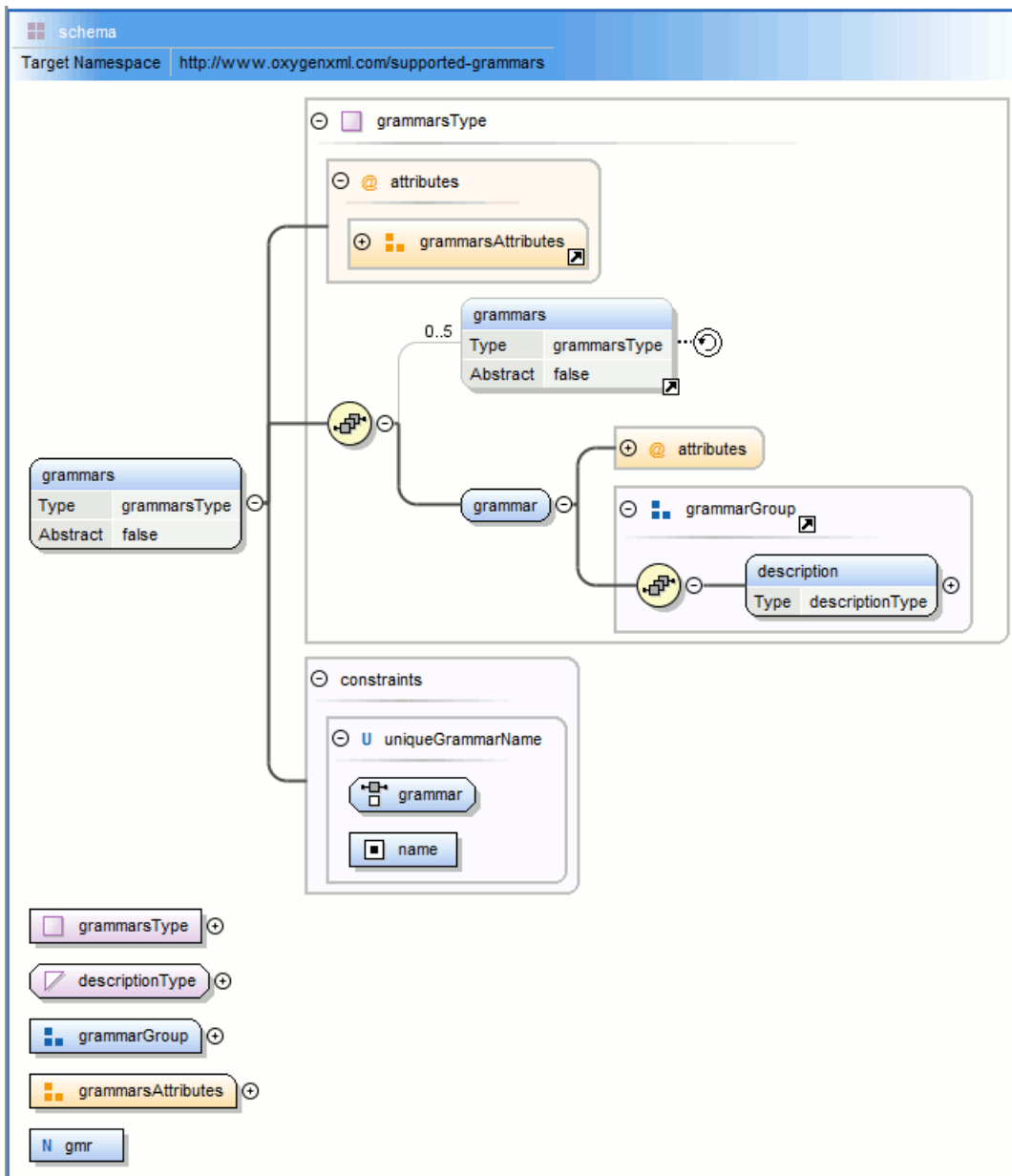


Figure 164: XML Schema Diagram

To watch our video demonstration about the basic aspects of designing an XML Schema using the new Schema Editor, go to http://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 Developer.
-  **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.

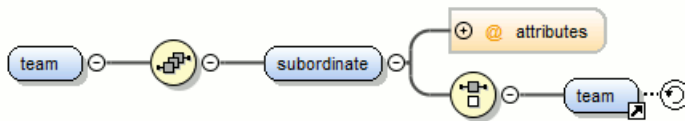


Figure 165: Recursive Reference

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/define component. An edit dialog box is displayed, allowing you to customize the directives.

Related information

[Searching and Refactoring Actions in XML Schemas](#) on page 413

[Component Dependencies View for XML Schema](#) on page 411

[XML Schema Resource Hierarchy / Dependencies View](#) on page 409

[Generating Sample XML Files](#) on page 415

[Schema Design Preferences](#) on page 78

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 **X 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.

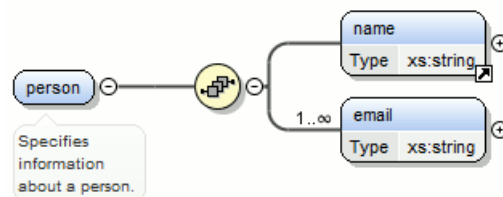


Figure 166: Extracting a Global Element

If you use the **Extract Global Element** action on a **name** element, the result is:

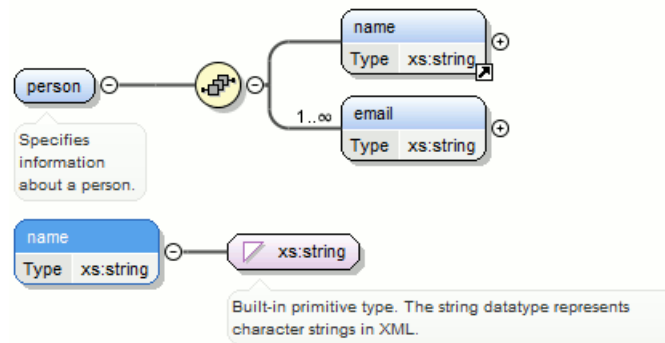


Figure 167: Extracting a Global Element on a name Element

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.

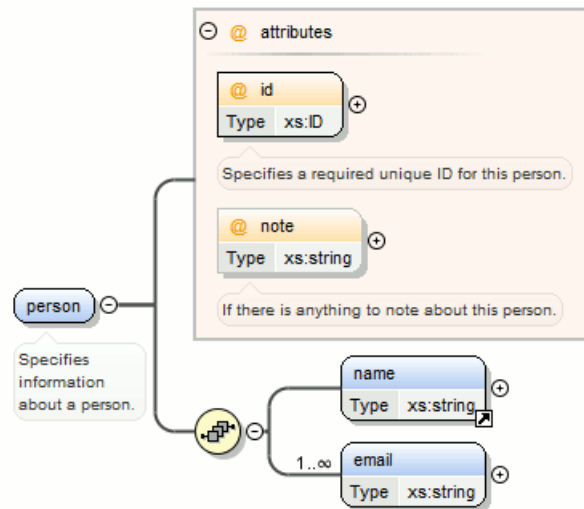


Figure 168: Extracting a Global Attribute

If you use the **Extract Global Attribute** action on a `note` attribute, the result is:

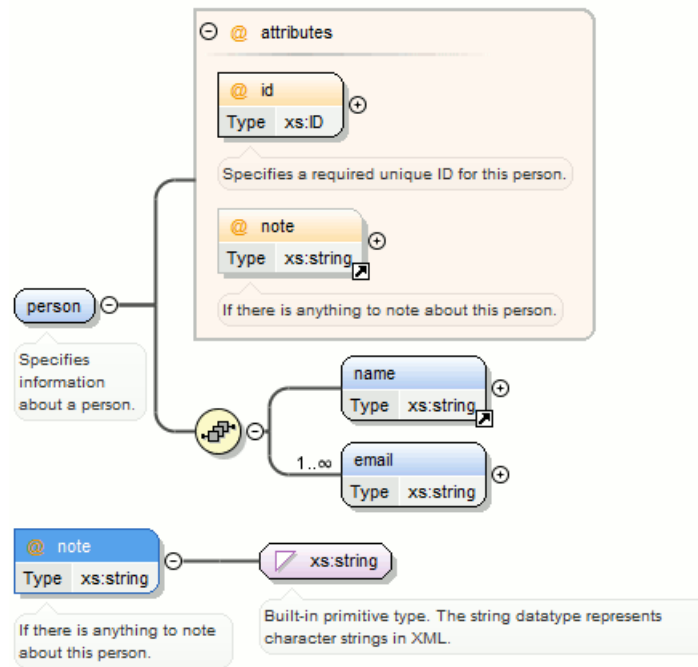
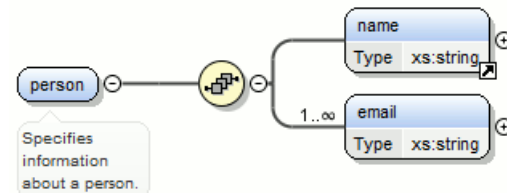


Figure 169: Extracting a Global Attribute on a note Attribute

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 enabled 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 170: Extracting a Global Group

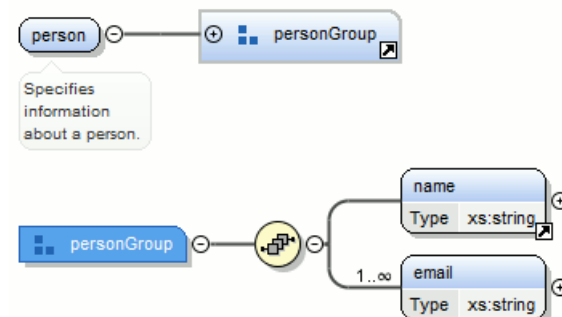


Figure 171: Extracting a Global Group on a sequence Element

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.

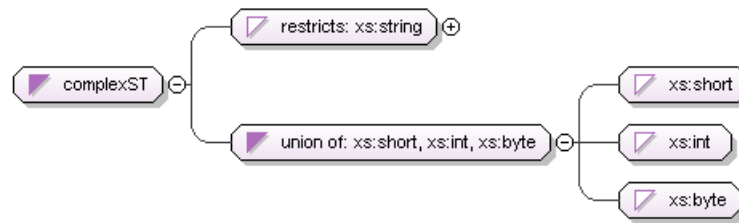


Figure 172: Extracting a Global Simple Type

If you use the action on the `union` component and choose `numericST` for the new global simple type name, the result is:

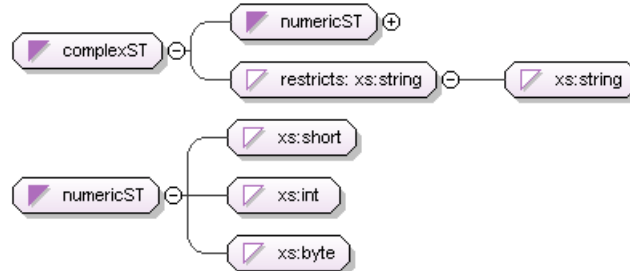


Figure 173: Extracting a Global Simple Type on a union Component

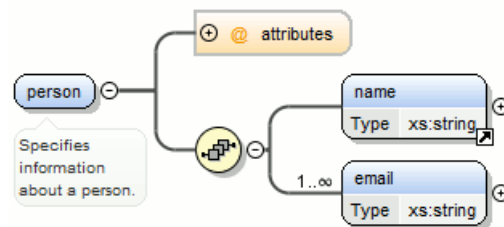


Figure 174: Extracting a Global Complex Type

If you use the action on a `person` element and choose `person_type` for the new complex type name, the result is:

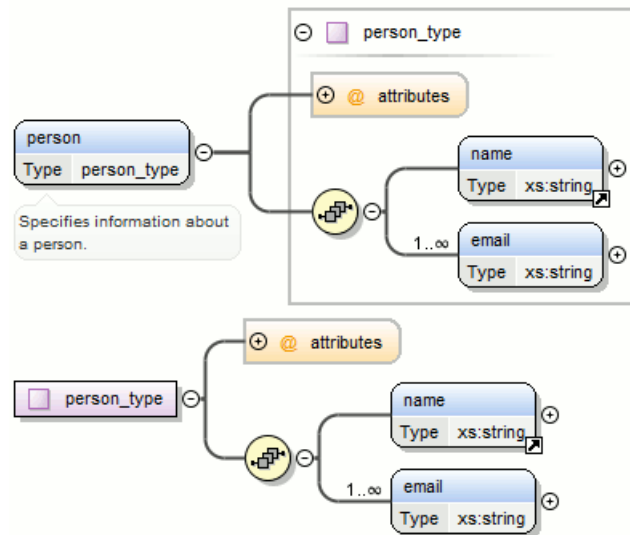


Figure 175: Extracting a Global Complex Type on a person Element

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	Searches all references of the item found at current cursor position in the defined scope if any.
References	
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	Allows you to see the dependencies for the current selected component.
Resource Hierarchy	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).

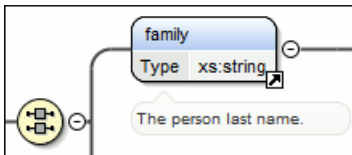


Figure 176: Example: Required Component

- A thin line to identify a connection with an optional component (in the following image, `email` is an optional element).

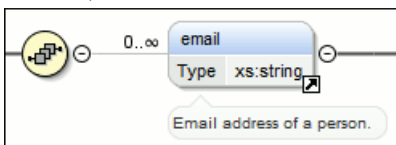
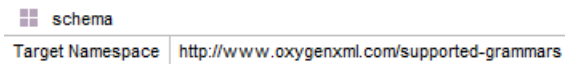


Figure 177: Example: Optional Component

The following topics explain in detail all available components and their symbols as they appear in an XML schema diagram.

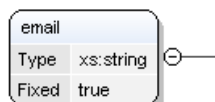
xs:schema**Figure 178: The xs:schema Component**

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.

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 <code>block</code> attribute of <code>xs:element</code> and <code>xs:complexType</code> .	#all, extension, restriction, substitution, restriction extension, restriction substitution, extension substitution, restriction extension substitution, [Empty].
Final Default	Default value of the <code>final</code> attribute of <code>xs:element</code> and <code>xs:complexType</code> .	#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 179: 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>.

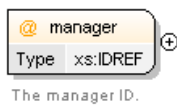
An element by default displays the following properties when rendered in the diagram: *default*, *fixed*, *abstract* and *type*. When referenced or declared locally, the element graphical representation also contains the value for the *minOccurs* and *maxOccurs* properties (for 0..1 and 1..1 occurs the values are implied by the connector style) and the connectors to the element are drawn using dotted lines if the element is optional.

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.
Min Occurs	Minimum number of occurrences of the element.	A numeric positive value. Default value is 1	Only for references/local elements

Property Name	Description	Possible Values	Mentions
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
Nilable	When this attribute is set to true, the element can be declared as nil using an <code>xsi:nil</code> 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.
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 <code>xsi:type</code> 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 <code>blockDefault</code> attribute of the parent <code>xs:schema</code> .	#all, restriction, extension,substitution, extension restriction, extension substitution, restriction substitution, restriction extension substitution	For global elements and element references

Property Name	Description	Possible Values	Mentions
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 <code>finalDefault</code> attribute of the parent <code>xs:schema</code> .	#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 180: 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.

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.

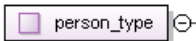
Property Name	Description	Possible Value	Mentions
Default	Default value. When specified, an attribute is added by the schema processor (if it is missing from the instance XML document) and it is given this value. The default and fixed attributes are mutually exclusive.	Any string	For all local or global 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 <code>attributeFormDefault</code> attribute of the <code>xs:schema</code> document element.	unqualified/qualified	For local attributes.
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 181: 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>.

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 182: 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 info set 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.

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.

Property Name	Description	Possible Values	Mentions
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.
Abstract	When set to <code>true</code> , this complex type cannot be used directly in the instance documents and needs to be substituted using an <code>xsi:type</code> attribute.	true/false	For global and anonymous complex types.
Block	Controls if a substitution (either through a <code>xsi:type</code> 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 <code>blockDefault</code> attribute of <code>xs:schema</code> .	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 <code>defaultAttributes</code> attribute, which identifies an attribute group. Each <code>complexType</code> defined in the schema document then automatically includes that attribute group, unless this is overridden by the <code>defaultAttributesApply</code> attribute on the <code>complexType</code> element.	true/false	This property is available only for XML Schema 1.1.

Property Name	Description	Possible Values	Mentions
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 183: 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.

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).

Name	Description	Possible Values	Scope
Member Types	Category for grouping union members.	Not editable property.	For global and anonymous simple types with the derivation method set to union.
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.

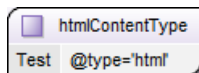


Figure 184: The xs:alternative Component

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 185: 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.

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.
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.

Property Name	Description	Possible Values	Mentions
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
 include: xhtml11-model-1.xsd
Figure 186: 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>.

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
 import: http://www.renderx.com/XSL/Extensions (rxxsd.xsd)
Figure 187: 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>.

xs:import Properties

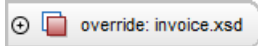
Property Name	Description	Possible Values
Schema Location	Imported schema location	Any URI
Namespace	Imported schema namespace	Any URI
ID	Import ID	Any ID
Component	The component name	Not editable property.

xs:redefine
 redefine: ../personal.xsd
Figure 188: 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>.

xs:redefine Properties

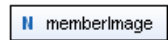
Property Name	Description	Possible Values
Schema Location	Redefine schema location.	Any URI
ID	Redefine ID	Any ID
Component	The component name.	Not editable property.

xs:override**Figure 189: 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>.

xs:override Properties

Property Name	Description	Possible Values
Schema Location	Redefine schema location.	Any URI
ID	Redefine ID	Any ID

xs:notation**Figure 190: 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>.

xs:notation Properties

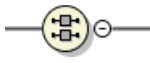
Property Name	Description	Possible values	Mentions
Name	The notation name. Always required.	Any NCName.	If missing, will be displayed as '[notation]' in diagram.
System Identifier	The notation system identifier.	Any URI	Required if public identifier is absent (otherwise, optional).
Public Identifier	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.
System ID	The component system id.	Not editable property.	For all notations.

xs:sequence / xs:choice / xs:all**Figure 191: 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 192: 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 193: 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.

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 194: 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.

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

Property Name	Description	Possible Values
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 195: 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>.

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

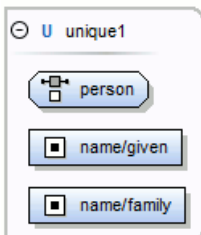


Figure 196: The xs:unique Component

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>.

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

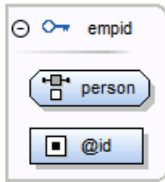


Figure 197: The xs : key Component

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>.

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

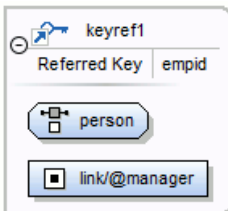


Figure 198: The xs : keyRef Component

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.

xs : keyRef Properties

Property Name	Description	Possible Values
Name	The keyref name. Always required.	Any NCName.

Property Name	Description	Possible Values
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 199: 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>.

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 200: 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>.

xs:field Properties

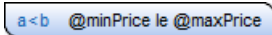
Property Name	Description	Possible Values
XPath	Relative XPath expression identifying the field(s) composing the key, key reference, or unique constraint.	An XPath expression.
ID	The component id.	Any ID.
Component	The edited component name.	Not editable property.
System ID	The component system id.	Not editable property.

xs:assert

Assertions provide a flexible way to control the occurrence and values of elements and attributes available in an XML Schema.



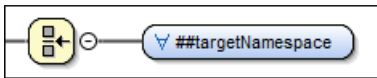
Note: `xs:assert` is available for XML Schema 1.1.


Figure 201: The xs:assert Component

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 202: 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>.

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

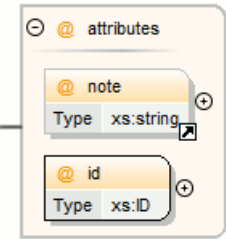


Figure 203: Attributes Construct

Groups all attributes and attribute groups belonging to a complex type.

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

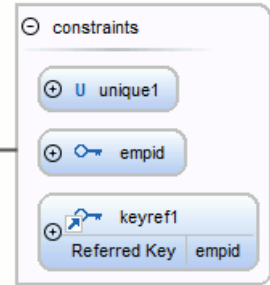


Figure 204: Constraints Construct

Groups all constraints ([xs:key](#), [xs:keyRef](#) or [xs:unique](#)) belonging to an element.

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

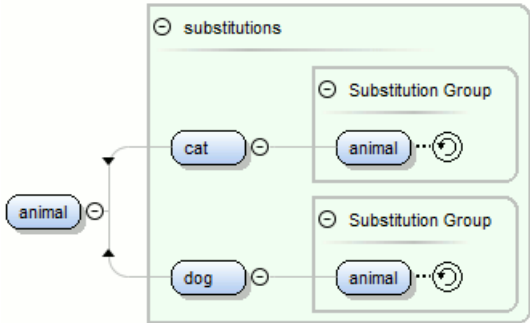


Figure 205: Substitutions Construct

Groups all elements that can substitute the current element.

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 Developer [XML documents validation](#) capability.

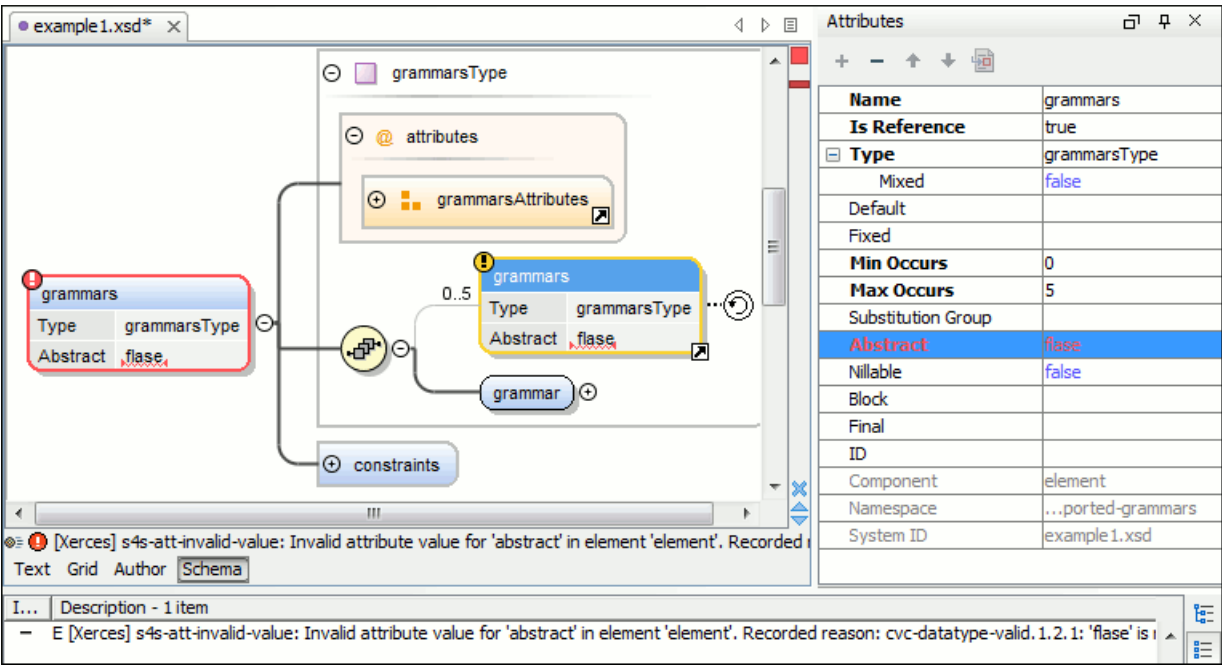


Figure 206: XML Schema Validation

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.

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 from the **Window > Show View** menu.

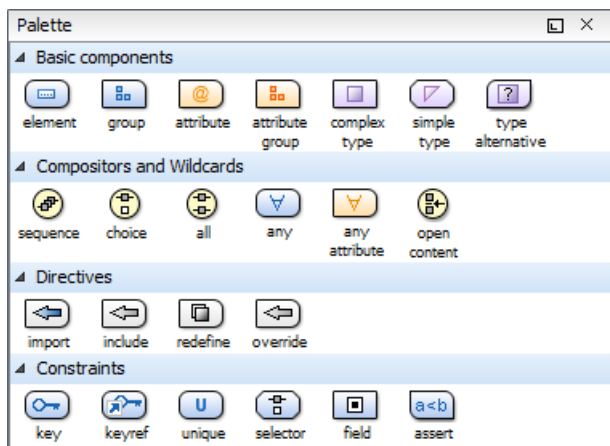


Figure 207: Palette View

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

http://oxygenxml.com/demo/Schema_Palette.html and http://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 from the **Window > Show View** menu.

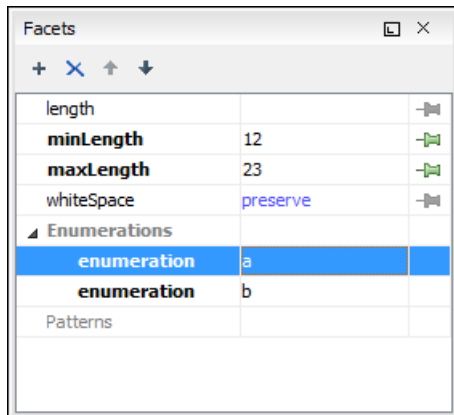


Figure 208: Facets View

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.

Editing XML Schema in Text Editing Mode

The Oxygen XML Developer **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, enable the *Show Full Model XML Schema diagram* checkbox from 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 Developer 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 main file using a validation scenario, add validation units that point to the main schemas. Oxygen XML Developer 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 includes components defined in modules other than the currently edited one.
- The **Outline** 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 disable 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 Developer 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 282

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.

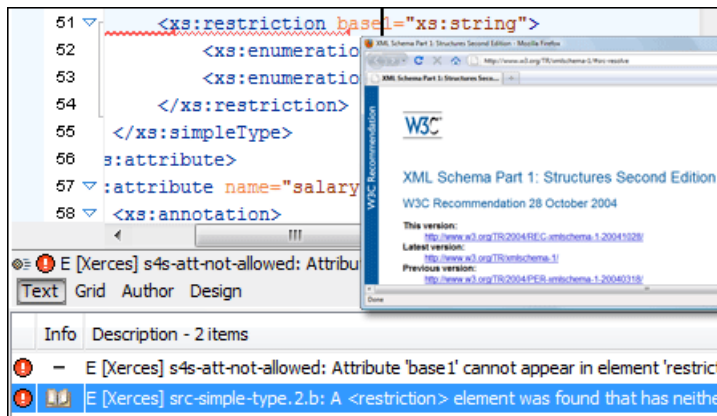


Figure 209: Link to Specification for XML Schema Errors

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 Developer window. Otherwise, for large values of the `minOccurs` and `maxOccurs` attributes, the validator fails with an *OutOfMemory* error that might make Oxygen XML Developer 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.

Content Completion in XML Schema

The intelligent **Content Completion Assistant** available in Oxygen XML Developer enables rapid, in-line identification and insertion of elements, attributes and attribute values valid in the 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 disable the *Enable content completion* option.

When active, the **Content Completion Assistant** displays a list of context-sensitive proposals valid at the current cursor position. Elements can be selected in the list 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 Developer populates the proposals list with information taken either from XML Schema 1.0 or 1.1.

Oxygen XML Developer 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.

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 from the **Window > Show View** menu.

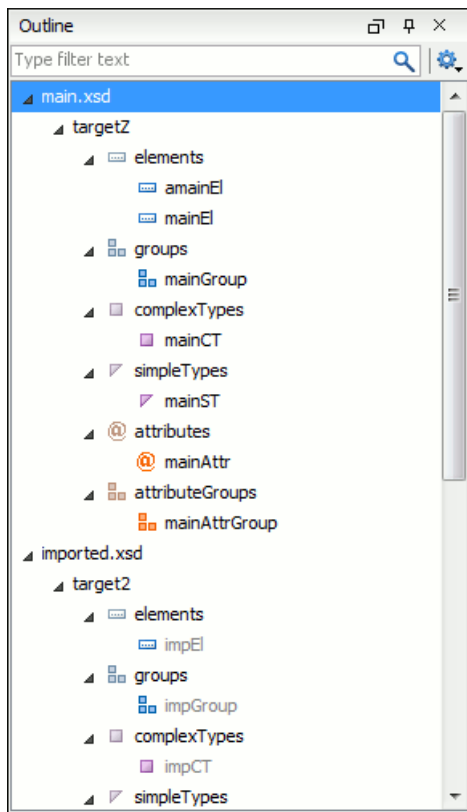


Figure 210: Outline View for XML Schema

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 main 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 main 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)	Allows you to see the dependencies for the current selected component.
Resource Hierarchy (F4)	Allows you to see the hierarchy for the current selected resource.
Resource Dependencies (Shift + F4)	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 413

[Component Dependencies View for XML Schema](#) on page 411

[XML Schema Resource Hierarchy / Dependencies View](#) on page 409

[Generating Sample XML Files](#) on page 415

[Editing Relax NG Schema in the Master Files Context](#) on page 464

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 from the **Window > Show View** menu.

Name	Value
family	
Type	[ST - union]
Member Types	
Member	xs:string
Default	
Fixed	
Substitution Group	
Abstract	false
Nillable	false
Block	
Final	
ID	
Component	element
Namespace	
System ID	personal.xsd

Figure 211: 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 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 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.

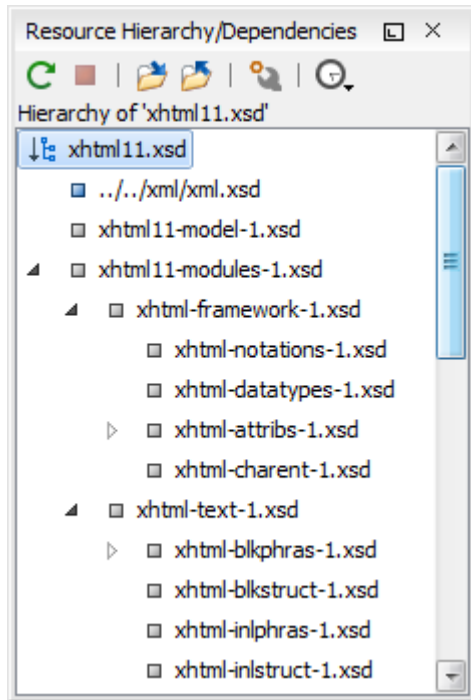


Figure 212: Resource Hierarchy/Dependencies View - Hierarchy for xhtml11.xsd

The build process for the dependencies view is started with the **Resource Dependencies** action available on the contextual menu.



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 308

[Search and Refactor Operations Scope](#) on page 303

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** - Enable 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)** - Enable 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 enabled, 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 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).

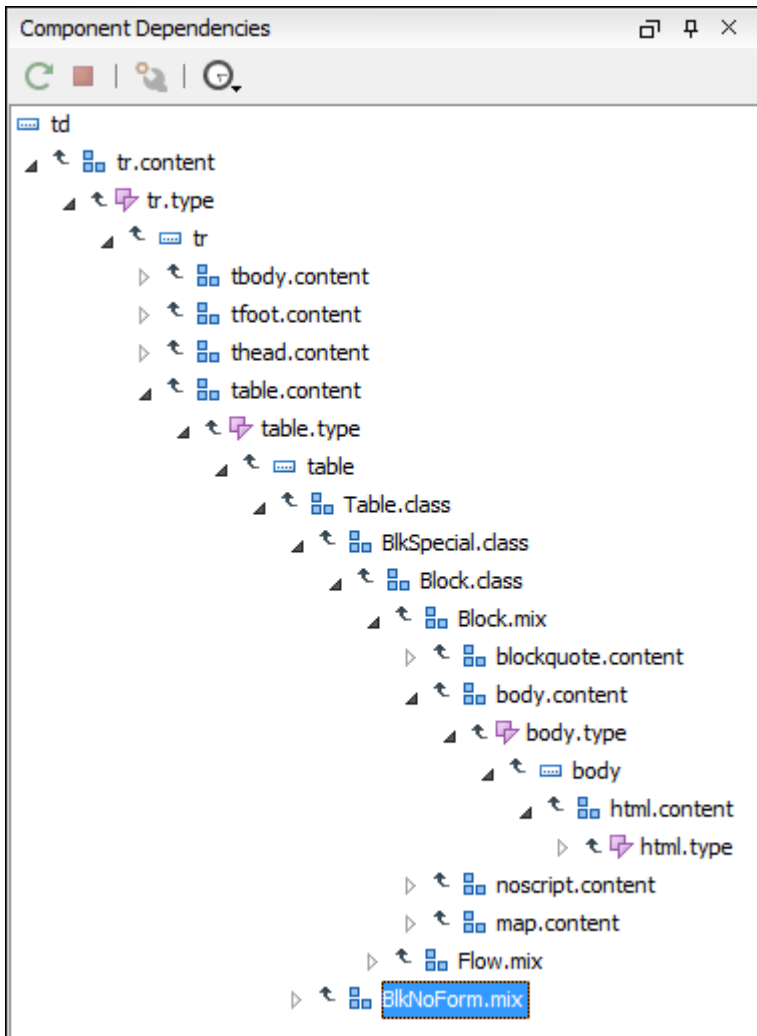


Figure 214: Component Dependencies View - Hierarchy for xhtml11.xsd

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 303

Highlight Component Occurrences

When a component (for example types, elements, attributes) is found at current cursor position, Oxygen XML Developer 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 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 ()** 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.

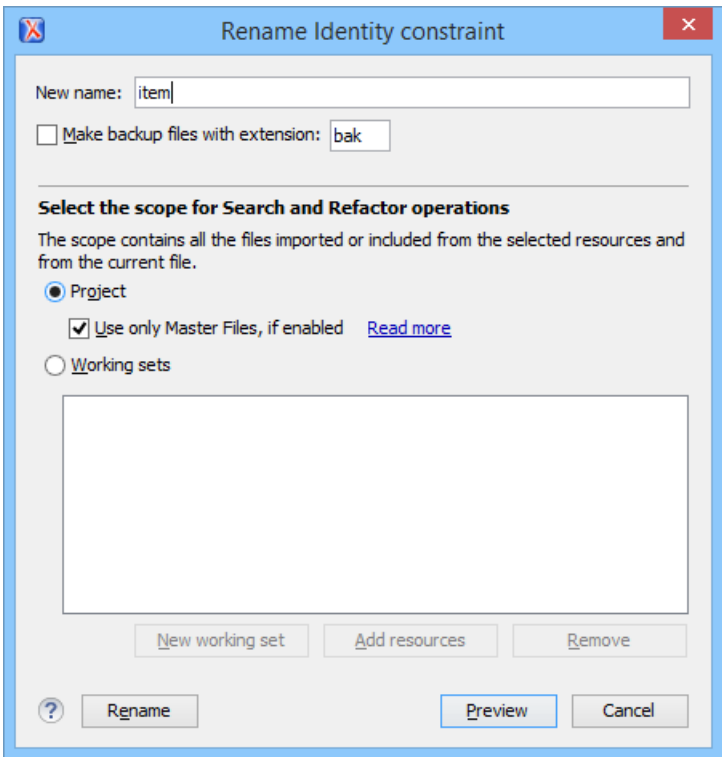


Figure 215: Rename Identity Constraint Dialog Box

Related information

[Search and Refactor Operations Scope](#) on page 303
Search and Refactor Operations Scope

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.

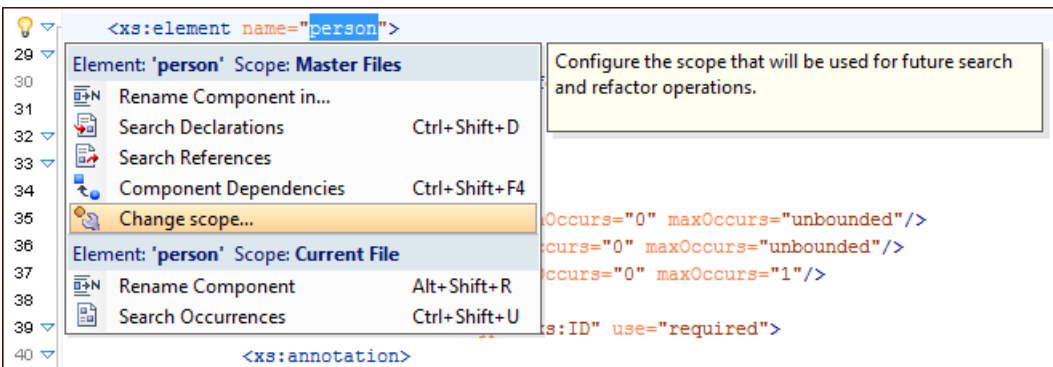


Figure 216: Quick Assist Support

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 http://oxygenxml.com/demo/Quick_Assist.html.

Related information

[Resource Hierarchy / Dependencies View](#) on page 409

[Component Dependencies View](#) on page 411

[Searching and Refactoring Actions](#) on page 413

Generating Sample XML Files

Oxygen XML Developer 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 http://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.

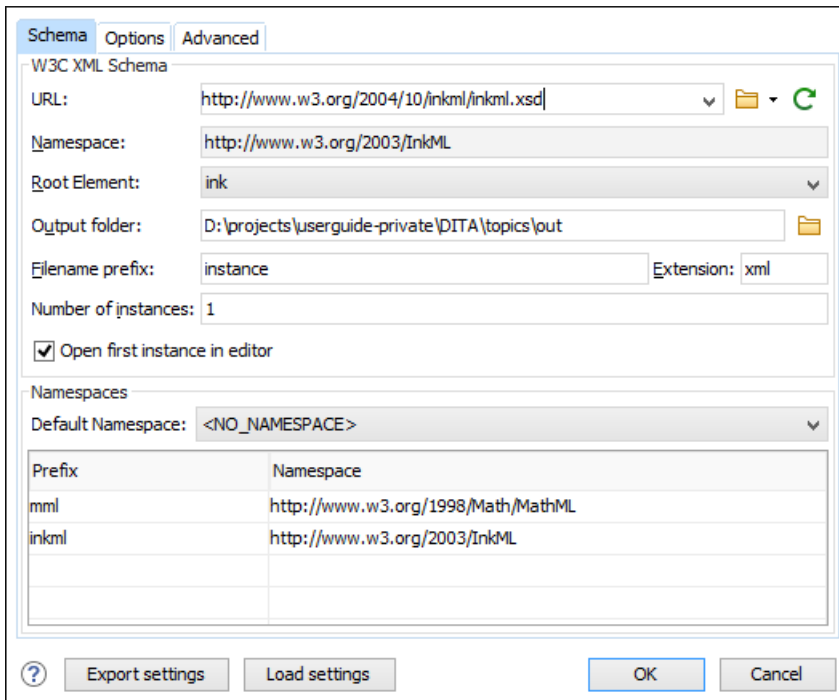


Figure 217: Generate Sample XML Files Dialog Box (Schema Tab)

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: <code>prefixN.extension</code> , where N represents an incremental number from 0 up to the specified <i>Number of instances</i> .
Number of instances	The number of XML files to be generated.
Open first instance in editor	When checked, 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.
Load settings	Use this button to load previously exported settings.
Export settings	Use this button to save the current settings for future use.

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.

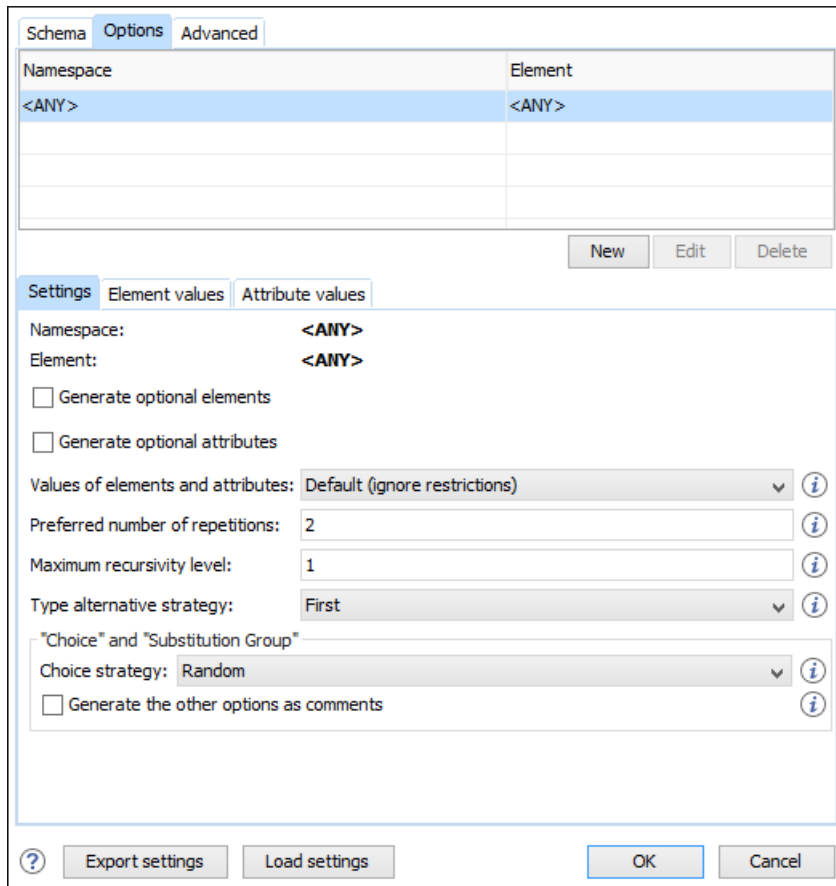


Figure 218: Generate Sample XML Files Dialog Box (Options Tab)

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 checked, all elements are generated, including the optional ones (having the <code>minOccurs</code> attribute set to 0 in the schema).
	Generate optional attributes	When checked, all attributes are generated, including the optional ones (having the <code>use</code> attribute set to <code>optional</code> 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 XML Instances Generator preferences page). Note that type restrictions are ignored when this option is enabled. 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 **XML Instances Generator** 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 enabled, 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.

Load settings Use this button to load previously exported settings.

Export settings Use this button to save the current settings for future use.

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.

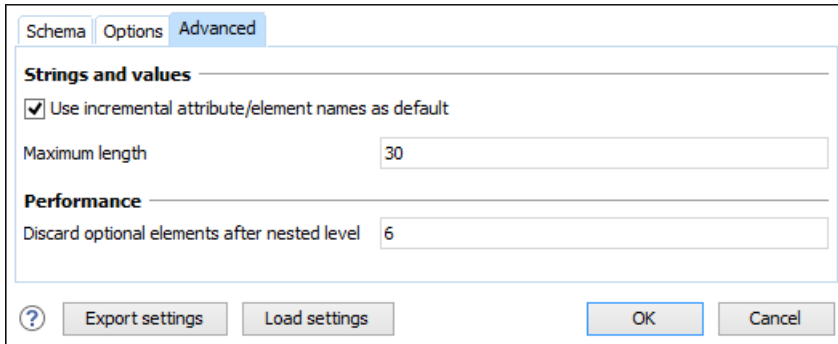


Figure 219: Generate Sample XML Files Dialog Box (Advanced Tab)

This tab includes the following options:

- Use incremental attribute / element names as default** If checked, 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 checked, 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.

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 Developer 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:

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 Developer 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.

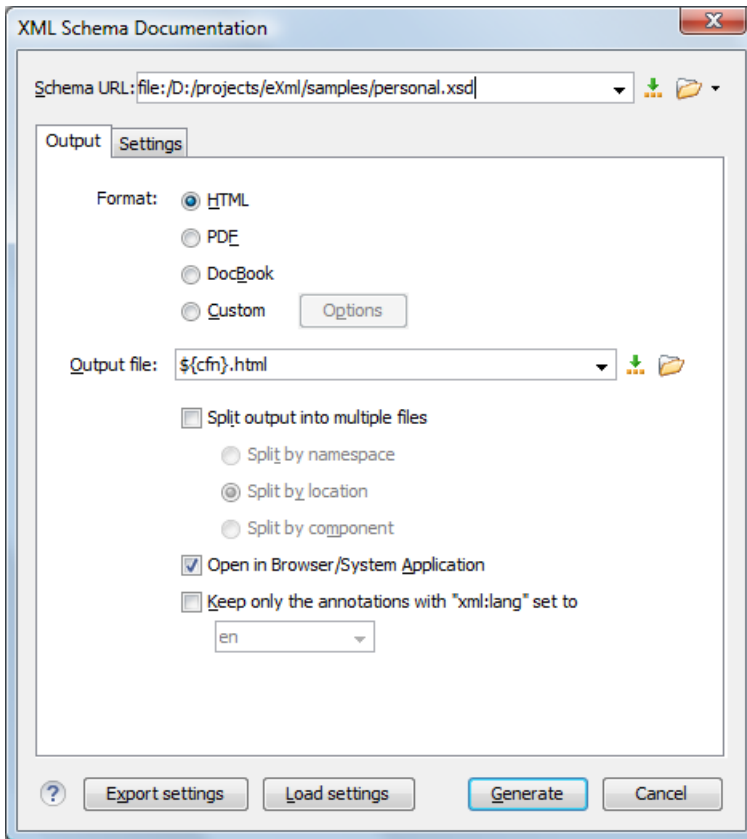


Figure 220: XML Schema Documentation Dialog Box

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](#).
 - **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.



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.

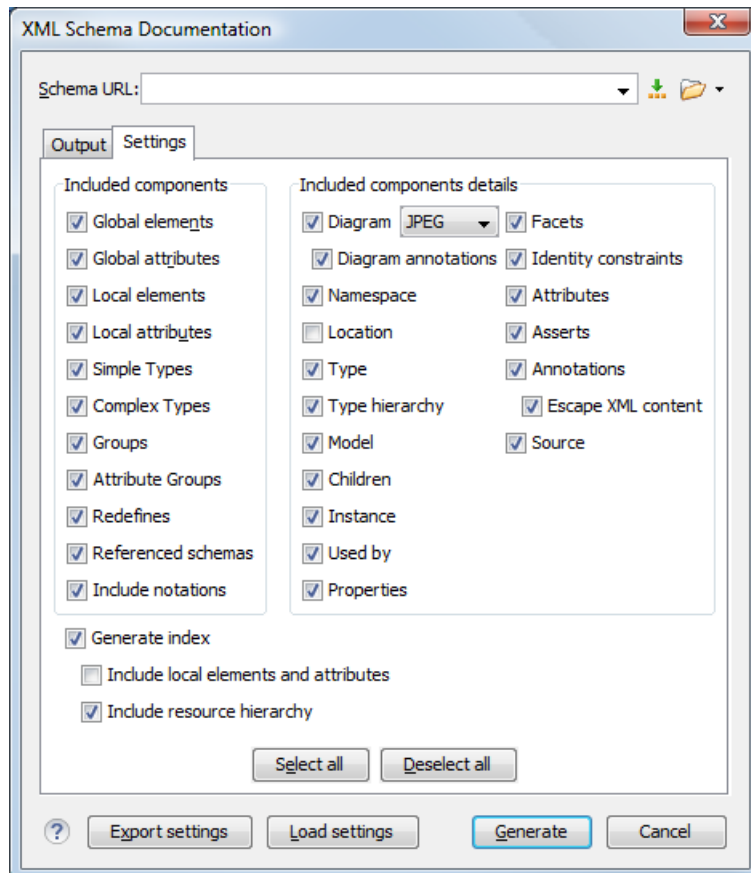


Figure 221: Settings Tab of the XML Schema Documentation Dialog Box

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 checked, 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 disabled 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](#).)

Load 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 426

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 Developer XSD editor panel](#).

Figure 222: 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.

Namespace	No namespace
Properties	Attribute Form Default: unqualified Element Form Default: unqualified
Schema location	file:/D:/personal.xsd

Figure 223: 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.

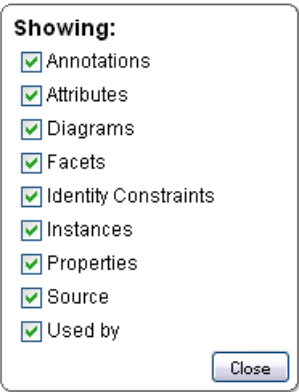


Figure 224: Showing Options

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.

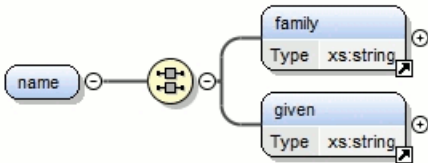
Namespace	No namespace	
Annotations	Specifies the person family and given name.	
Diagram		
Properties	Content	complex
Used by	Element	person
Model	ALL(family given)	
Children	family, given	
Instance	<pre><name> <family>{1,1}</family> <given>{1,1}</given> </name></pre>	
Source	<pre><xs:element name="name"> <xs:annotation> <xs:documentation>Specifies the person family and given name.</xs:documentation> </xs:annotation> <xs:complexType> <xs:all> <xs:element ref="family"/> <xs:element ref="given"/> </xs:all> </xs:complexType> </xs:element></pre>	

Figure 225: Documentation for a Schema Component

If the schema contains imported or included modules, their dependencies tree is generated in the documentation.

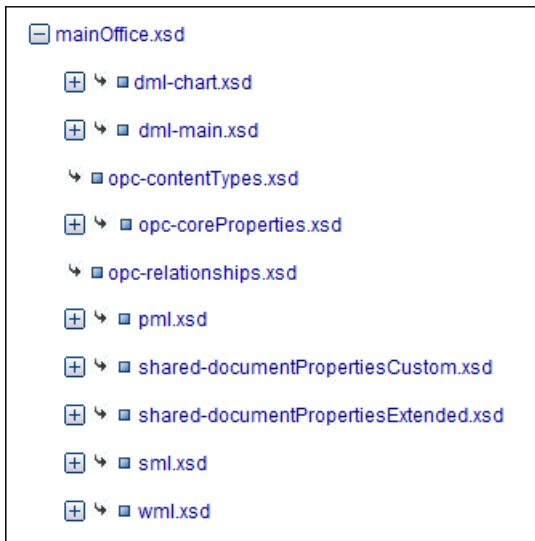


Figure 226: Example: Generated Documentation

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 the [Customizing the PDF Output of Generated XML Schema Documentation](#) on page 426 topic.

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 transformation scenario (such as, *DocBook PDF* or *DocBook HTML*) on the output file, or configure your own transformation scenario for it 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, and DocBook 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.

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 the 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:

http://www.oxygenxml.com/demo/DocBook_Editing_in_Author.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. Enable 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 enable 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 `endor sed` inside the `[OXYGEN_INSTALL_DIR]/lib` directory and place the created JAR or ZIP file inside it.
3. Restart Oxygen XML Developer 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 Developer 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 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 Developer generates an approximation of the source schema. Oxygen XML Developer 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.

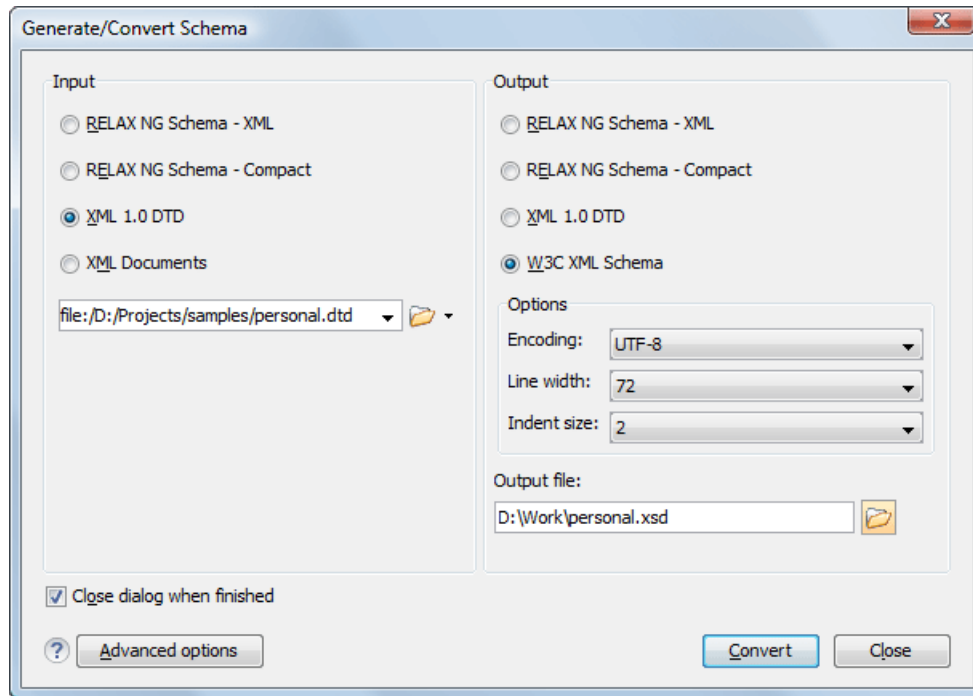


Figure 227: Generate/Convert Schema Dialog Box

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 This section is available if you select **W3C XML Schema** for the output.

Schema Output section

- **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 Developer 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.

- Select the database column you want to configure.
- 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.
- 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 Developer 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.

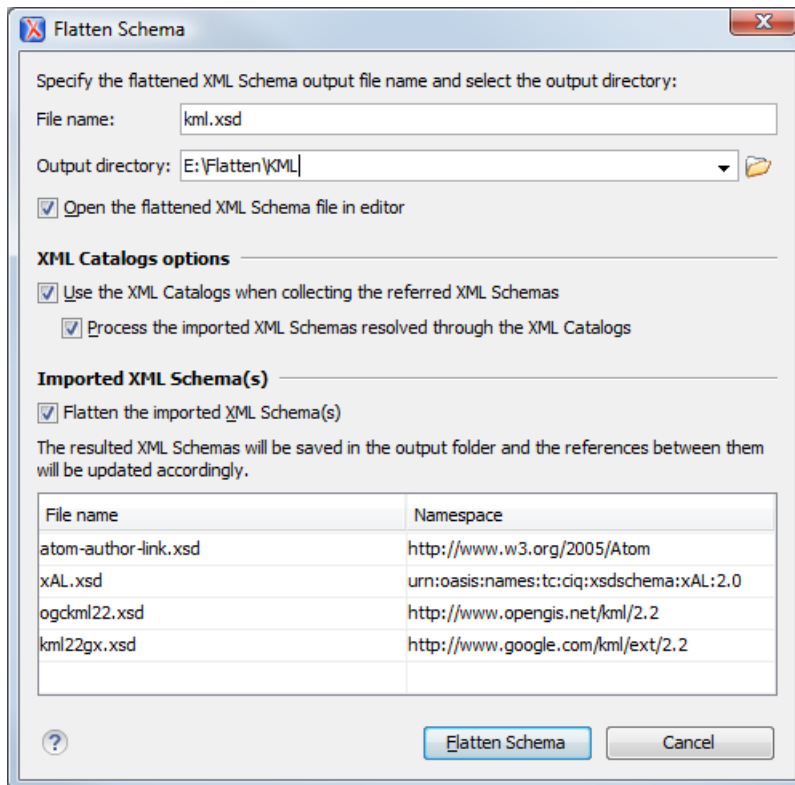


Figure 228: Flatten Schema Dialog Box

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 Developer 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.

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
```

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
```

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.

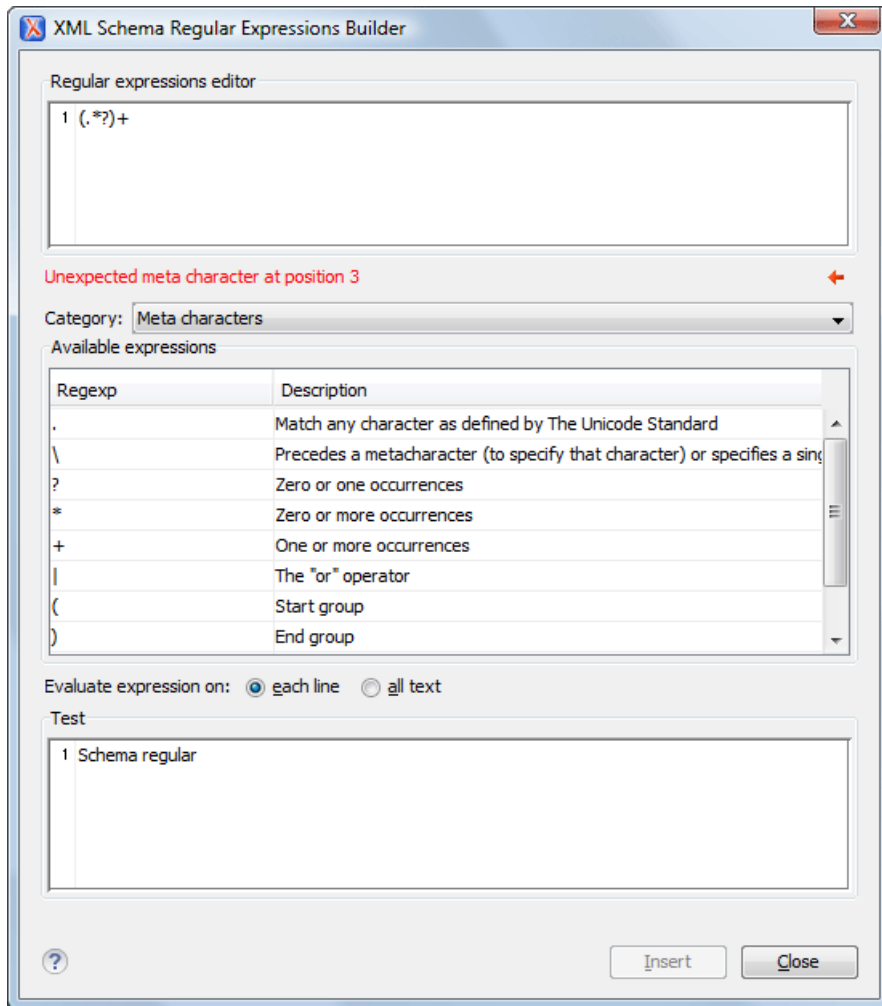


Figure 229: XML Schema Regular Expressions Builder Dialog Box

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 disabled 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 Developer 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.

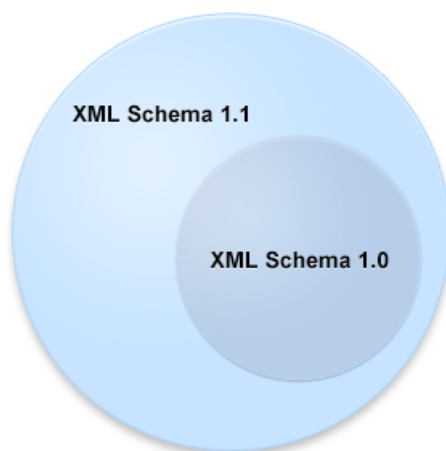


Figure 230: XML Schema 1.1

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 http://oxygenxml.com/demo/XML_Schema_11.html.

Related information

[Setting the XML Schema Version](#) on page 436

Setting the XML Schema Version

Oxygen XML Developer 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 6: Using the `minVersion` and `maxVersion` Attributes to Set the XML Schema Version

Versioning Attributes	XML Schema Version
<code>minVersion = "1.0" maxVersion = "1.1"</code>	1.0
<code>minVersion = "1.1"</code>	1.1
<code>minVersion = "1.0" maxVersion = greater than "1.1"</code>	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 Developer to its default options.

Oxygen XML Developer 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 Developer takes into account the `minVersion` and `maxVersion` attributes defined in the XML Schema.

Related information

[XML Schema 1.1](#) on page 435

Editing XQuery Documents

This section explains the features of the XQuery editor and how to use them.

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 from the **Window > Show View** menu.

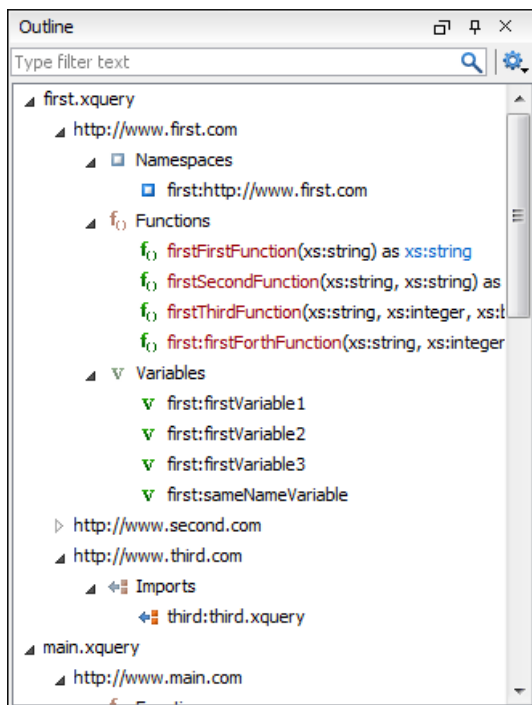


Figure 231: XQuery Outline View

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:

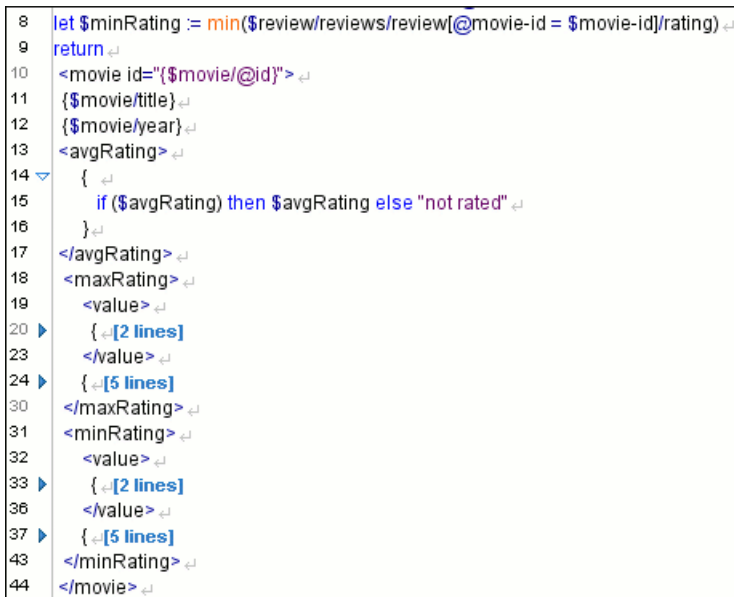
- * - any string
- ? - any character
- , - patterns separator

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.

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] }
21     </value>
22   </maxRating>
23   <minRating>
24     <value>
25       { [5 lines] }
26     </value>
27   </minRating>
28 </movie>

```

Figure 232: 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.

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 Developer 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.

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.

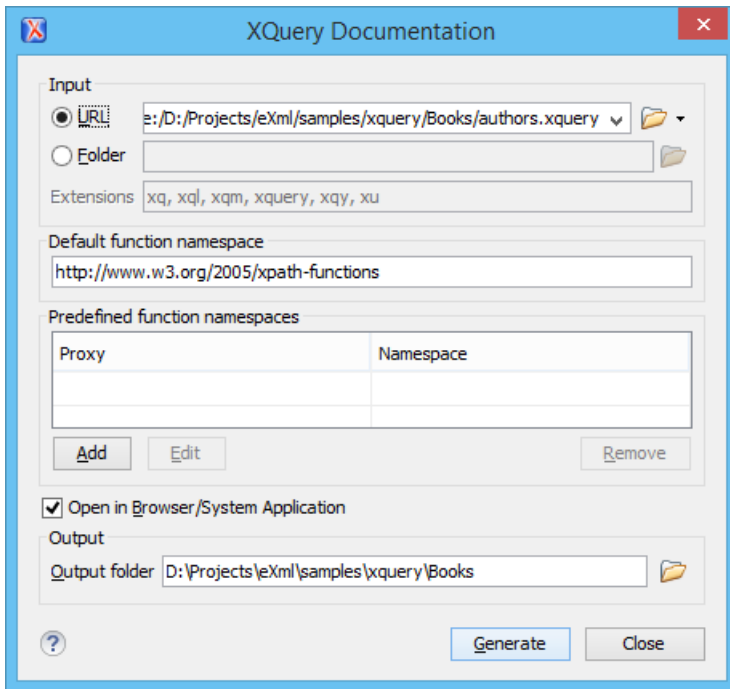


Figure 233: XQuery Documentation Dialog Box

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.

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 Developer 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 Developer 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.

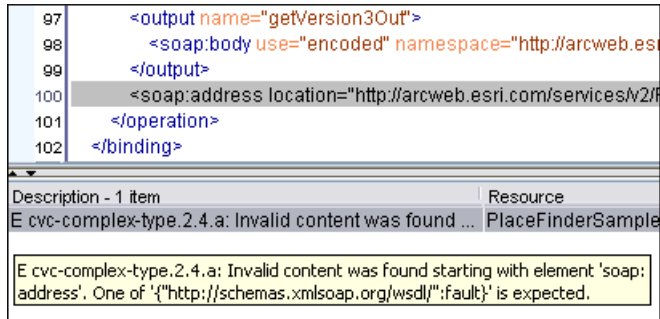


Figure 234: Validating a WSDL file

To watch our video demonstration about the WSDL editing support in Oxygen XML Developer, go to http://www.oxygenxml.com/demo/Create_New_WSDL.html.

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 Developer 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 can set a main WSDL 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 modules. Oxygen XML Developer 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** 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 http://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 disable 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. 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.

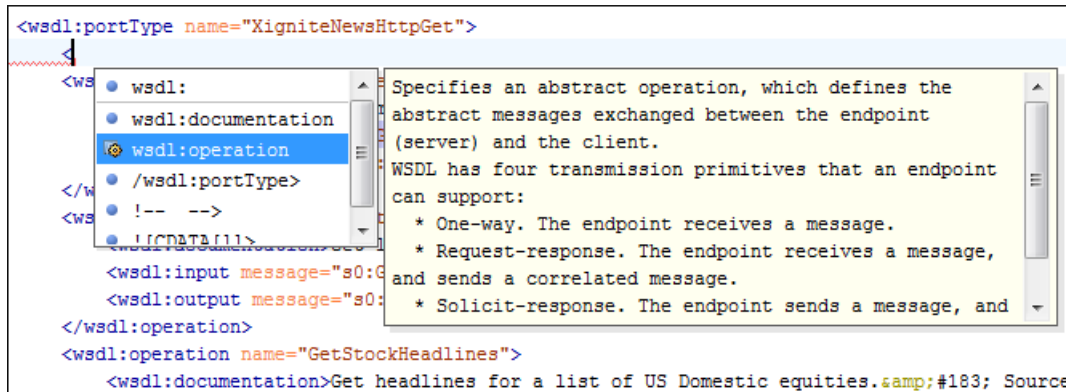


Figure 235: WSDL Content Completion Assistant

 **Note:** 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](#).

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 236: 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 Developer provides an easy mode to declare them by proposing a prefix for these namespaces.

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 from the **Window > Show View** menu.

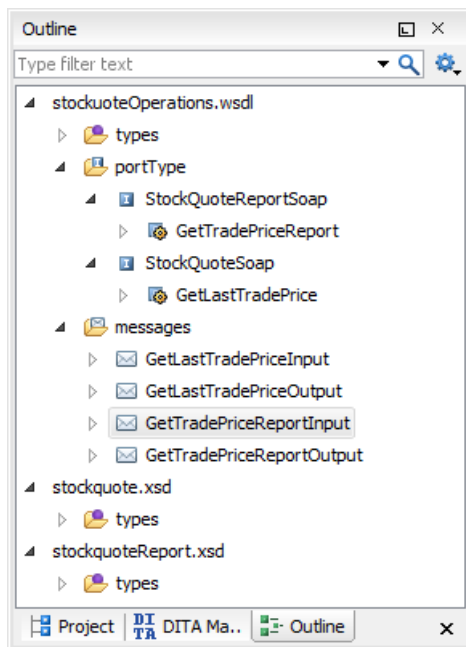


Figure 237: WSDL Outline View

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	Displays the dependencies of the currently selected component.
Resource Hierarchy	Displays the hierarchy for the currently selected resource.
Resource Dependencies	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**

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 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.

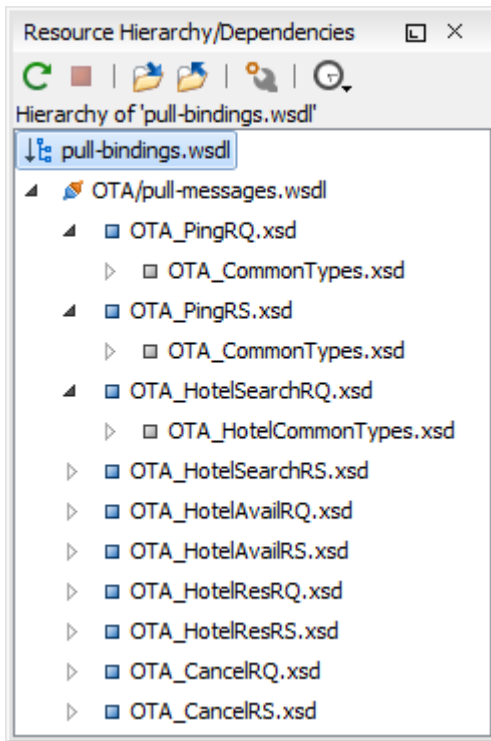


Figure 238: Resource Hierarchy/Dependencies View

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.

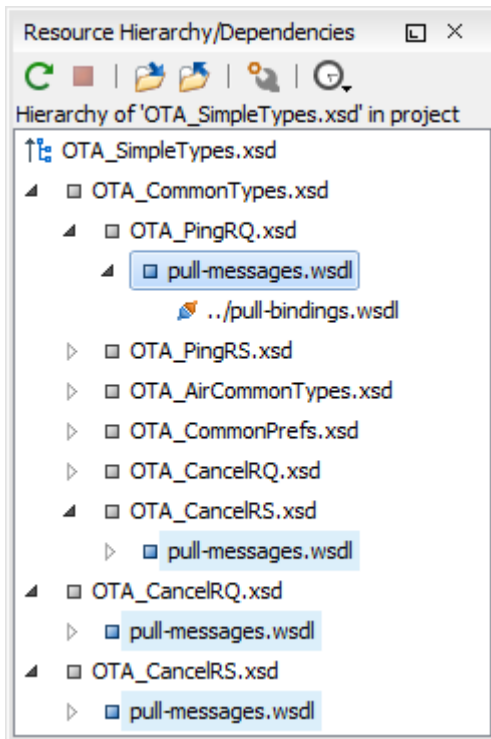


Figure 239: Resource Hierarchy/Dependencies View

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 308

[Search and Refactor Operations Scope](#) on page 303

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** - Enable 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)** - Enable 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 enabled, 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 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.

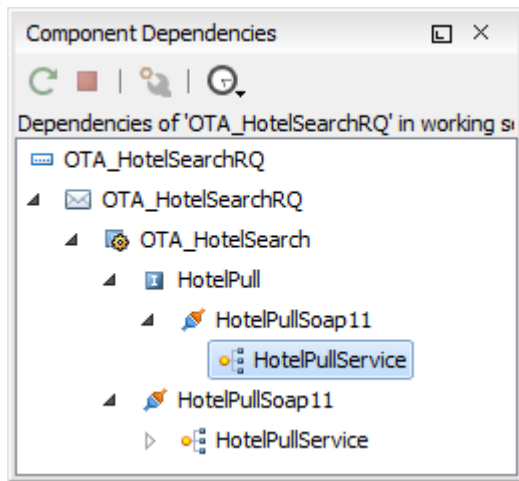


Figure 240: Component Dependencies View

The following actions 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 checking the corresponding check box.



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 Developer 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.

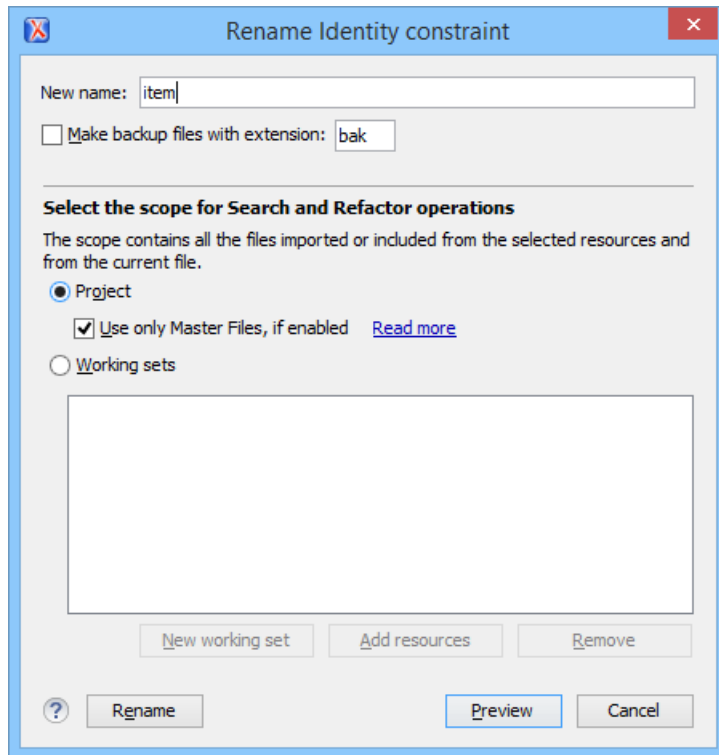


Figure 241: Rename Identity Constraint Dialog Box

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](#).

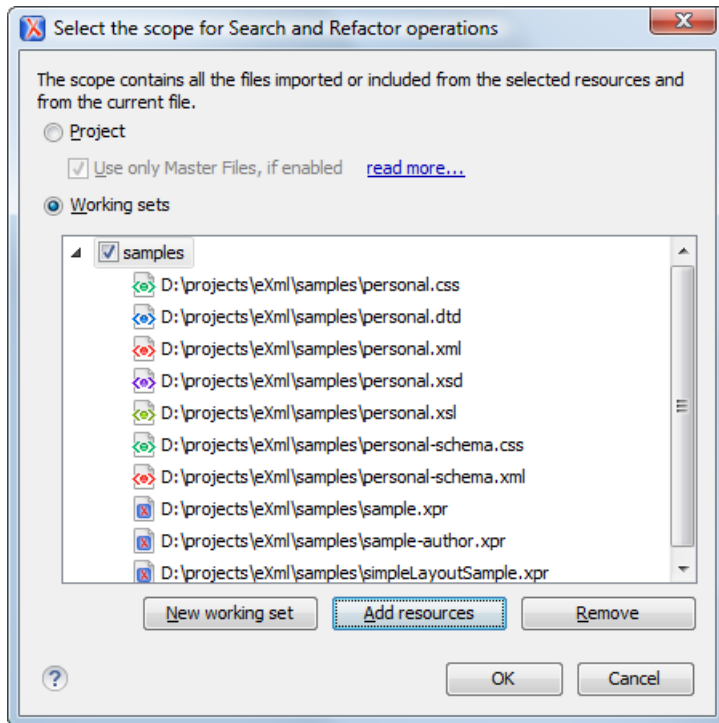


Figure 242: Change Scope Dialog Box

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.

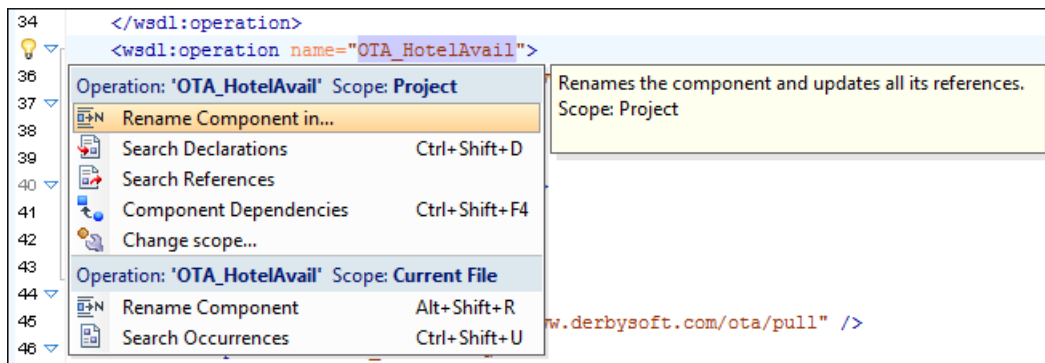


Figure 243: WSDL Quick Assist Support

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 Developer 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.

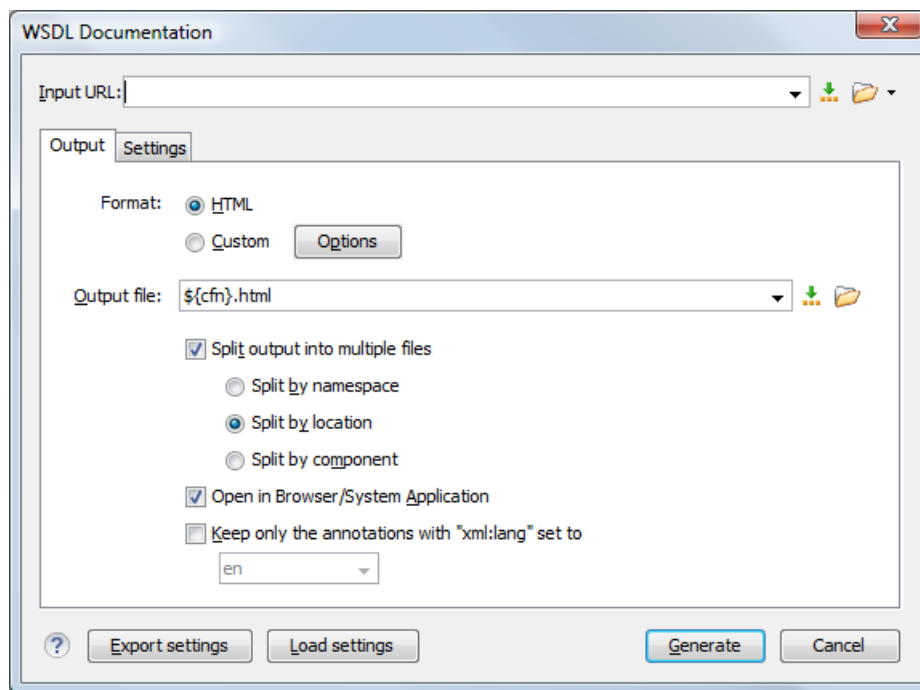


Figure 244: WSDL Documentation Dialog Box

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.

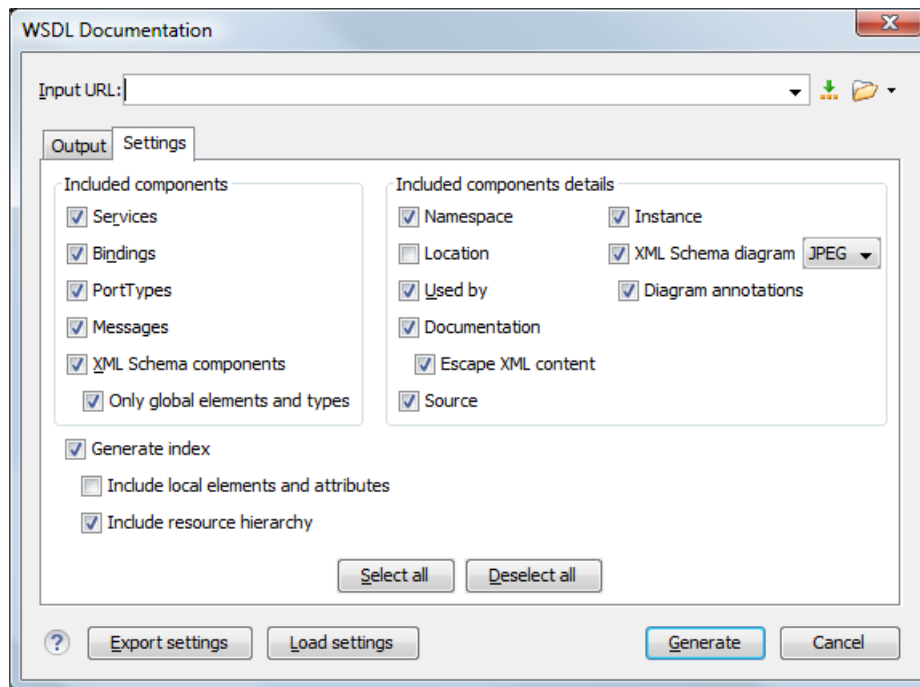


Figure 245: Settings Tab of the WSDL Documentation Dialog Box

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 checked, 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 disabled 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](#).)

Load 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.

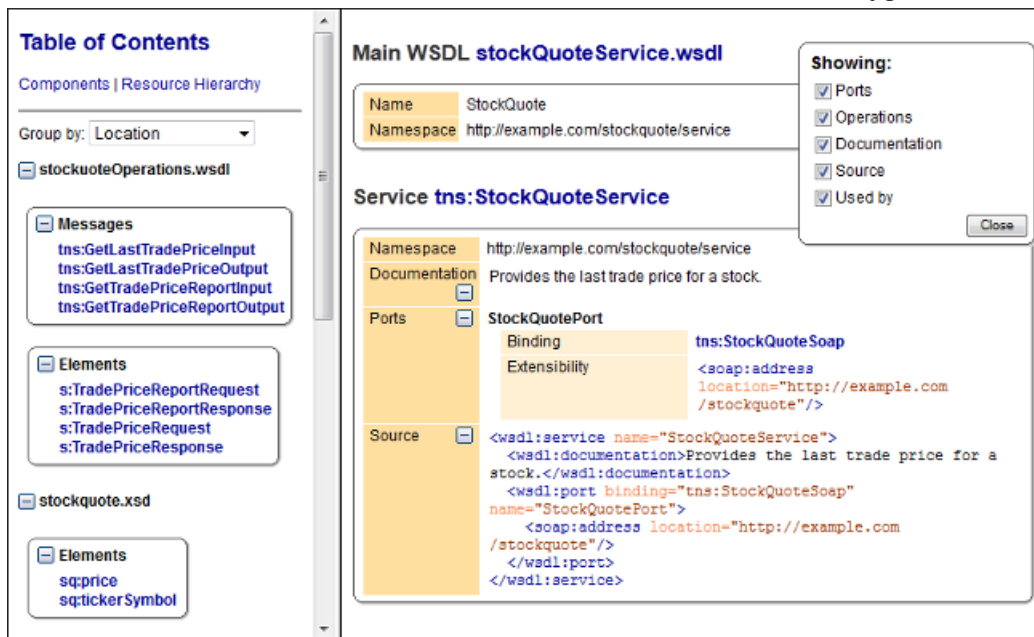


Figure 246: WSDL Documentation in HTML Format

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.

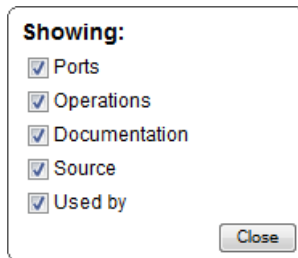


Figure 247: Showing Options

Generating WSDL Documentation in a Custom Format

To obtain the default HTML documentation output from a WSDL document, Oxygen XML Developer 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.

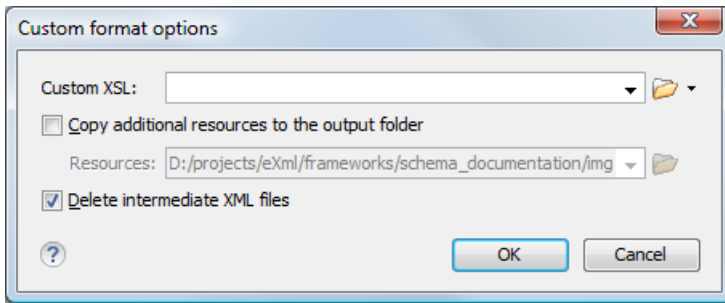


Figure 248: Custom Format Options Dialog Box

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 Developer. You can integrate the scripts in an external batch process launched from the command-line interface.

WSDL SOAP Analyzer

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).

Composing a SOAP Request

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.

Oxygen XML Developer 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.

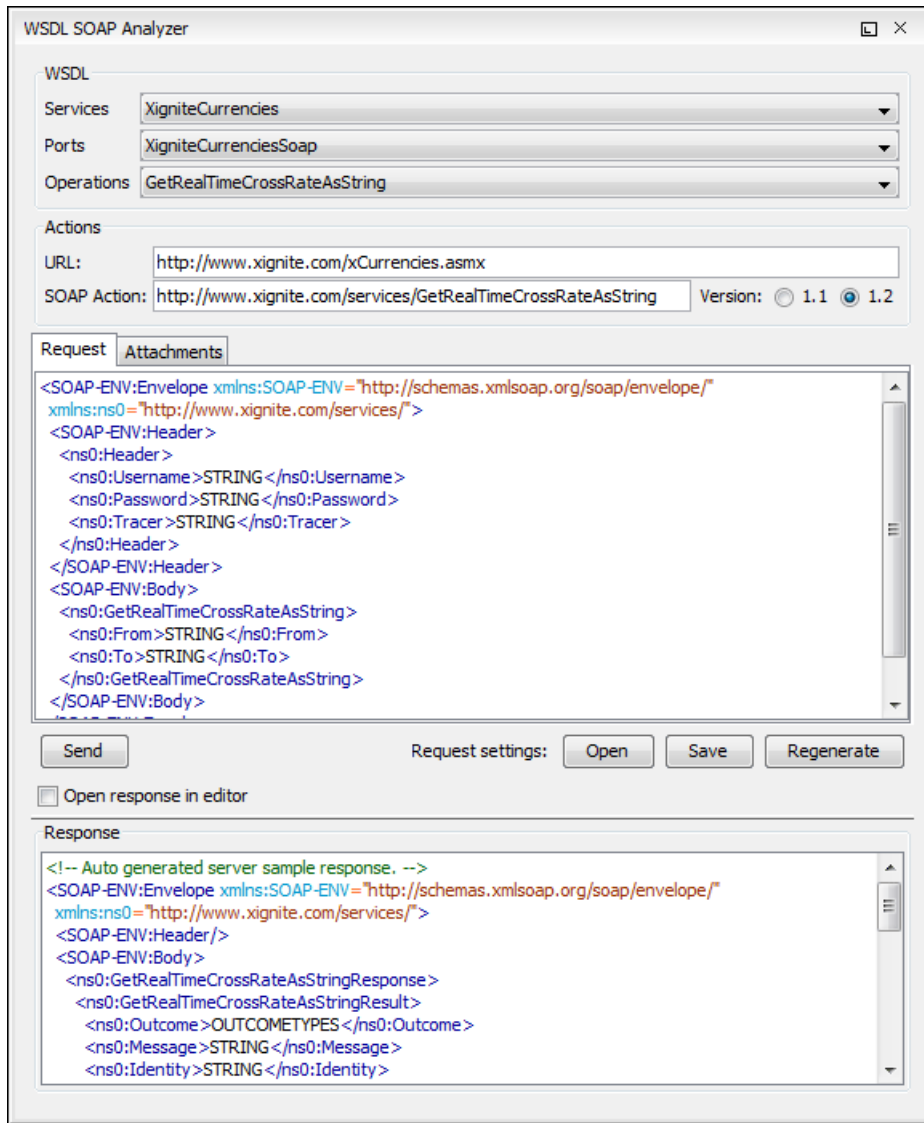


Figure 249: WSDL SOAP Analyzer Dialog Box

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 Developer 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 Developer 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 Developer 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 Developer 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.

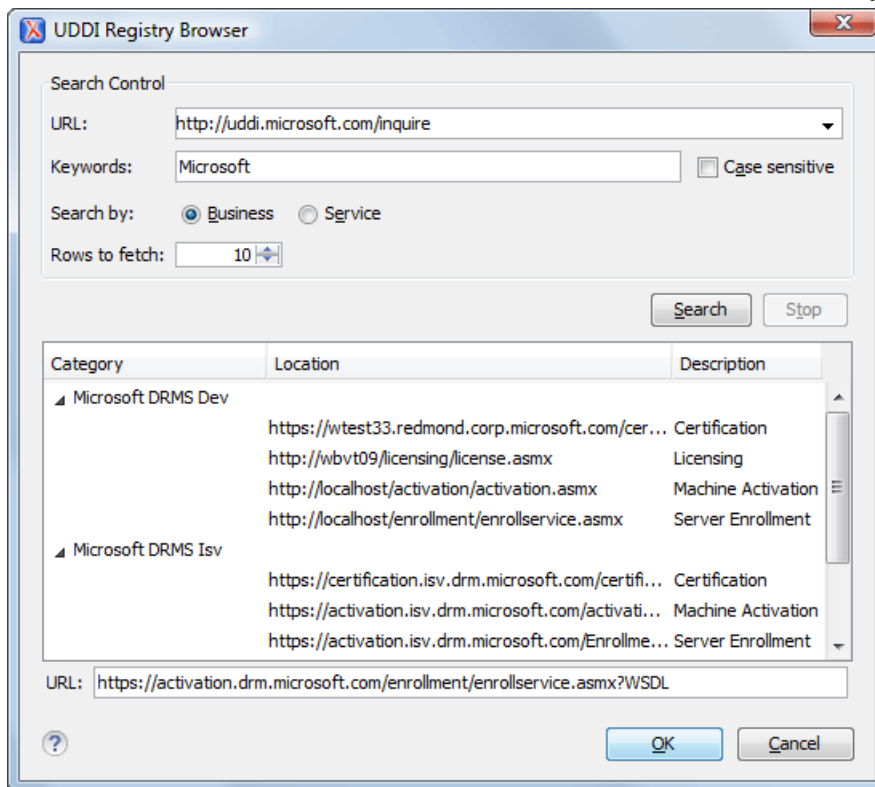


Figure 250: UDDI Registry Browser Dialog Box

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 checked, 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 Developer 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 Developer 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 Developer 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 and the CSS extensions specific for Oxygen. 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

Lists the steps required for specifying 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 is 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.

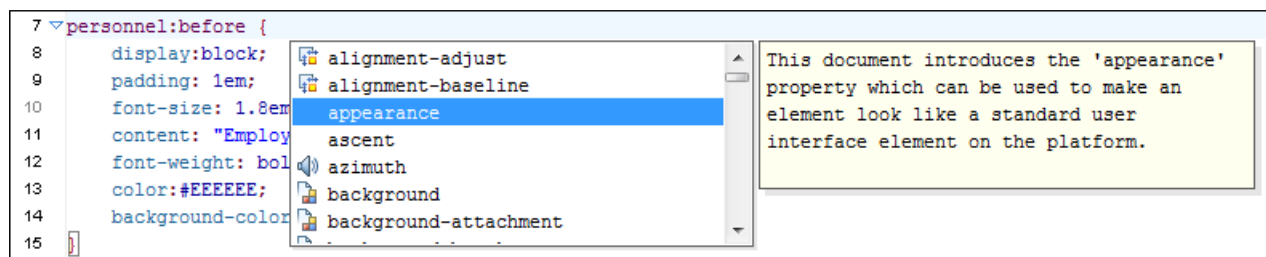


Figure 251: Content Completion in CSS Stylesheets

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.

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 tasks

[Specify Custom CSS Properties](#) on page 460

Lists the steps required for specifying custom CSS properties.

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 Developer 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 from the **Window > Show View** menu.

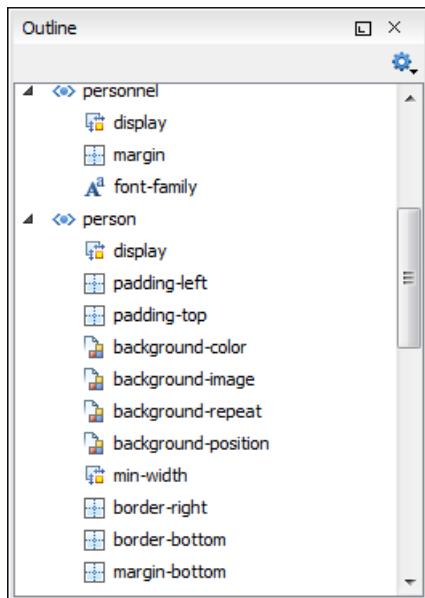


Figure 252: CSS Outline View

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 Developer 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 Developer 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 Developer 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 Developer 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 Developer 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 Developer 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 Developer 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 is 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.

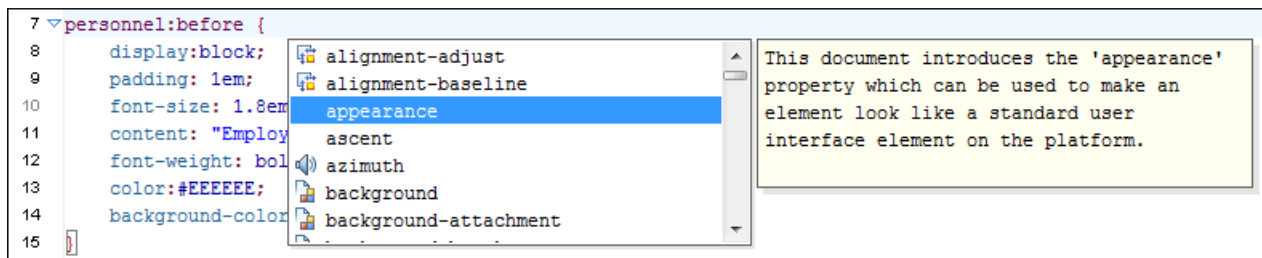


Figure 253: Content Completion in LESS Stylesheets

The properties and values available are dependent on the CSS Profile selected in the [CSS preferences](#).

Compiling LESS Stylesheets to CSS

When editing LESS files, you can compile the files into CSS. Oxygen XML Developer 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. Enable 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 enabled, when you save a LESS file it will automatically be compiled to CSS (this option is disabled by default).

 **Important:** If this option is enabled, 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 Developer 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.

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 Developer 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 Relax NG 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 Developer 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 includes components defined in modules other than the currently edited one.
- The **Outline** displays the components collected from the entire schema structure.

Related tasks

[Creating a New Validation Scenario](#) on page 287

Related information

[XML Schema Outline View](#) on page 198

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 Developer 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 Developer 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**.

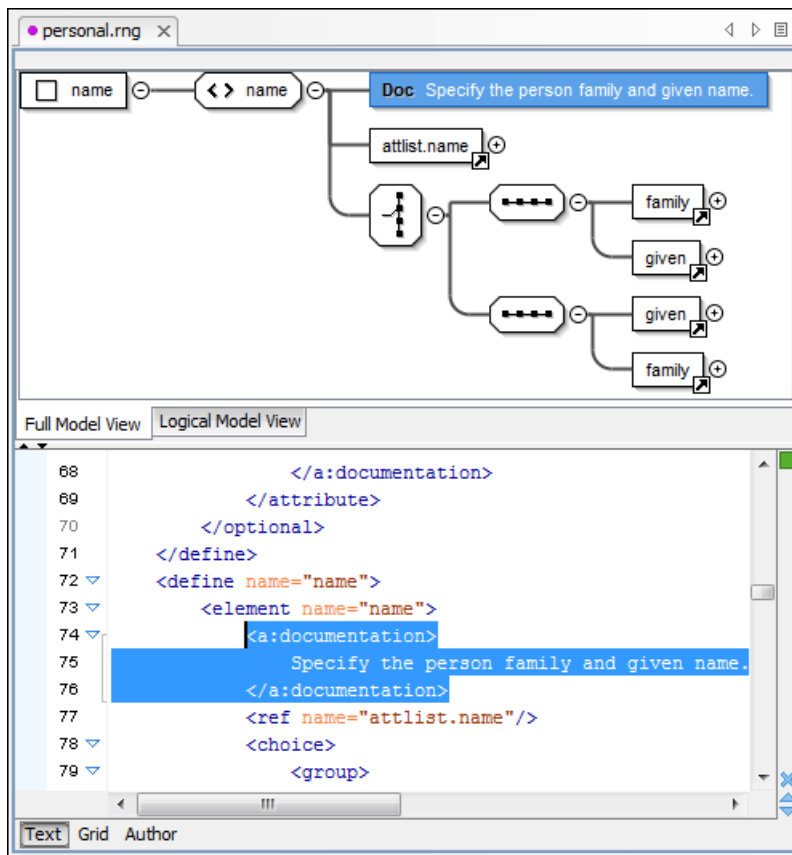


Figure 254: Relax NG Schema Editor - Full Model View

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.

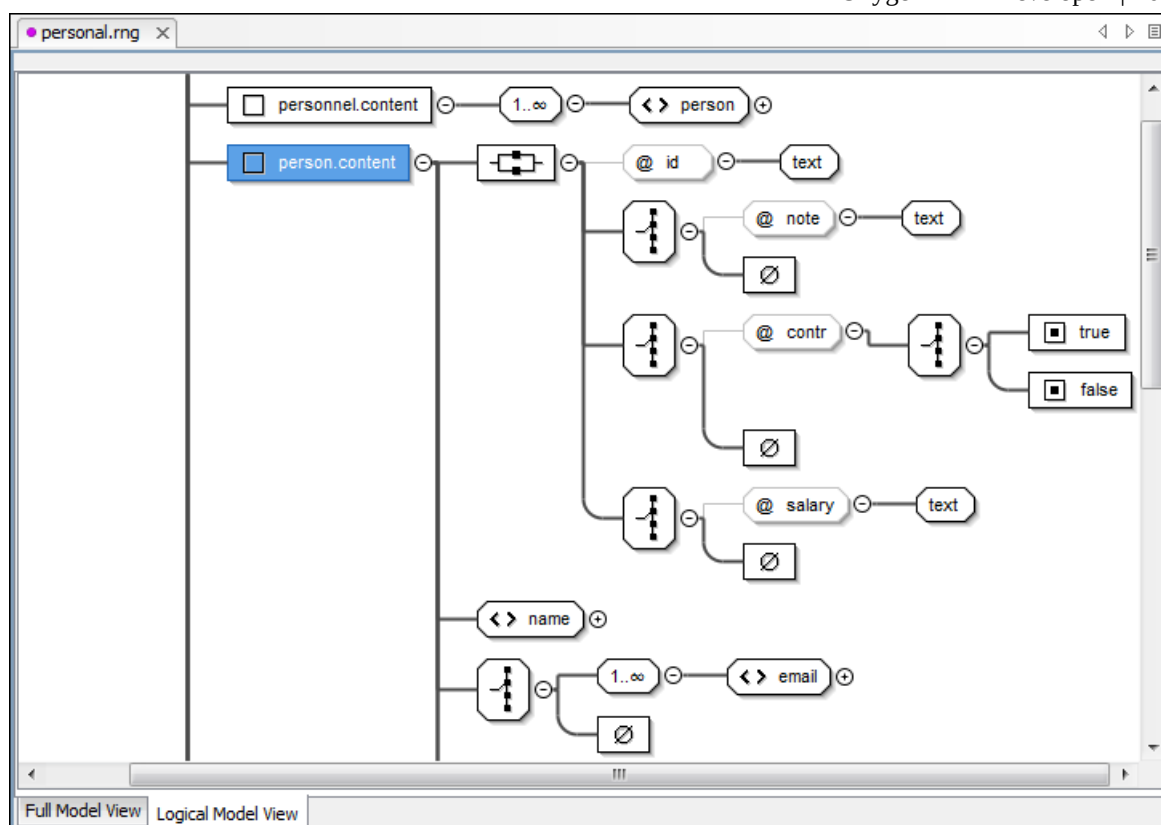
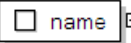
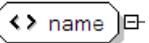
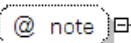
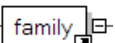
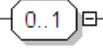
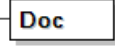


Figure 255: Logical Model View for a Relax NG Schema

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 the 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 disable 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 Developer 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 282

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 from the **Window** > **Show View** menu.

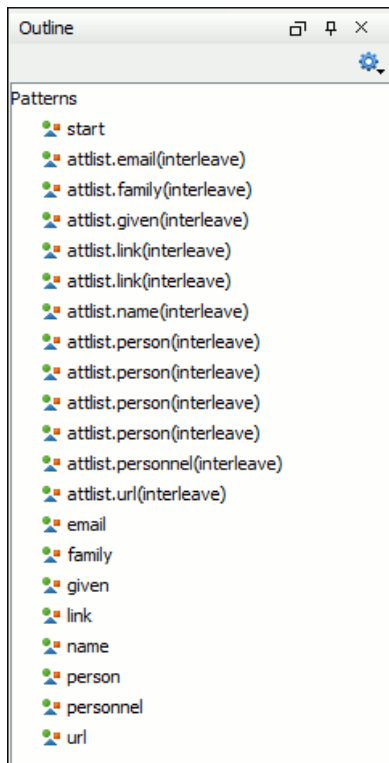


Figure 256: Relax NG Outline View

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	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 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.

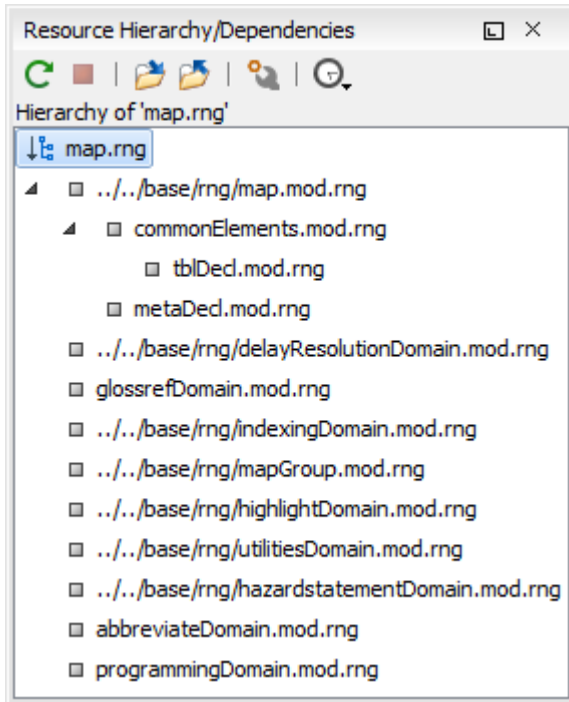


Figure 257: Resource Hierarchy/Dependencies View - hierarchy for map.rng

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.

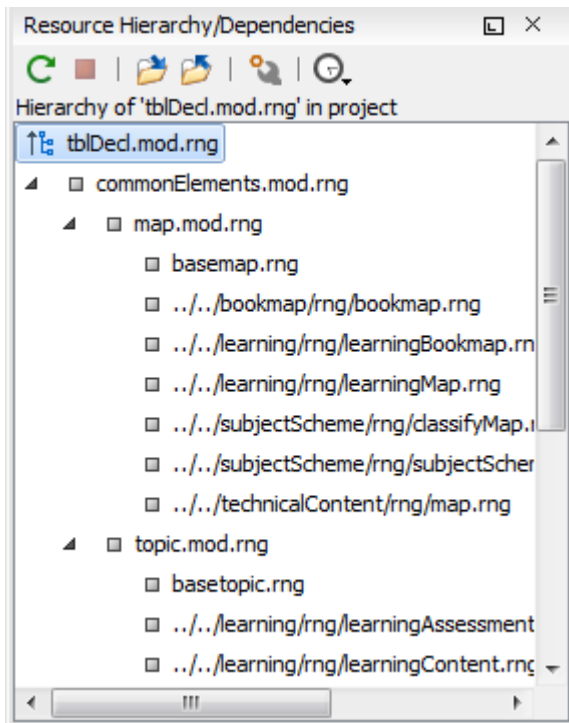


Figure 258: Resource Hierarchy/Dependencies View - Dependencies for tblDed.mod.rng

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 308

[Search and Refactor Operations Scope](#) on page 303

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** - Enable 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)** - Enable 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 enabled, 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 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.

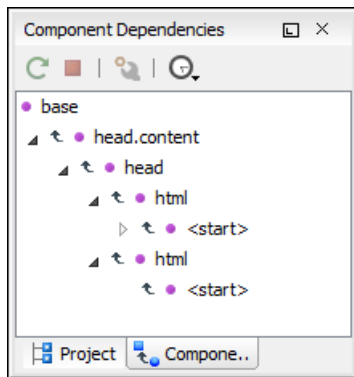


Figure 259: Component Dependencies View - Hierarchy for base.rng

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 checking 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 303

Search and Refactor Operations Scope

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.

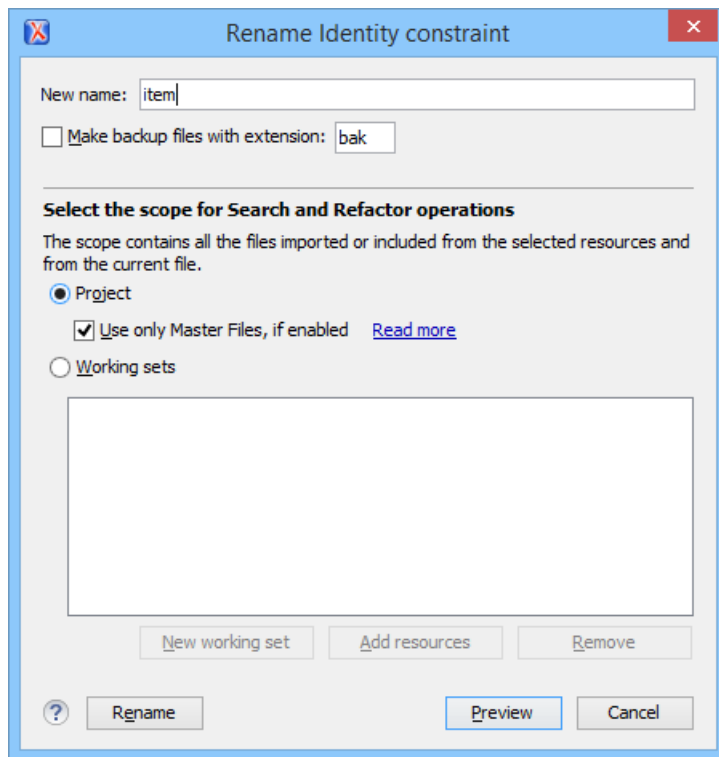


Figure 260: Rename Identity Constraint Dialog Box

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.

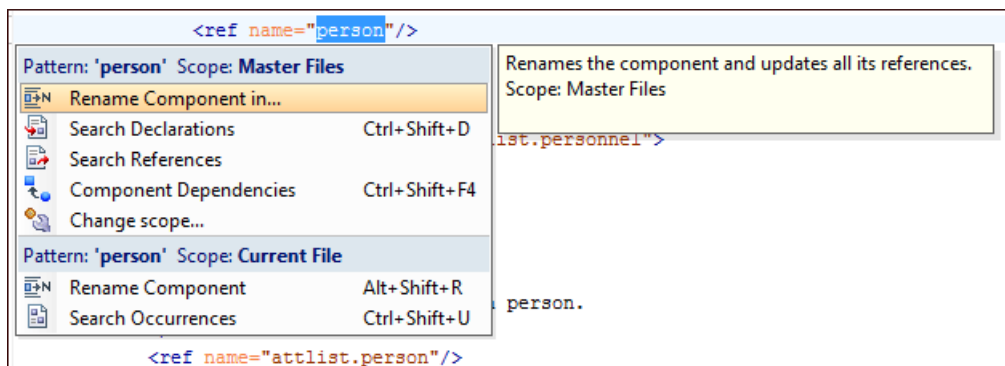


Figure 261: RNG Quick Assist Support

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 472

[Resource Hierarchy/Dependencies View](#) on page 469

[Searching and Refactoring Actions](#) on page 473

[Search and Refactor Operations Scope](#) on page 303

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 Developer.

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 Developer 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.

NVDL Schema Diagram

This section explains how to use the graphical diagram of a NVDL schema.

Introduction to NVDL Schema Diagram Editor

Oxygen XML Developer 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 Developer 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**.

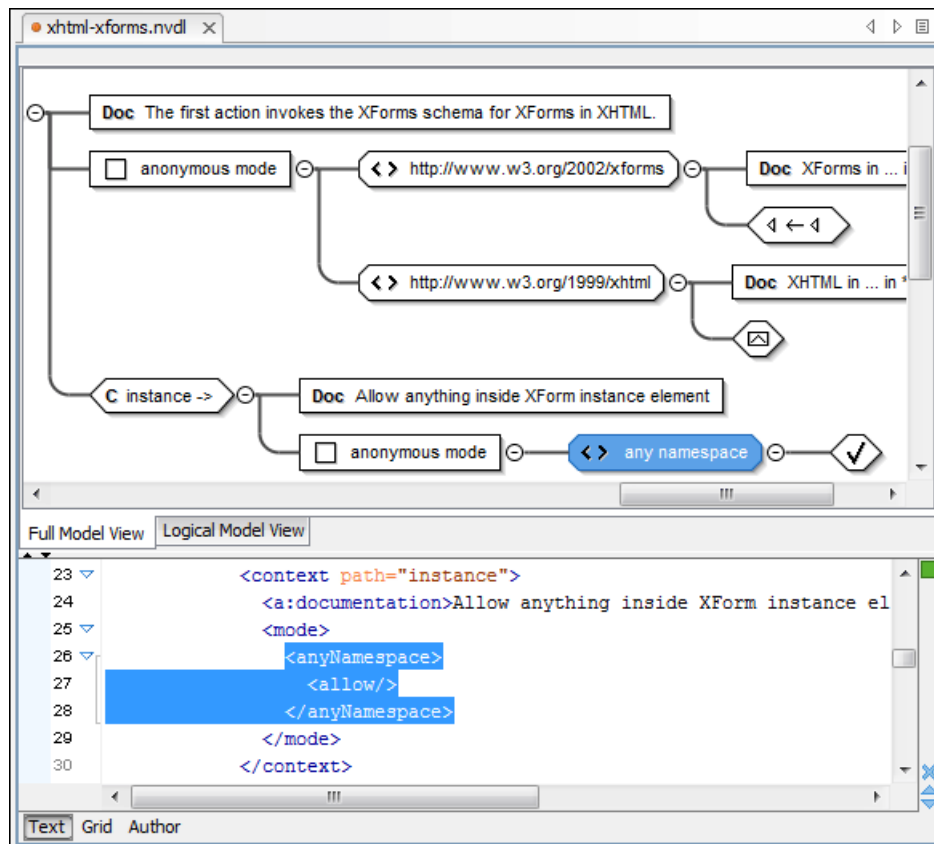


Figure 262: NVDL Schema Editor - Full Model View

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.

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.

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 from the **Window > Show View** menu.

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 disable 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 Developer 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 282

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 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.

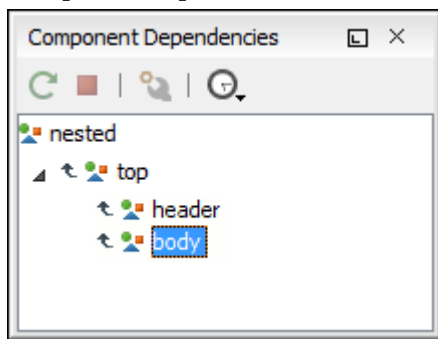


Figure 263: Component Dependencies View - Hierarchy for test.nvdl

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.

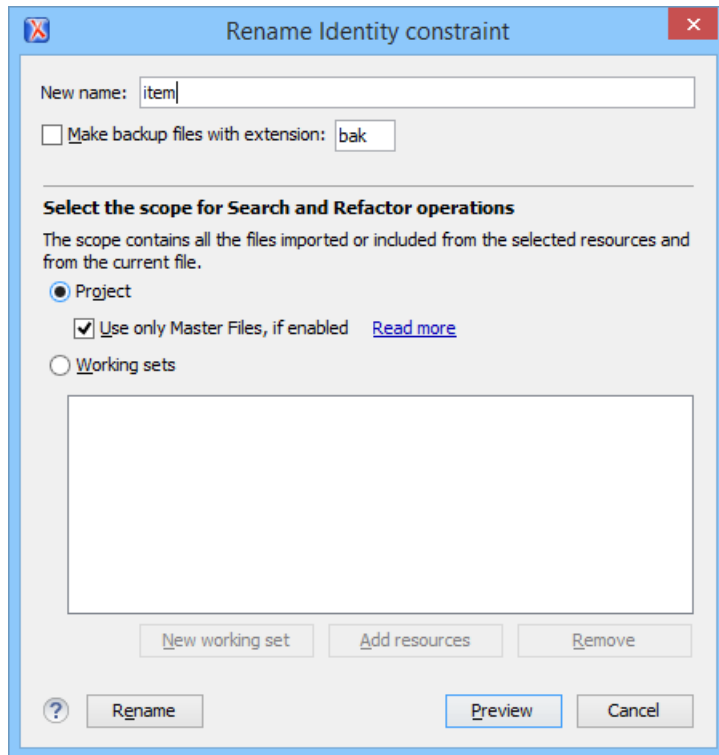


Figure 264: Rename Identity Constraint Dialog Box

Editing JSON Documents

This section explains the features of the Oxygen XML Developer 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.

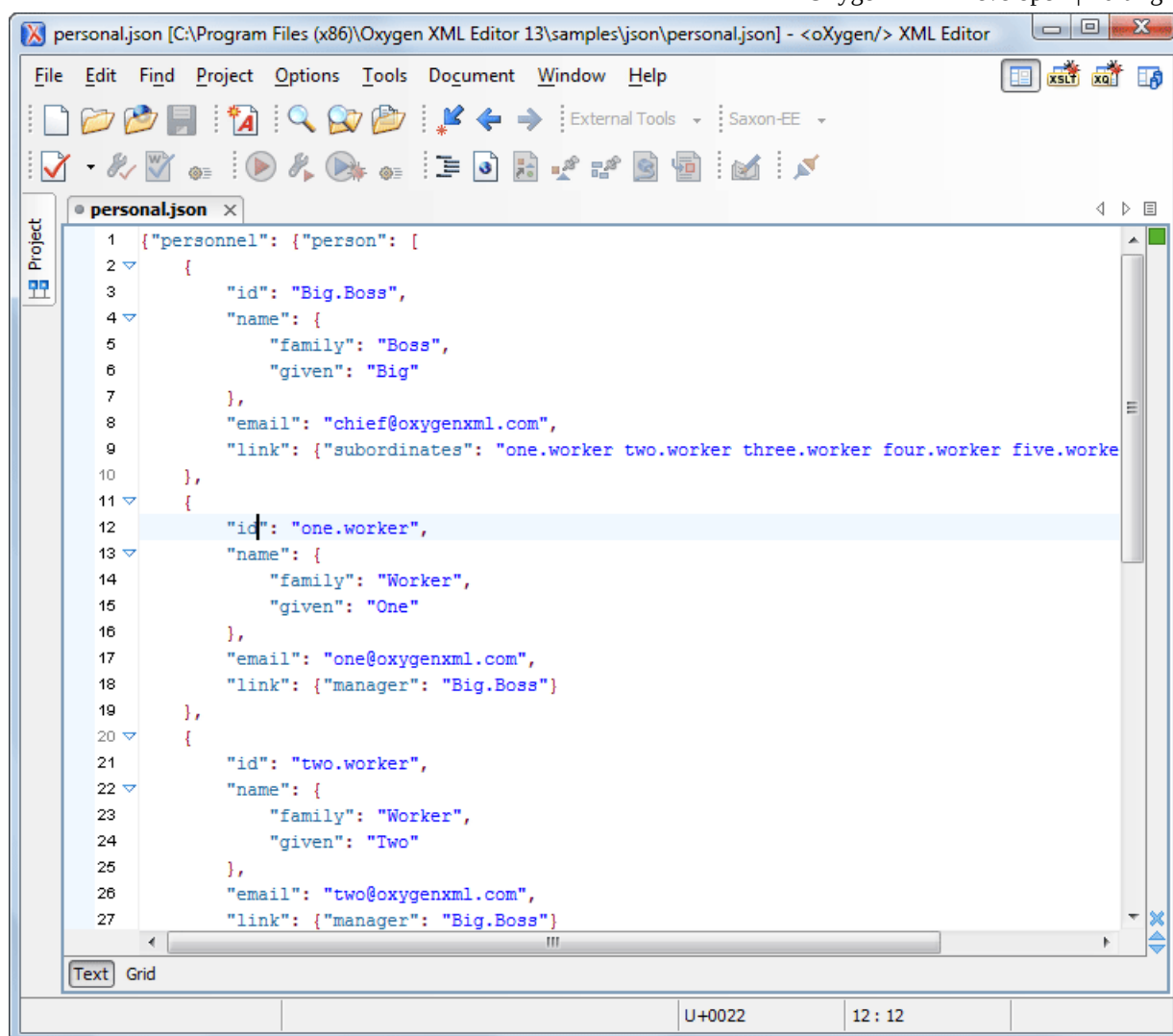


Figure 265: JSON Editor Text Mode

Related information

[Text Preferences](#) on page 76

[Editing JSON Documents in Grid Mode](#) on page 481

Syntax Highlight in JSON Documents

Oxygen XML Developer supports [Syntax Highlight](#) for JavaScript / JSON editors and provides default configurations for the JSON set of tokens. You can customize the foreground color, background color and the font style for each JSON token type.

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.

Editing JSON Documents in Grid Mode

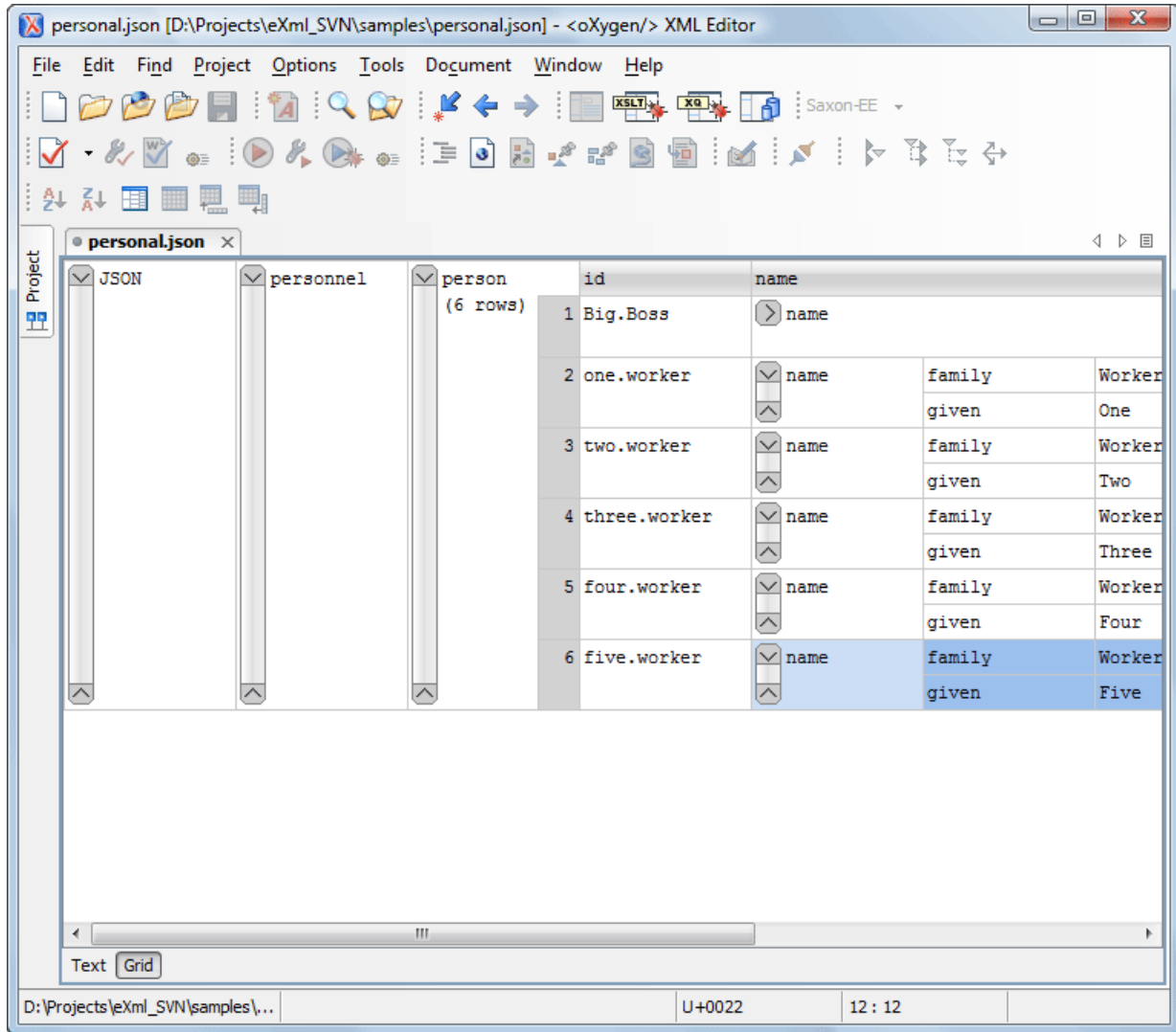


Figure 266: JSON Editor Grid Mode

Oxygen XML Developer 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 <i>value</i> to <i>array</i> .
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 192

[Grid Preferences](#) on page 77

Validating JSON Documents

Oxygen XML Developer 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 191

[Document Checking Preferences](#) on page 96

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 from the **Window > Show View** menu.

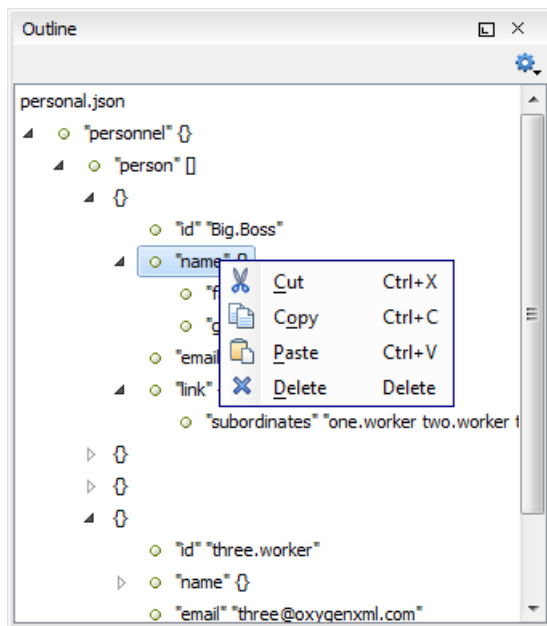


Figure 267: JSON Outline View

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 allows you to enable **Selection update on cursor move**. This option controls the synchronization between the **Outline** view and source the document. Oxygen XML Developer 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 Developer 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:

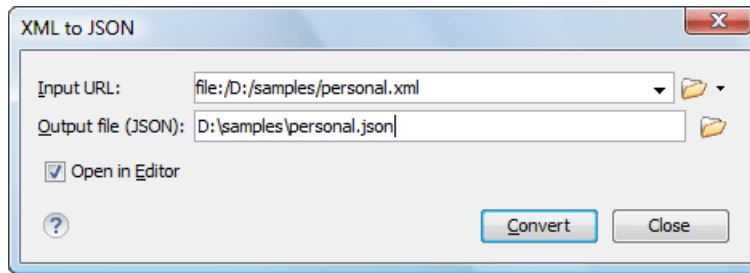


Figure 268: XML to JSON Dialog Box

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. Check the **Open in Editor** option to open the JSON result of the conversion in the Oxygen XML Developer JSON Editor.
5. Click the **Convert** button.

The original XML document is now converted to a JSON document.

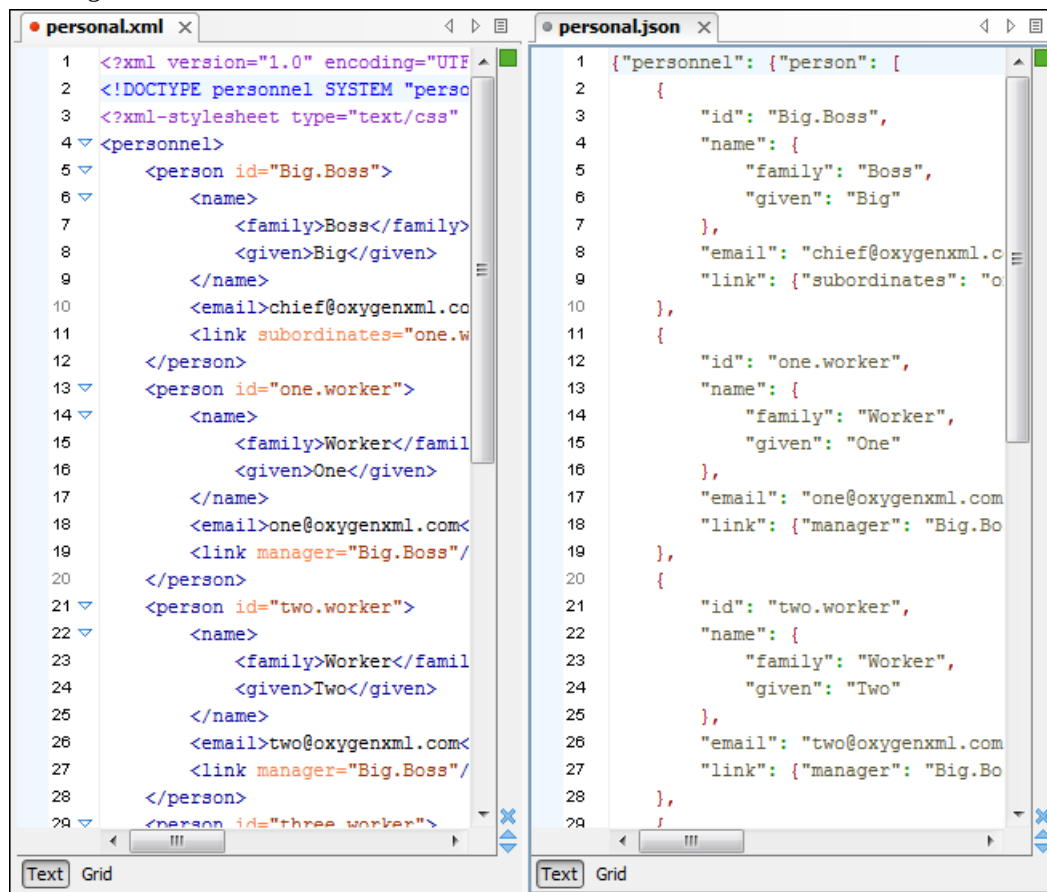


Figure 269: Example: XML to JSON Operation Result

Editing StratML Documents

Strategy Markup Language (StratML) is an XML vocabulary and schema for strategic plans. Oxygen XML Developer 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 Developer 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 Developer 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 Developer JavaScript Editor and how you can use them.

JavaScript Editor Text Mode

Oxygen XML Developer 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.


```

12 function newPage(filename, overlay) {
13     divs = document.getElementsByTagName("div");
14
15     if (divs) {
16         var xdiv = divs[0];
17
18         if (xdiv) {
19             var xid = xdiv.getAttribute("id");
20
21             var mytoc = window.top.frames[0];
22             if (mytoc.lastUnderlined) {
23                 mytoc.lastUnderlined.style.textDecoration = "none";
24             }
25
26             var tdiv = xbGetElementById(xid, mytoc);
27
28             if (tdiv) {
29                 var ta = tdiv.getElementsByTagName("a").item(0);
30                 ta.style.textDecoration = "underline";
31                 mytoc.lastUnderlined = ta;
32             }
33         }
34     }
35
36     if (overlay != 0) {
37         overlaySetup('lc');
38     }

```

Figure 270: 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 Developer 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 stripe on the left side of the editor, the following actions are available: <table border="0" data-bbox="467 688 1476 1010"> <tr> <td> Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on OS X))</td><td>Folds all the elements except the current element.</td></tr> <tr> <td> Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on OS X))</td><td>Folds the elements indented with one level inside the current element.</td></tr> <tr> <td> Expand Child Folds</td><td>Unfolds all child elements of the currently selected element.</td></tr> <tr> <td> Expand All (Ctrl + NumPad* (Command + NumPad* on OS X))</td><td>Unfolds all elements in the current document.</td></tr> </table>	 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.
 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 Developer 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 Developer can validate a JavaScript document both on-request and automatically.

Content Completion in JavaScript Files

When you edit a JavaScript document, the **Content Completion Assistant** presents you a list of the elements you can insert at the cursor position. 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.

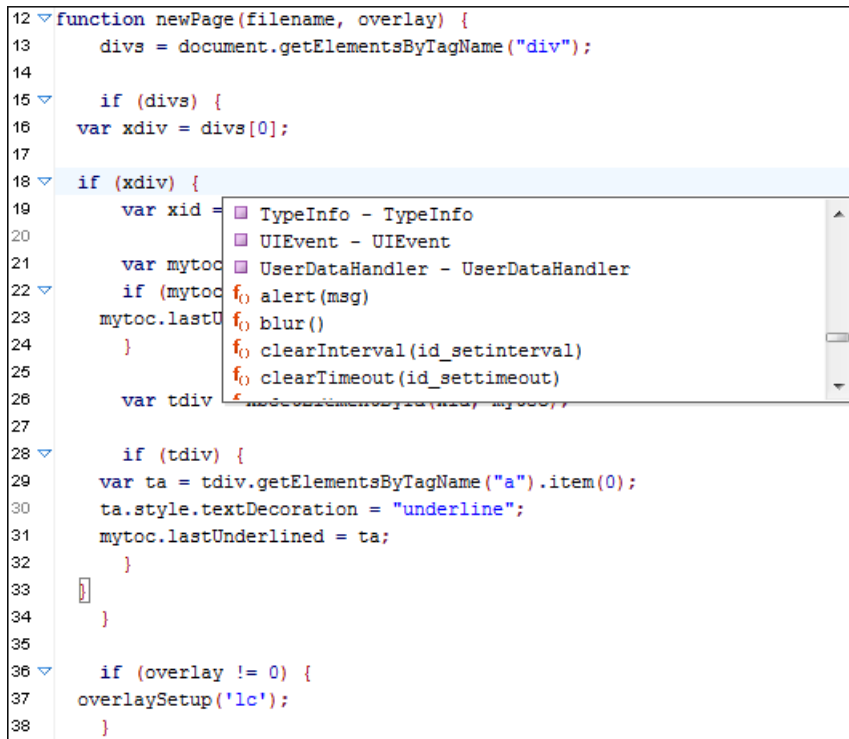


Figure 271: JavaScript Content Completion Assistant

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.

JavaScript Outline View

Oxygen XML Developer 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 from the **Window > Show View** menu.

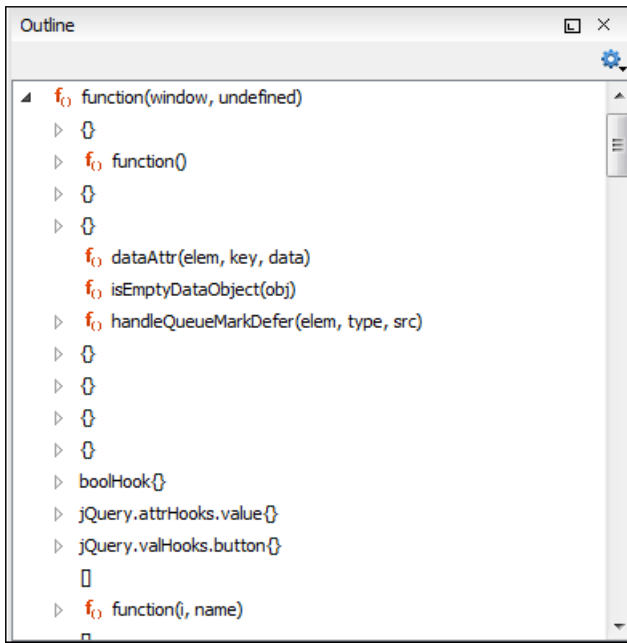


Figure 272: JavaScript Outline View

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 enable 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 Developer version 18.0.

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.

The XProc editor assists you in writing XPath expressions by offering a **Content Completion Assistant** and dedicated coloring schemes. To customize the coloring schemes, *open the Preferences dialog box (Options > Preferences)* and go to **Editor > Syntax Highlight**.

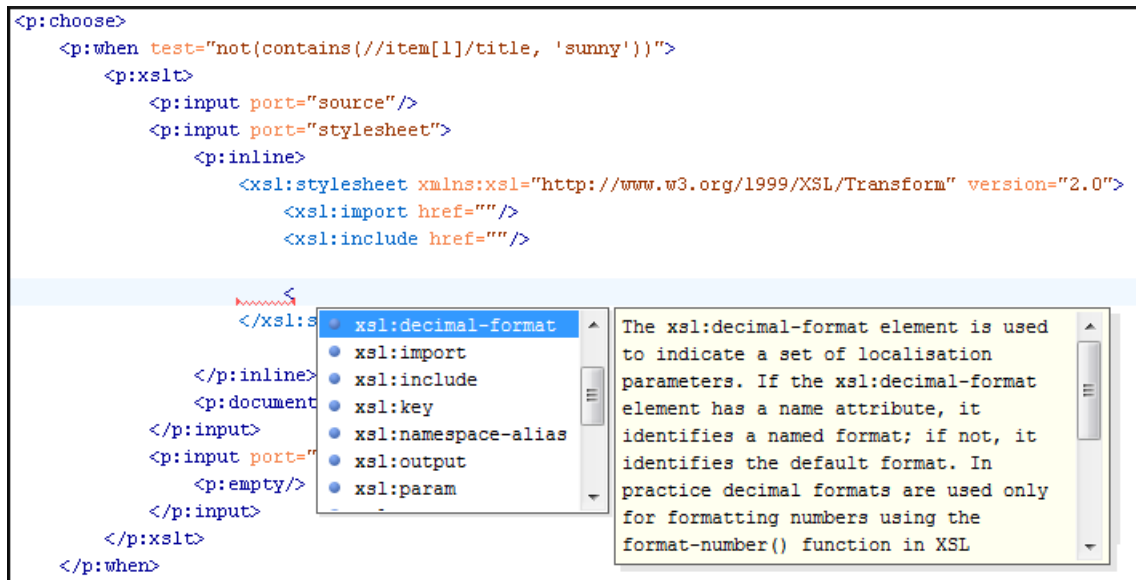


Figure 273: XProc Content Completion

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 Developer 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*.

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 Developer 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 Developer, go to http://www.oxygenxml.com/demo/Schematron_Validation.html, and to http://www.oxygenxml.com/demo/Editing_Schematron_Schemas/Editing_Schematron_Schemas.html.

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 Developer 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 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 Developer 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 includes 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 disable 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 Developer 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 Developer offers an error management mechanism capable of pinpointing errors in XPath expressions and in the included schema modules.

Related information

[Validate an XML Document Against Schematron](#) on page 294

Content Completion in Schematron Documents

Oxygen XML Developer helps you edit a Schematron schema, offering, through the Content Completion Assistant, items that are valid at the cursor position. 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 9.6.0.7 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:

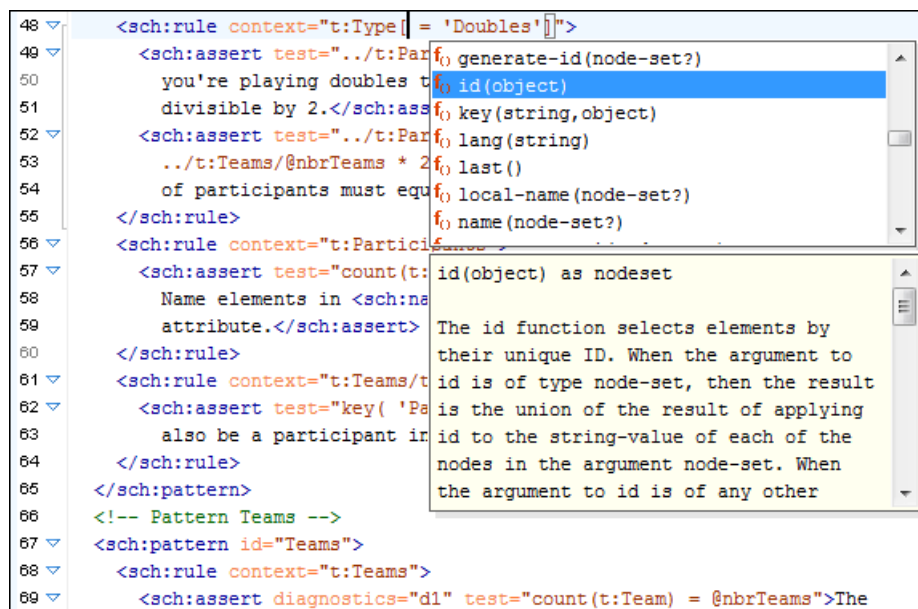


Figure 274: XSLT extension functions in Schematron schemas documents

The **Content Completion Assistant** also includes [code templates that can be used to quickly insert code fragments](#) into Schematron documents.

RELAX NG/XML Schema 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 Developer accepts such documents as Schematron validation schemas and it is able to extract and use the embedded rules. 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. 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 Developer 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.

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 from the **Window > Show View** menu.

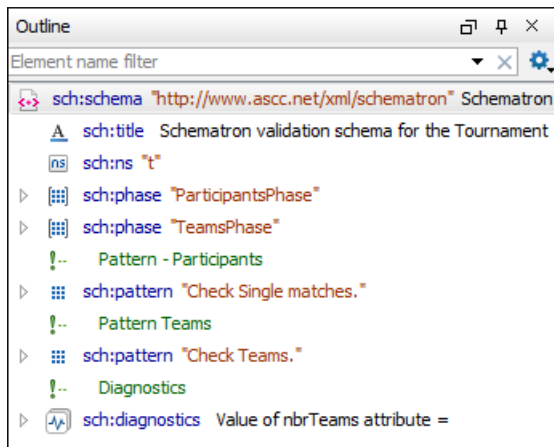


Figure 275: Schematron Outline View

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 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.

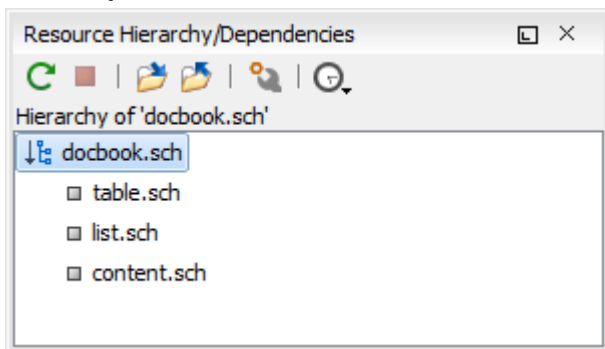


Figure 276: Resource Hierarchy/Dependencies View

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.

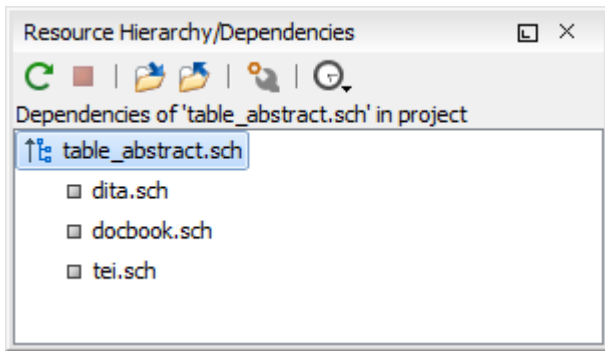


Figure 277: Resource Hierarchy/Dependencies View - Dependencies for table_abstract.sch

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** - Enable 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)** - Enable 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 enabled, 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 Developer 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.

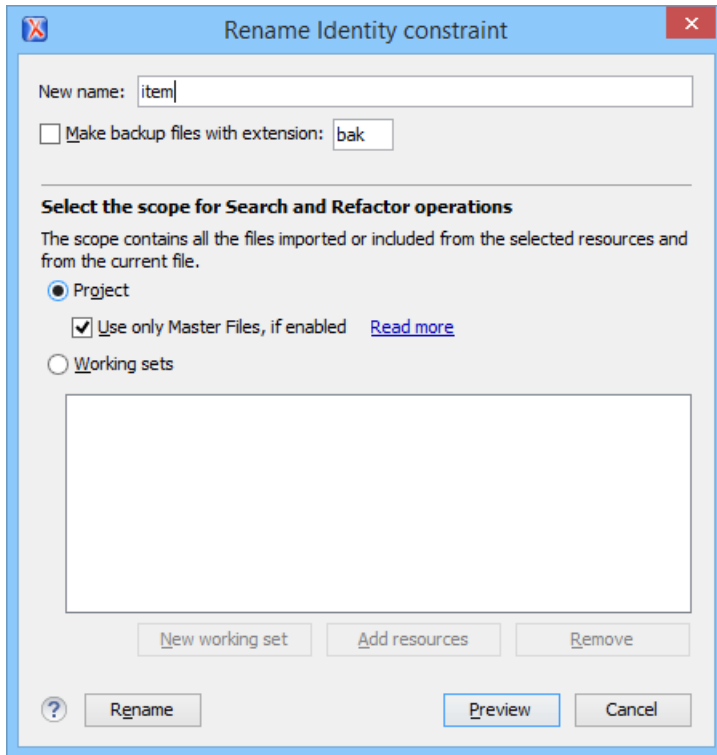


Figure 278: Rename Identity Constraint Dialog Box

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](#).

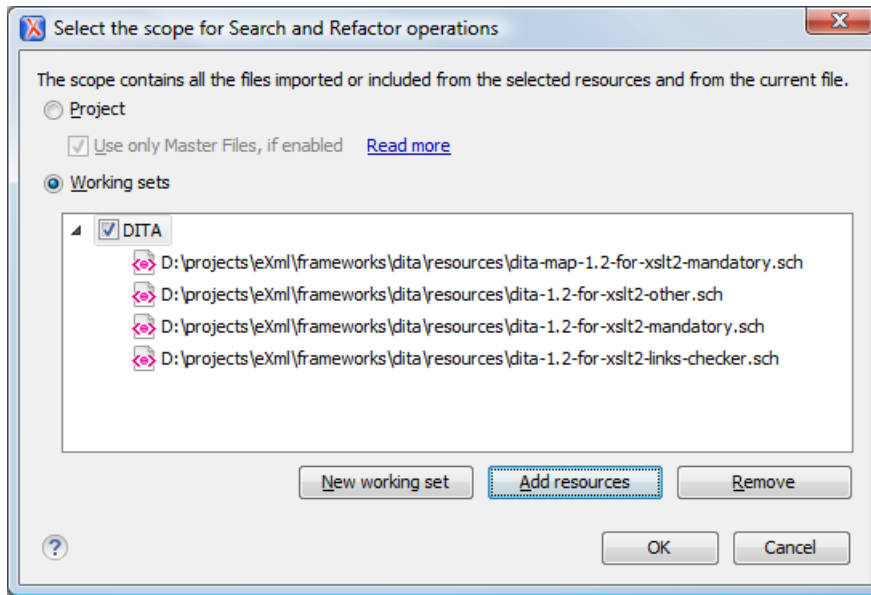


Figure 279: Change Scope Dialog Box

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.

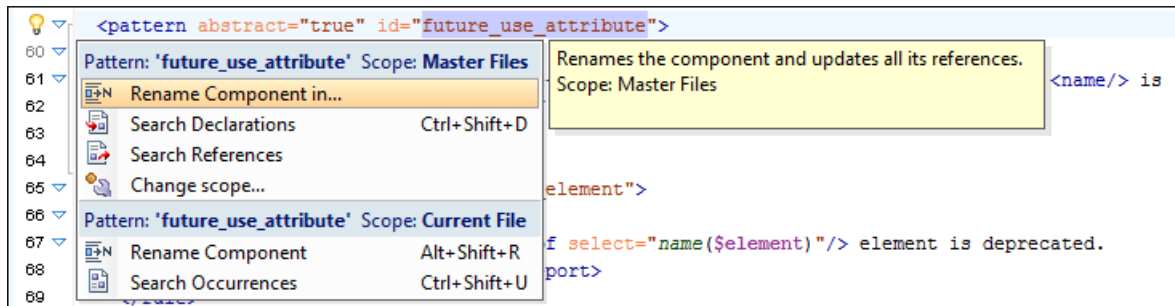


Figure 280: Schematron Quick Assist Support

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 Developer 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 Developer assists you in editing Schematron Quick Fixes with schema-based content completion, syntax highlighting, and validation as you type.

This section includes details about the Schematron Quick Fixes feature and how to customize them.

Related information

<http://blog.oxygenxml.com/2015/05/schematron-checks-to-help-technical.html>

oXygen XML Blog: Schematron Checks to Help Technical Writing

Customizing Schematron Quick Fixes

You can customize Schematron Quick Fixes by editing them directly in the current Schematron file or in a separate 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 types of operations are supported:
 - `<sqf:add>` - To add a new node or fragment in the document.
 - `<sqf:delete>` - To remove a node from the document.
 - `<sqf:replace>` - To replace a node with another node or fragment.
 - `<sqf:stringReplace>` - To replace text content with other text or a fragment.

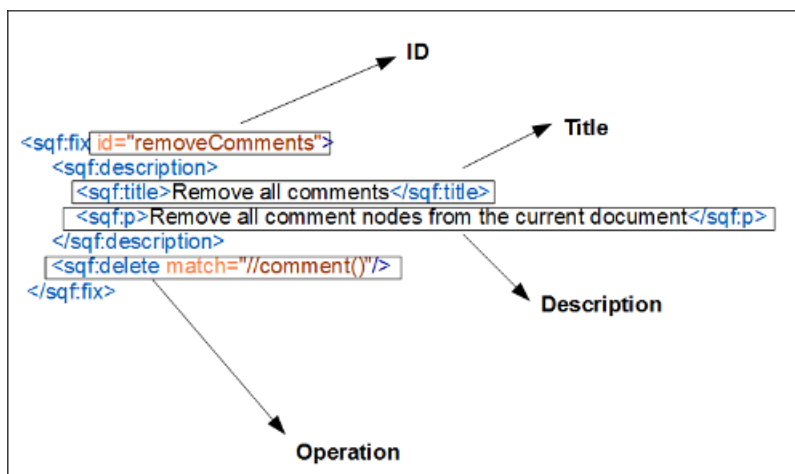


Figure 281: Schematron Quick Fix Components

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.

Schematron Quick Fix Operations

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 Developer 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 is 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: Regular expressions in the <sqf:stringReplace> element always has 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>
```

Formatting and Indenting Inserted Content

The content that is inserted by the **Add**, **Replace**, or **String Replace** 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>
    <entry>First column</entry><xsl:text>
    <entry>Second column</entry><xsl:text>
  </row><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>
```

Use-When Condition

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 $lastWidth" role="replace">
  <sqf:description>
    <sqf:title>Subtract the excessive width from the last element.</sqf:title>
  </sqf:description>
  <let name="delta" value="$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>
```

Executing Schematron Quick Fixes in Documents Other than the Current One

You can apply Schematron Quick Fixes over the nodes from referred documents (referred 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">
```

Additional Elements Supported in the Schematron Quick Fixes

- <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 Developer 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.
- <sqf:user-entry>** Allows you to specify a value that will be inserted after the user selects the quick fix. If multiple `user-entry` elements are defined, Oxygen XML Developer will display a dialog box for each one, in which the user can insert values.

Other SQF Notes

 **Note:** The `sqf:default-fix` attribute is ignored in Oxygen XML Developer.

For more details on editing Schematron Quick Fixes, go to: [Schematron Quick Fix Specifications](#)

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 disable 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.

Content Completion in SQF

Oxygen XML Developer helps you edit Schematron Quick Fixes embedded in a Schematron document by offering items that are valid at the cursor position in a *Content Completion Assistant*. 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 9.6.0.7 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 Developer 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.

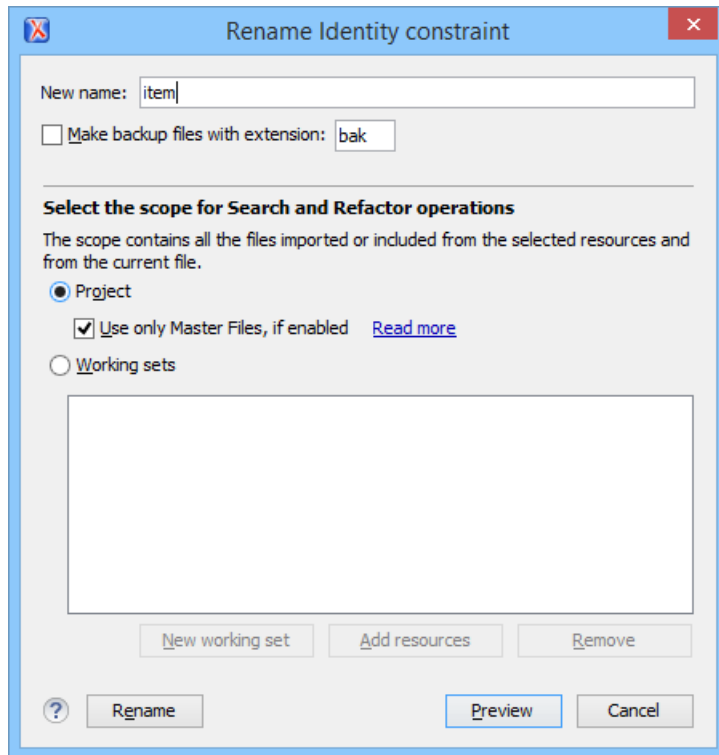


Figure 282: Rename Identity Constraint Dialog Box

Embed Schematron Quick Fixes in Relax NG or XML Schema

Schematron Quick Fixes can be embedded into a Relax NG or XML Schema within the Schematron rules from annotations (using the `appinfo` element), or in any Schematron rule of a RELAX NG Schema.

Oxygen XML Developer is able to extract and use the embedded Schematron Quick Fixes. To make the Schematron Quick Fixes available, validate the document with both the RELAX NG schema and its embedded Schematron rules.

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 Developer 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 Developer `lib` folder.
2. Restart the application.

Standalone SVG Viewer

Oxygen XML Developer includes a simple *SVG Viewer* that allows you to work with SVG images.

To open the viewer, select **SVG Viewer** from the **Tools** menu.

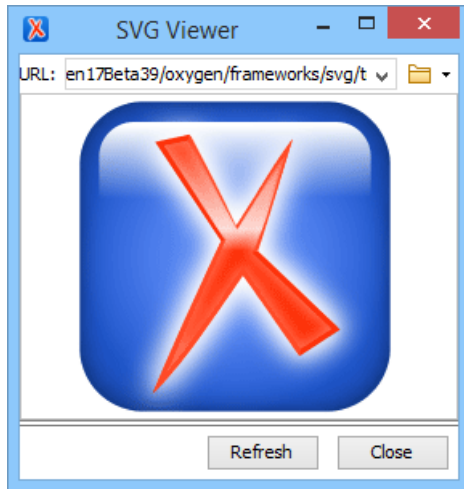


Figure 283: SVG Viewer

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

[SVG 1.2 Rendering Issues](#) on page 1009

Integrated SVG Viewer in the Results Panel

Oxygen XML Developer 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 Developer or exported from an 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.

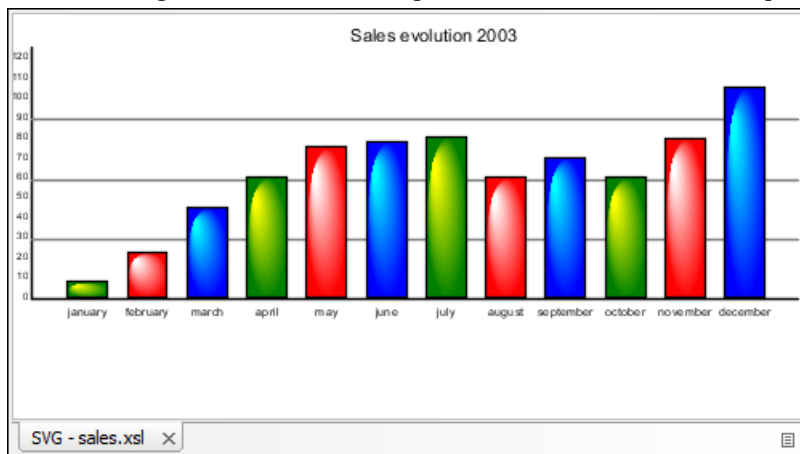


Figure 284: Integrated SVG Viewer

Editing XHTML Documents

Oxygen XML Developer provides support for editing XHTML files. Oxygen XML Developer includes XHTML catalogs and document templates to help you get started. You can also find a sample file in the `[OXYGEN_INSTALL_DIR]/samples/xhtmll` 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](#).
- **Validation** - Easily identify errors and their location with the Oxygen XML Developer [XML validation features](#).
- **Content Completion** - The **Content Completion Assistant** displays a list of context-sensitive proposals that are valid at the current cursor position.

- **Import HTML as XHTML** - Oxygen XML Developer includes support for [importing HTML files as an XML document](#).
- **Remote Editing** - Oxygen XML Developer 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**.

Related information

[XHTML Document Type](#) on page 814

Spell Checking

Oxygen XML Developer includes an [automatic \(as-you-type\) spell checking feature](#), as well as a manual spell checking action to opens a **Spelling** dialog box that offers a variety of options. To open this dialog box, use the  **Check Spelling** action on the toolbar or from the **Edit** menu.

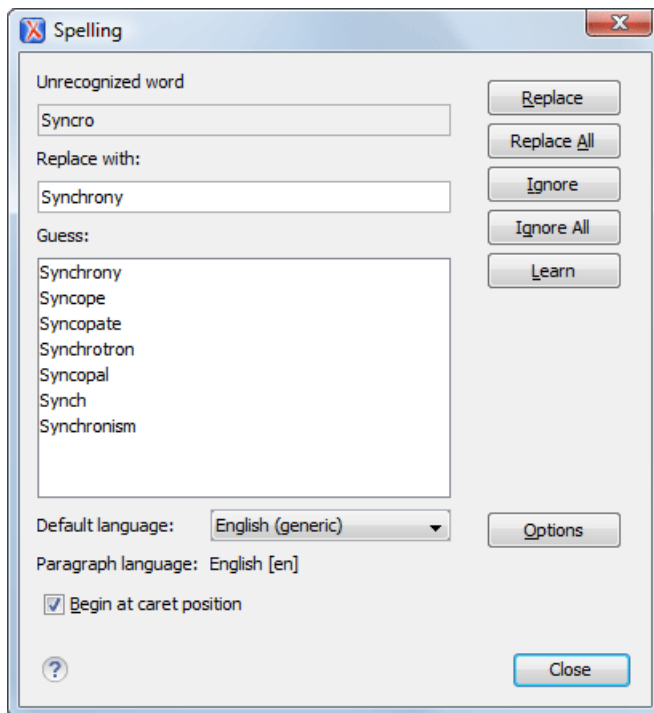


Figure 285: Check Spelling Dialog Box

The **Spelling** dialog box contains the following fields:

- **Unrecognized word** - Contains 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 is suggested to replace the unrecognized 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 dictionary used by the spelling engine.
- **Paragraph language** - In an XML document, you can mix content written in multiple languages. To tell the spell checker engine what language was used to write a specific section, you need to set the language code in the `lang` or `xml:lang` attribute to that section. Oxygen XML Developer automatically detects such sections and instructs the spell checker engine to apply the appropriate dictionary.

- **Replace** - Replaces the currently highlighted word in the XML document, with the selected word in the **Replace with** field.
- **Replace All** - Replaces all occurrences of the currently highlighted word in the XML document, with the selected word in the **Replace with** field.
- **Ignore** - Ignores the first occurrence of the unrecognized word and allows you to continue checking the document. Oxygen XML Developer skips the content of the XML elements *marked as ignorable*.
- **Ignore All** - Ignores all instances of the unknown word in the current document.
- **Learn** - Includes the unrecognized word in the list of valid words.
- **Options** - Sets the configuration options of the spell checker.
- **Begin at cursor position** - Instructs the spell checker to begin checking the document starting from the current cursor position.
- **Close** - Closes the dialog box.

Spell Checking Dictionaries

There are two spell checking engines available in Oxygen XML Developer: **Hunspell** checker (default setting) and **Java** checker. You can set the spell check engine in the *Spell checking engine* preferences page. The dictionaries used by the two engines differ in format, so you need to follow specific procedures to add another dictionary to your installation of Oxygen XML Developer.

Dictionaries for the Hunspell Checker

The Hunspell spell checker is open source and has LGPL license. The format of the Hunspell spell dictionary is supported by Mozilla, OpenOffice and the Chrome browser. Oxygen XML Developer includes the following built-in dictionaries for the Hunspell checker:

- English (US)
- English (UK)
- French
- German
- Spanish.

Each language-country variant combination has its specific 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 dictionary from this list follow *these instructions*.

Add Dictionaries for the Hunspell Checker

To add new spelling dictionaries to Oxygen XML Developer, or to replace an existing one, follow these steps:

1. *Download the files* you need for your language dictionary.
2. The downloaded .oxt file is a *zip* archive. If you are creating a new dictionary, copy the .aff and .dic files from this archive in the `spell` subfolder of the Oxygen XML Developer preferences folder.

The Oxygen XML Developer preferences folder is

>[APPLICATION-DATA-FOLDER]/com.oxygenxml.developer, where [APPLICATION-DATA-FOLDER] is:

- C:\Users\[USER-NAME]\AppData\Roaming on Windows Vista/7/8/10
- [USER-HOME-FOLDER]/Library/Preferences on OS X
- [USER-HOME-FOLDER] on Linux

3. If you are updating an existing dictionary, copy the .aff and .dic files into the following folder:
[OXYGEN_INSTALL_DIR]/dicts/spell.
4. Restart the application after copying the dictionary files.



Note: You can specify that Oxygen XML Developer includes dictionaries and term lists from a custom location by selecting the *Include dictionaries and term list from option* and specifying its path in the **Dictionaries** preferences page.

Related information

[Adding Term Lists](#) on page 509

Dictionaries for the Java Checker

A Java spell checker dictionary has the form of a `.dar` file located in the directory `[OXYGEN_INSTALL_DIR]/dicts`. Oxygen XML Developer includes the following built-in dictionaries for the Java checker:

- English (US)
- English (UK)
- English (Canada)
- French (France)
- French (Belgium)
- French (Canada)
- French (Switzerland)
- German (old orthography)
- German (new orthography)
- Spanish

A pre-built dictionary can be added by copying the corresponding `.dar` file to the folder `[OXYGEN_INSTALL_DIR]/dicts` and restarting Oxygen XML Developer. There is one dictionary for each language-country variant combination.

Learned Words

Spell checker engines rely on dictionaries to decide that 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](#).



Note: To delete items from the list of learned words, select **Delete learned words** in the [Editor > Spell Check > Dictionaries preferences page](#).

Adding 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 Developer provides a way of adding industry-specific terms to the spell checker engine. This involves creating a file (*term list*) similar in structure to the file used by Oxygen XML Developer for storing *learned words*.

The term list files are specific for each language and can also be specific to each domain or area of expertise (for example, *legal*, *medical*, *automotive*, etc.).

To create a custom term file, follow this procedure:

1. Create a text file using one of the following naming conventions:

- `<languageCode>-<name>.tdi`

For example, a term file for a list of French medical terms could be named `fr-medical.tdi`.

- `<languageCode_countryCode>-<name>.tdi`

For example, a term file for a list of English legal terms can be named `en_US-legal.tdi` for American English, or `en_GB-legal.tdi` for their British counterparts.

2. Edit the term file you just created and add the terms. Make sure that you only add one term per line.
3. Save the term file in the Oxygen XML Developer default dictionary location:

- [HOME_DIR]\AppData\Roaming\com.oxygenxml.developer/spell folder on Windows Vista/7/8/10.
- [HOME_DIR]\Application Data\com.oxygenxml.developer\spell folder on Windows XP.
- [HOME_DIR]/Library/Preferences/com.oxygenxml.developer/spell folder on OS X.
- [HOME_DIR]/com.oxygenxml.developer/spell folder on Linux.



Note: To change the default dictionary location folder go to the [Editor > Spell Check > Dictionaries preferences page](#).

Ignored Words (Elements)

The content of some XML elements such as `programlisting`, `codeblock`, or `screen` should always be skipped by the spell checking process. The skipping can be done manually, word by word by the user using the **Ignore** button of [the Spelling dialog box](#), or automatically by maintaining a set of known element names that should never be checked. You maintain this set of element names [in the user preferences](#) as a list of XPath expressions that match the elements.

Only a small subset of XPath expressions is supported, that is only the '/' and '/' separators and the '*' wildcard. Two examples of supported expressions are `/a/*/b` and `//c/d/*`.

Automatic Spell Check

Oxygen XML Developer includes an option to automatically check the spelling as you type. 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 enabled, 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.

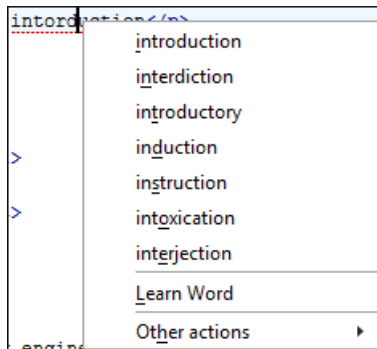


Figure 286: Automatic Spell Checking in Text Mode

The contextual menu includes the following actions:

Delete Repeated Word	Allows you to delete repeated words.
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.
Other actions	This submenu give you access to all the usual contextual menu actions.

Related information

[Learned Words](#) on page 509
Learned Words

Spell Checking in 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.

The spelling corrections are displayed in *the Results view*, that allows you to group the reported errors as a tree with two levels.

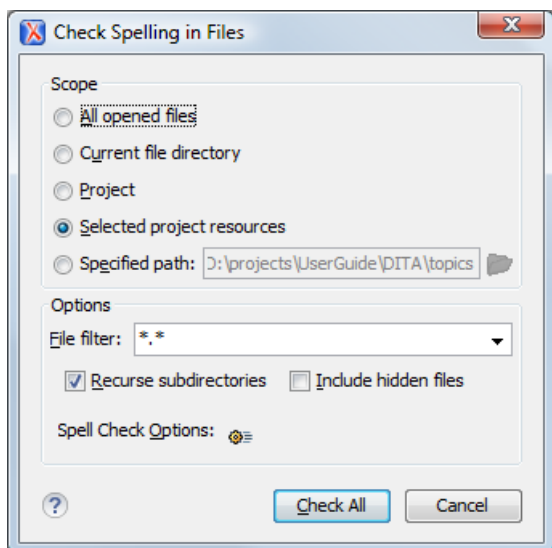


Figure 287: Check Spelling in Files Dialog Box

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 enabled, 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 enabled, 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*.

Loading Large Documents

When you open a document with a file size larger than the limit configured in *Open/Save preferences*, Oxygen XML Developer 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 Developer 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 option Format and indent the document on open](#) is disabled 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 disabled 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

A handy addition to the document editing is the **Scratch Buffer** view 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 contextual menu contains basic edit actions such as **Cut**, **Copy**, and **Paste**.

Handling Read-Only Files

If a file marked as read-only is opened in Oxygen XML Developer 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

Documents that contain long lines can affect performance when opened in **Text** mode. If you choose to present the document with the [Line wrap option](#) enabled, the following features may be affected:

- 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.

To canonicalize something means to put it in a standard format that everyone generally uses. 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 term canonical XML refers to XML that is in canonical form. The XML canonicalization method is the algorithm defined by this specification that generates the canonical form of a given XML document or document subset. The term XML canonicalization refers to the process of applying the XML canonicalization method to an 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 allow 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 Developer: Canonical XML (or Inclusive XML Canonicalization)([XMLC14N](#)) and Exclusive XML Canonicalization([EXCC14N](#)). The first is used for XML where the context doesn't change while the second was designed for canonicalization where the context might 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 perspective 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 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 tasks

[Example of How to Digitally Sign XML Files or Content](#) on page 518

Related information

[Certificates](#) on page 515

[Canonicalizing Files](#) on page 515

[Signing Files](#) on page 516

[Verifying Signature](#) on page 518

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*.

A *keystore* is an encrypted file that contains private keys and certificates. 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*.

In Oxygen XML Developer there are provided 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 513

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.

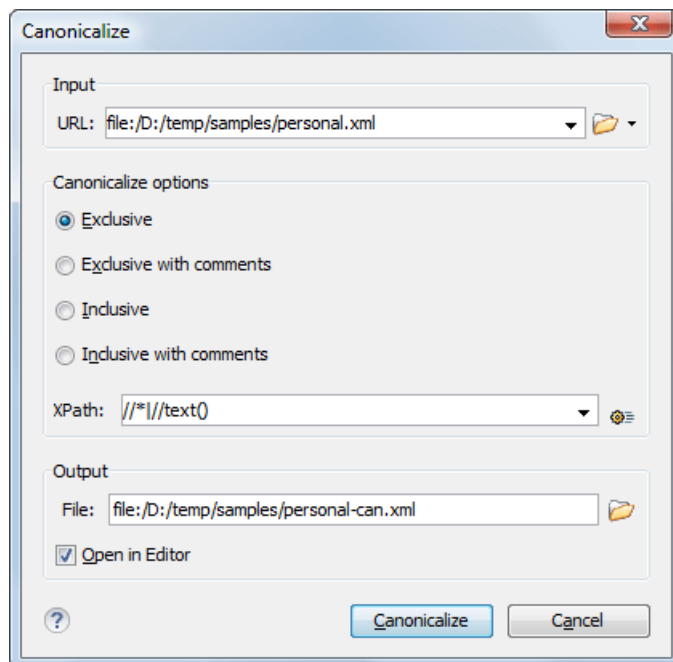


Figure 288: Canonicalization Settings Dialog Box

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 perspective 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 checked, the output file will be opened in the editor.

Related information

[Digital Signatures Overview](#) on page 513

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.

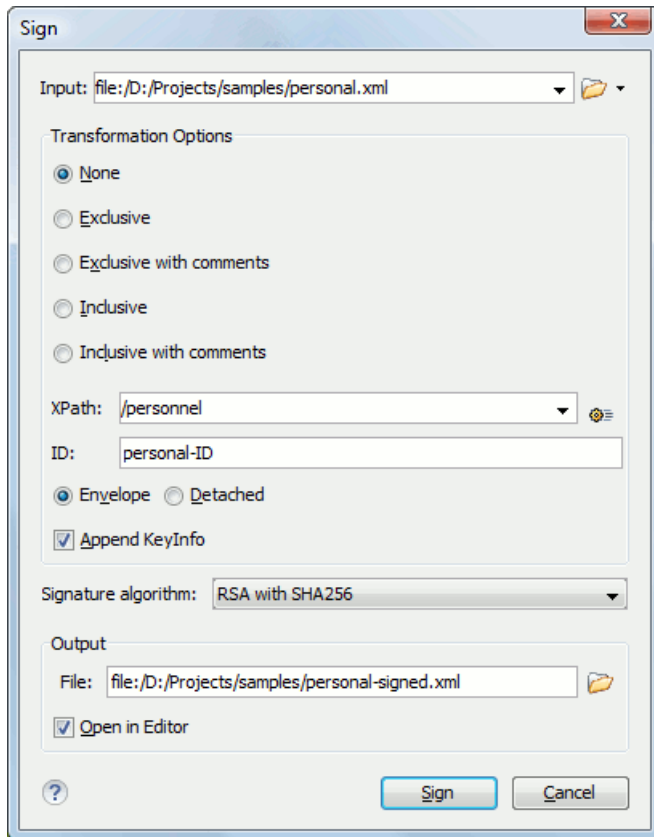


Figure 289: Signature Settings Dialog Box

The following options are available:



Note: If Oxygen XML Developer 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.

! Could not obtain a valid certificate. [You must 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 perspective 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 checked, 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 checked, the output file will be opened in Oxygen XML Developer.

Related tasks

[Example of How to Digitally Sign XML Files or Content](#) on page 518

Related information

[Digital Signatures Overview](#) on page 513

[Verifying Signature](#) on page 518

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 tasks

[Example of How to Digitally Sign XML Files or Content](#) on page 518

Related information

[Digital Signatures Overview](#) on page 513

[Signing Files](#) on page 516

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 Developer includes a signature tool that allows you to digitally sign XML documents or specific content.

The Oxygen XML Developer 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 Developer 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 Developer 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 513

[Signing Files](#) on page 516

[Verifying Signature](#) on page 518

Compare Files or Directories

Oxygen XML Developer 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 Developer 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*).

For example, to start the comparison between the two Windows directories `c:\Program Files` and `c:\ant`, use the following command:

```
diffDirs.bat "c:\Program Files" "c:\ant"
```

If there are spaces in the path names, surround the paths with quotes. If you pass only one argument, you are prompted to manually choose the second directory or archive. This is also true for the files comparison utility.

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 Developer installation folder (`diffFiles.exe`).

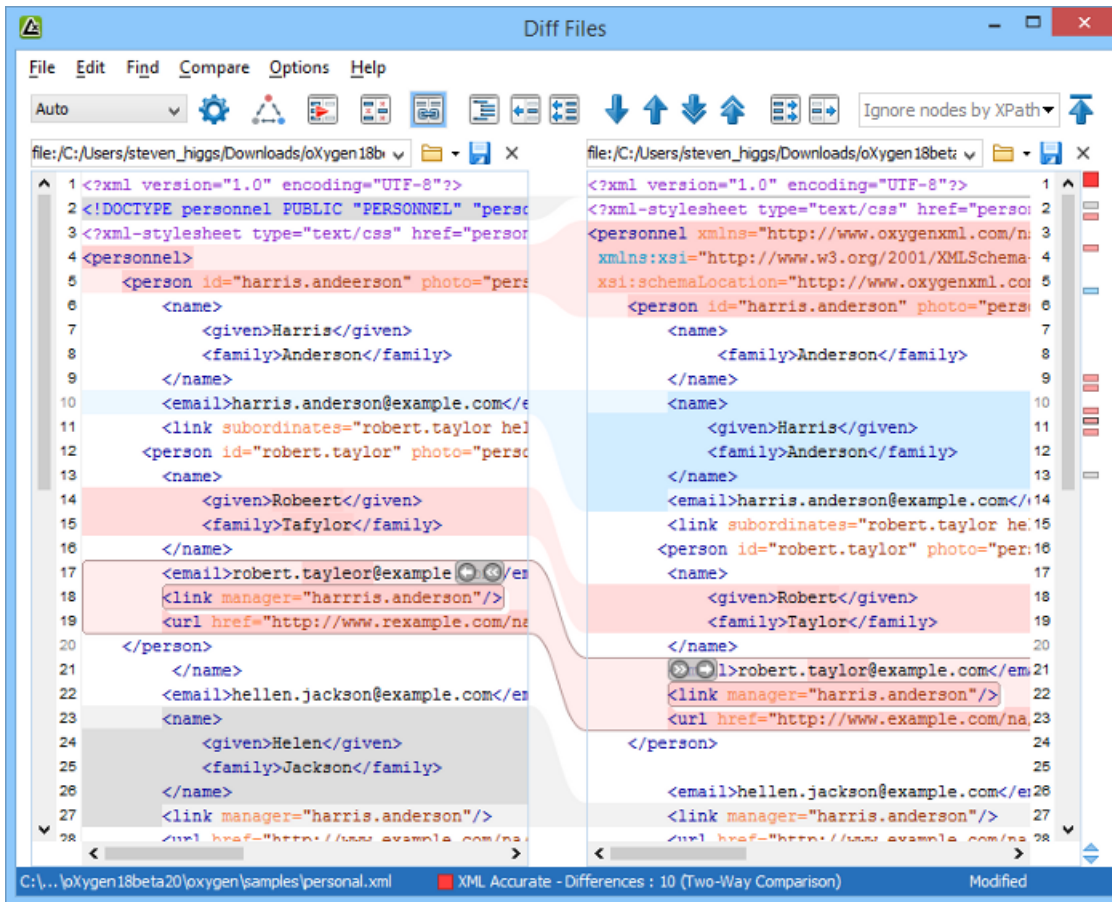


Figure 290: Compare Files Tool

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 perspective 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.

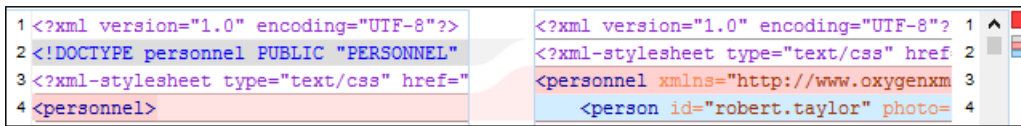


Figure 291: Two-Way Differences

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 Developer 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 Developer 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, a remote 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.

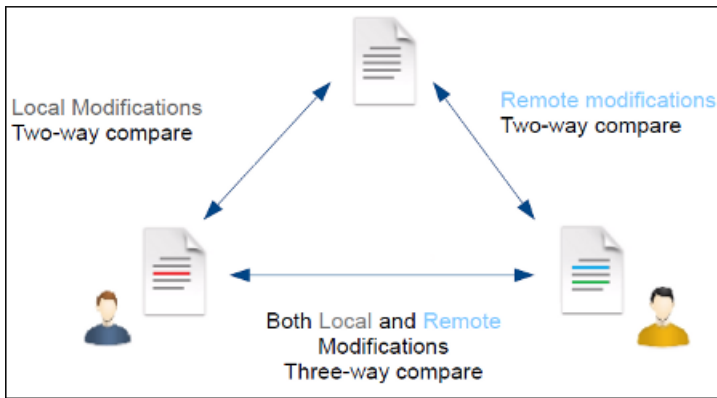


Figure 292: Three-Way Comparison

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 perspective 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 **Ancestor** 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.

7	<given>Robert</given>		<given>Helen</given>	8
8	<family>Taylor</family>		<family>Jackson</family>	9
9	</name>		</name>	10
10	<email>robert.taylor@example		<email>helen.jackson@example	11

Figure 293: 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 Developer 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

Oxygen XML Developer 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 are processed on text nodes using a word level comparison, meaning that it looks for identical words. This second level comparison makes it easier to spot precise differences and you can merge or reject the precise modifications.

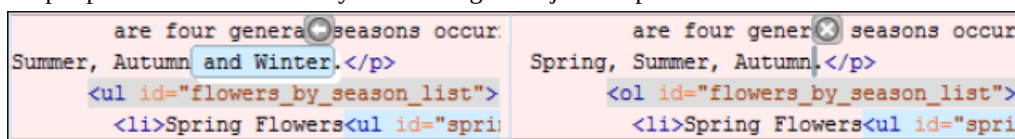


Figure 294: Second Level Diff Comparison



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.

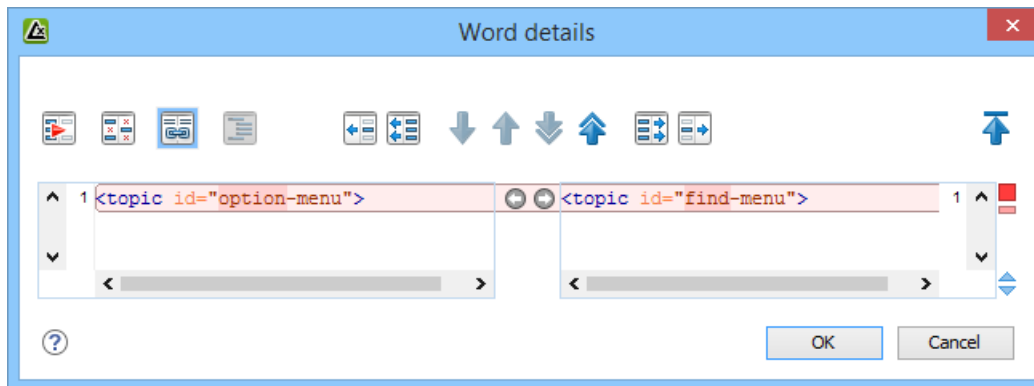


Figure 295: Word Level Comparison

To do a character level comparison, select **Show Character Level details** from the contextual menu or **Compare** menu.

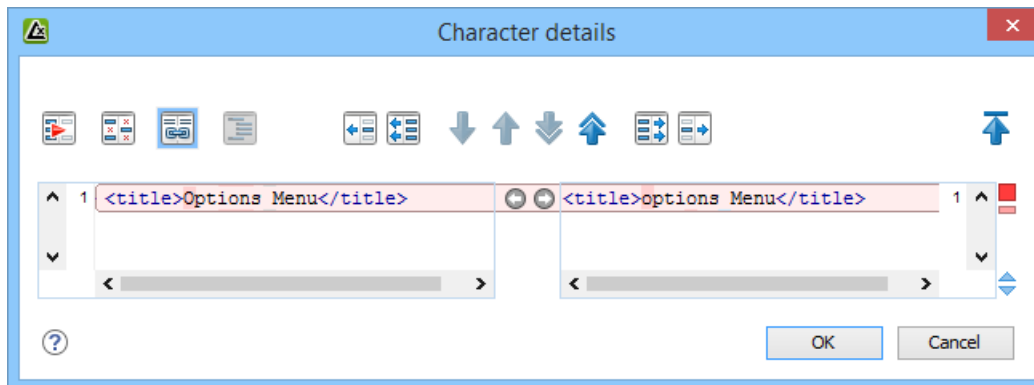


Figure 296: Character Level Comparison

Related information

[Files Comparison Preferences Page](#) on page 132

[Compare Directories](#) on page 530

Compare Files Tool Menus

This section describes the actions that are available in the menus of the **Files Comparison** interface.

File Menu

The **File** menu of the files comparison tool contain the following actions:

Source	The file that is displayed in the left side of the application window.
- **Source** > **Open** - Browsers for a source file.
 - **Source** > **Open URL** - Opens the URL to be used as a source file. See [Open URL](#) for details.
 - **Source** > **Open File from Archive** - Browsers an archive content for a source file.
 - **Source** > **Save** - Saves the changes made in the source file.
 - **Source** > **Save As** - Displays the **Save As** dialog box that allows you to save the source file with a new name.

- **Source** >  **Close** - Closes the source file.

Target The file that is displayed in the right side of the application window.

- **Target** >  **Open** - Browses for a target file.
- **Target** >  **Open URL** - Opens the URL to be used as a target file. See [Open URL](#) for details.
- **Target** >  **Open File from Archive** - Browses an archive content for a target file.
- **Target** >  **Save** - Saves the changes made in the target file.
- **Target** > **Save As** - Displays the **Save As** dialog box that allows you to save the target file with a new name.
- **Target** >  **Close** - Closes the target file.

Ancestor The base file that will be compared with both files in a [three-way comparison](#).

- **Ancestor** >  **Open** - Browses for a target file.
- **Ancestor** >  **Open URL** - Opens the URL to be used as a target file. See [Open URL](#) for details.
- **Ancestor** >  **Open File from Archive** - Browses an archive content for a target file.

Exit Quits the application.

Edit Menu

The following actions are available in the **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

The **Find** menu includes the following actions:

 Find/Replace	Perform <i>find/replace</i> operations in the currently focused Editor .
Find Next	Go to the next match using the same options as the last <i>find</i> operation. This action runs in both editor panels.
Find Previous	Go to the previous match using the same options as the last <i>find</i> operation. This action runs in both editor panels.

Compare Menu

The following actions are available in the **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 of the application window.

 Next Block of Changes (Ctrl + Period (Command + Period on OS X))

Jumps to the next block of changes. This action is disabled 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 disabled 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 disabled 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 disabled 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 target file in the right side to the source file in the left side.

 Copy All Changes from Left to Right

Copies all changes from the source file in the left side to the target file in the right side.

 Copy Change from Right to Left

Copies the selected difference from the target file in the right side to the source file in the left side.

 Copy Change from Left to Right

Copies the selected difference from the source file in the left side to the target file in the right side.

 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 the same in both files to make it easier to compare the files.

Options Menu

The **Options** menu contains the following actions:

Preferences

Opens the preferences.

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

The **Help** menu contains the following actions:

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 enabled, when you select Help or press F1 while hovering over any part of the interface, Oxygen XML Developer attempts to open the help documentation in online mode. If this option is disabled 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 <code>com.oxygenxml.report.problems.url</code> system property. The report is sent in XML format through the <code>report</code> parameter of the POST HTTP method.
Support Center	Opens the Oxygen XML Developer Support Center web page in a browser.
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 Developer. 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 XML Developer support team.

Compare Toolbar of the Compare Files Tool

The toolbar of the **Compare Files** tool contains the operations that can be performed on the source and target files or XML fragments.



Figure 297: 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 Developer, 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 of the application window.

 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

Enables or disables synchronized scrolling so that a selected difference can be seen on both sides of the application window. This option is enabled 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 the same in both files to make it easier to compare the files.

 Copy Change from Right to Left

Copies the selected difference from the target file in the right side to the source file in the left side.

 Copy All Changes from Right to Left

Copies all changes from the target file in the right side to the source file in the left side.

 Next Block of Changes (Ctrl + Period (Command + Period on OS X)**)**

Jumps to the next block of changes. This action is disabled 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 disabled 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 disabled 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 disabled 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 source file in the left side to the target file in the right side.

 Copy Change from Left to Right

Copies the selected difference from the source file in the left side to the target file in the right side.

Ignore Nodes by XPath

You can use this text field to enter an XPath 2.0 expression to ignore certain nodes from the comparison. This field is only available when comparing XML documents using the **XML Fast** or **XML Accurate** algorithms. If an XPath 2.0 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.

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 Developer installation folder (diffDirs.exe).

The directories comparison results are presented as a tree of files and directories. The directories and folders 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 Developer starts a [file comparison](#) between the two files, using the **Compare Files** tool.

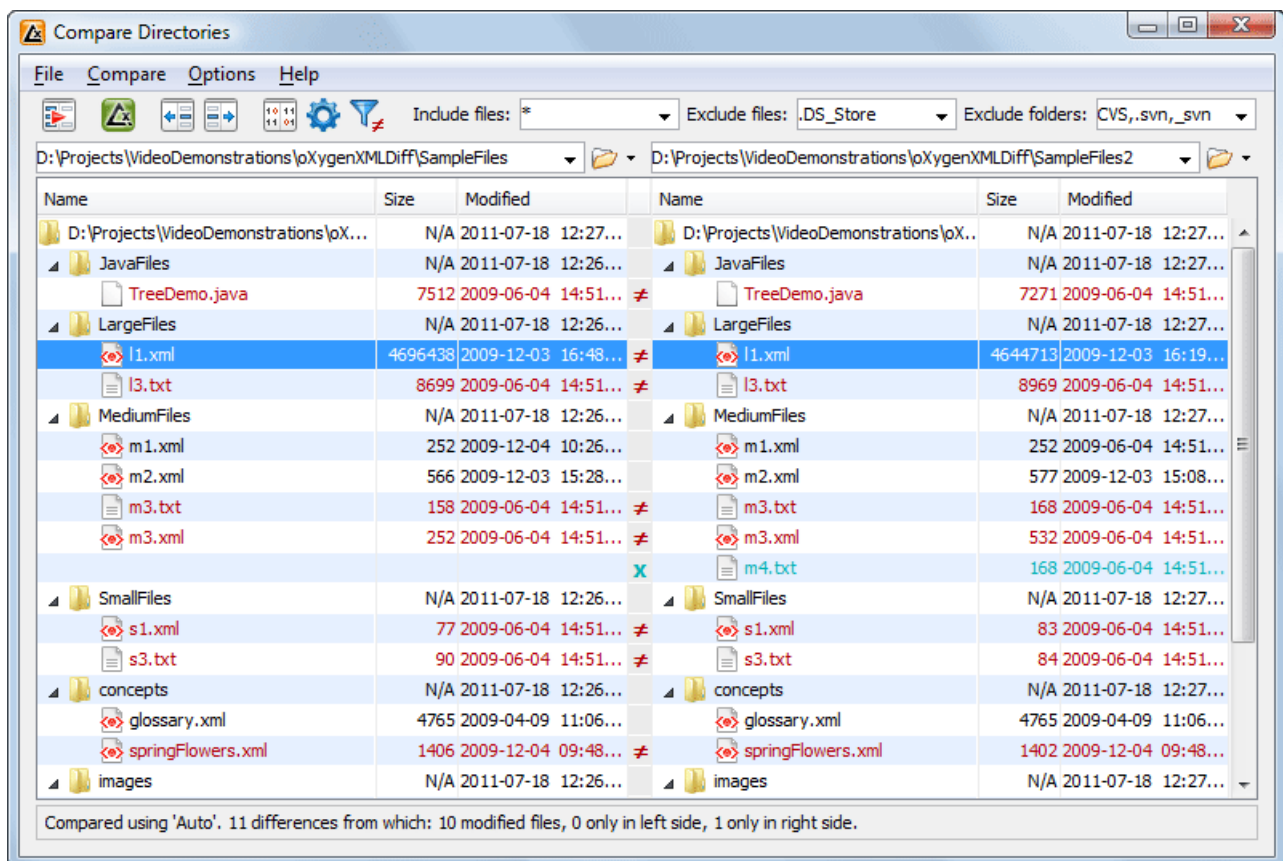


Figure 298: Compare Directories Tool

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 Developer treat supported archives as directories, enable 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.

Related information

[Compare Files](#) on page 519

Compare Directories Tool Menus

This section describes the actions that are available in the menus of the **Directories Comparison** interface.

Compare Menu

This menu contains the following 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).

Options Menu

The **Options** menu contains the following actions:

Preferences	Opens the preferences.
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

The **Help** menu contains the following actions:

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 enabled, when you select Help or press F1 while hovering over any part of the interface, Oxygen XML Developer attempts to open the help documentation in online mode. If this option is disabled 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 <code>com.oxygenxml.report.problems.url</code> system property. The report is sent in XML format through the <code>report</code> parameter of the POST HTTP method.
Support Center	Opens the Oxygen XML Developer Support Center web page in a browser.
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 Developer. 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 XML Developer support team.

Compare Toolbar of the Compare Directories Tool

The toolbar contains the following actions:

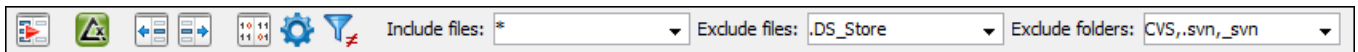


Figure 299: Compare toolbar

 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.

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*.

Chapter

8

Transforming Documents

Topics:

- [Transformation Scenarios](#)
- [WebHelp System Output](#)

This chapter presents the various transformation scenarios and outputs available in Oxygen XML Developer.

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 Developer 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 Developer, 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 Developer, 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 Developer 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 Developer included preconfigured built-in transformation scenarios that are used for common transformations. To obtain the desired output, use the **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on OS X))** action from the toolbar or the **Document > Transformation** menu and choose one of the built-in scenarios for the current document.

You can use the **Apply Transformation Scenario(s)** action 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 Developer 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. The **XSL URL** field of the default transformation scenario contains the URL from the `href` attribute of the processing instruction. By default, the **Use xml-stylesheet declaration** checkbox is enabled, Saxon is used as the transformation engine, and no FO processing is performed. The result of the transformation is store in a file with the same URL as the

edited document, but the extension is changed to `html`. The name and path are preserved because the output file name is specified with the help of two *editor variables*: `${cfd}` and `${cfn}`.

DocBook 4 Transformation Scenarios

Default transformation scenarios allow you to convert DocBook 4 to DocBook 5 documents and transform DocBook documents to WebHelp, PDF, HTML, HTML Chunk, XHTML, XHTML Chunk, EPUB and EPUB 3.

Related information

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

[Editing a Transformation Scenario](#) on page 596

DocBook4 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 the WebHelp Classic with Feedback System](#) on page 649.

To watch our video demonstration about the feedback-enabled WebHelp system, go to http://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.languageThis parameter is used if the language is not detected in the DITA map. The default value is `en`.**clean.output**Deletes all files from the output folder before the transformation is performed (or the default value is `no`).**l10n.gentext.default.language**This parameter is used to identify the correct stemmer that differs from language. 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 output (the default value is `false`).**webhelp.copyright**Adds a small copyright text that appears at the end of the *Table of Contents* page.**webhelp.custom.resources**

The file path to a directory that contains resources files. All files from this directory are included in the WebHelp output.

webhelp.faviconThe 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 page. To integrate social media features (such as widgets for Facebook™, Twitter™, Google™), the XML must be well-formed, each widget must be in separate `div` or `span` element, and each widget is included in an XML comment (also, the start and end tags for the XML comment). The following code excerpt 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];
    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-label="share">
    </div>
</div>
```

webhelp.footer.includeSpecifies whether or not to include footer in each WebHelp page. Possible values are `yes` and `no`. If set to `yes` and the `webhelp.footer.file` parameter is set, the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default 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, the browser goes to the 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 users are notified of updates.

**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 for indexing Japanese content in the WebHelp pages. This indexer does not come with Oxygen XML Developer or the Oxygen XML WebHelp plugin. To use it, you need to download it from the Internet.

<http://mvnrepository.com/artifact/org.apache.lucene/lucene-analyzers-kuromoji/4.0.0>
 DITA_OT_DIR/plugins/com.oxygenxml.webhelp/lib directory.

webhelp.search.ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer in 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 the `OXYGEN_INSTALL_DIR/frameworks/docbook/xsl/com.oxygenxml` directory, or it can be a custom skin CSS generated with the [Oxygen Skin Builder](#).

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 System Output](#) on page 611

DocBook to PDF Output Customization

Main steps for customization of PDF output generated from DocBook XML documents.

When the default layout and output look of the DocBook to PDF transformation need to be customized, the following main steps should be followed. In this example a company logo image is added to the front matter of a book. Other types of customizations should follow some similar 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 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 Developer and set the customization layer in [the XSL URL property of the scenario](#).

Related information

<http://www.sagehill.net/docbookxsl/PrintCustomEx.html>

The book **DocBook XSL: The Complete Guide** by [Bob Stayton](#) contains more details about customizing the PDF output.

http://www.oxygenxml.com/demo/DocBook_Customization.html

Video demonstration for creating a DocBook customization layer in Oxygen XML Developer.

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. In your DocBook to EPUB transformation, 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.

DocBook to DITA Transformation

Oxygen XML Developer 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.

DocBook 5 Transformation Scenarios

Default transformation scenarios allow you to transform DocBook 5 documents to WebHelp, PDF, HTML, HTML Chunk, XHTML, XHTML Chunk, EPUB, and EPUB 3.

Related information

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

[Editing a Transformation Scenario](#) on page 596

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 the WebHelp Classic with Feedback System](#) on page 649.

To watch our video demonstration about the feedback-enabled WebHelp system, go to http://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`.

clean.output

Deletes all files from the output folder before the transformation is performed (or not, if the value is `false`). The default value is `no`.

l10n.gentext.default.language

This parameter is used to identify the correct stemmer that differs from language to language. 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 search (or not, if the value 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 are included in 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 page. To integrate social media features (such as widgets for Facebook™, Twitter™, Google™), the XML must be well-formed, each widget must be in separate `div` or `span` element, and each widget is included in an XML comment (also, the start and end tags for the XML comment). The following code excerpt 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)) { 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-label="share">
  </div>
```

webhelp.footer.include

Specifies whether or not to include footer in each WebHelp page. Possible values are `true` or `false`. If set to `true`, the footer content of that file is used as footer. If the `webhelp.footer.file` has no value then the 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, the browser goes to the 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 users are notified of new versions.



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 analyzer. The analyzer is used for indexing Japanese content in the WebHelp pages. This indexer does not come with Oxygen XML Developer or the Oxygen XML WebHelp plugin. To use it, you need to download it from <http://mvnrepository.com/artifact/org.apache.lucene/lucene-analyzers-kuromoji/4.1.0> and place it in the `DITA_OT_DIR/plugins/com.oxygenxml.webhelp/lib` directory.

webhelp.search.ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the output. 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 the `OXYGEN_INSTALL_DIR/frameworks/docbook/xsl/com.oxygenxml.webhelp/skin` directory, or it can be a custom skin CSS generated with the *Oxygen Skin Builder*.

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 System Output](#) on page 611

DocBook to PDF Output Customization

Main steps for customization of PDF output generated from DocBook XML documents.

When the default layout and output look of the DocBook to PDF transformation need to be customized, the following main steps should be followed. In this example a company logo image is added to the front matter of a book. Other types of customizations should follow some similar 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 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 Developer and set the customization layer in [the XSL URL property of the scenario](#).

Related information

<http://www.sagehill.net/docbookxsl/PrintCustomEx.html>

The book **DocBook XSL: The Complete Guide** by [Bob Stayton](#) contains more details about customizing the PDF output.

http://www.oxygenxml.com/demo/DocBook_Customization.html

Video demonstration for creating a DocBook customization layer in Oxygen XML Developer.

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. In your DocBook to EPUB transformation, 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.

DocBook to DITA Transformation

Oxygen XML Developer 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.

DITA Topic Transformation Scenarios

The following default transformation scenarios are available for DITA Topics:

- **DITA XHTML** - Transforms a DITA topic to XHTML using DITA Open Toolkit.
- **DITA PDF** - Transforms a DITA topic to PDF using the DITA Open Toolkit and the Apache FOP engine.

Related information

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

[Editing a Transformation Scenario](#) on page 596

DITA Map Transformation Scenarios

The following default transformations scenarios are available:

- Predefined transformation scenarios allow you to transform a DITA map to a variety of outputs, such as PDF, ODF, XHTML, WebHelp, EPUB, and CHM files.
- **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 transformation scenario if you want to generate a DITA map statistics report containing 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 596

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

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 611

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 (or **no** and **yes** values are valid and the default value is **no**).

fix.external.refs.com.oxygenxml (Only supported when the DITA OT

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 u

transformation process is started from Oxygen XML Developer)	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 parameter has no impact on your edited DITA content. Allowed values: <code>true</code> or <code>false</code> (default is <code>false</code>).
use.stemming	Controls whether or not you want to include stemming search algorithms into the published output (default setting is <code>false</code>).
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 <i>favicon</i> in the WebHelp output.
webhelp.reload.stylesheet	Set this parameter to <code>true</code> if you have out of memory problems when generating WebHelp. It will increase processing time but decrease the memory footprint. The default value is <code>false</code> .
webhelp.search.japanese.dictionary	The file path of the dictionary that will be used by the <i>Kuromoji</i> morphological engine that Oxygen XML Developer uses for indexing Japanese content in the WebHelp pages. This indexer does not come bundled with Oxygen XML Developer or the Oxygen XML WebHelp plugin. To use it, you need to download it from http://mvnrepository.com/artifact/org.apache.lucene/lucene-analyzers-kuromoji/4.0.0 and add it in the <code>DITA_OT_DIR/plugins/com.oxygenxml.webhelp/</code> directory.
webhelp.search.ranking	If this parameter is set to <code>false</code> then the 5-star rating mechanism is no longer included in the search results that are displayed on the Search tab (default setting is <code>true</code>).
webhelp.show.changes.and.comments	When set to <code>yes</code> , user comments, replies to comments, and tracked changes are published in the WebHelp output. The default value is <code>no</code> .
webhelp.sitemap.base.url	Base URL for all the <code>loc</code> elements in the generated <code>sitemap.xml</code> file. The value of a <code>loc</code> element is computed as the relative file path from the <code>href</code> attribute of the <code>topicref</code> element from the DITA map, appended to this base URL value. The <code>loc</code> element is mandatory in <code>sitemap.xml</code> . If you leave this parameter set to its default empty value, then the <code>sitemap.xml</code> file is not generated.
webhelp.sitemap.change.frequency	The value of the <code>changefreq</code> element in the generated <code>sitemap.xml</code> file. The <code>changefreq</code> element is optional in <code>sitemap.xml</code> . If you leave this parameter set to its default empty value, then the <code>changefreq</code> element is not added in <code>sitemap.xml</code> . Allowed values: <code><empty string></code> (default), <code>always</code> , <code>hourly</code> , <code>daily</code> , <code>weekly</code> , <code>monthly</code> , <code>yearly</code> , <code>never</code> .
webhelp.sitemap.priority	The value of the <code>priority</code> element in the generated <code>sitemap.xml</code> 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 <code>priority</code> element is optional in <code>sitemap.xml</code> . If you leave this parameter set to its default empty value, then the <code>priority</code> element is not added in <code>sitemap.xml</code> .

Parameters Specific to WebHelp Responsive Output

webhelp.fragment.after.body	Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the body in the output.
webhelp.fragment.after.logo_and_title	Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the logo and title in the output.
webhelp.fragment.after.main.page.search	Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the search field in the output.

webhelp.fragment.after.toc_or_tiles	Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the table of contents or tiles in the main page of the output.
webhelp.fragment.after.top_menu	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the top menu in the output.
webhelp.fragment.before.body	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the body in the output.
webhelp.fragment.before.logo_and_title	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the logo and title in the output.
webhelp.fragment.before.main.page.search	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the search field in the output.
webhelp.fragment.before.toc_or_tiles	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the table of contents or tiles in the main page of the output.
webhelp.fragment.before.top_menu	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the top menu in the output.
webhelp.fragment.footer	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed as the footer in the output
webhelp.fragment.head	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed as the header in the output.
webhelp.fragment.welcome	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed as a welcome message (or title) in the output.
webhelp.show.breadcrumb	Specifies if the breadcrumb component will be presented in the output. The default value is yes .
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 <i>tree</i> 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.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 top 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.

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 (or **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 Developer)

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 parameter has no impact on your edited DITA content. Allowed values: **true** or **false** (default is **false**).

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.japanese.dictionary	The file path of the dictionary that will be used by the <i>Kuromoji</i> morphological engine that Oxygen XML Developer uses for indexing Japanese content in the WebHelp pages. This indexer does not come bundled with Oxygen XML Developer or the Oxygen XML WebHelp plugin. To use it, you need to download it from http://mvnrepository.com/artifact/org.apache.lucene/lucene-analyzers-kuromoji/4.0.0 and add it in the <i>DITA_OT_DIR</i> /plugins/com.oxygenxml.webhelp/1 directory.
webhelp.search.ranking	If this parameter is set to <code>false</code> then the 5-star rating mechanism is no longer included in the search results that are displayed on the Search tab (default setting is <code>true</code>).
webhelp.show.changes.and.comments	When set to <code>yes</code> , user comments, replies to comments, and tracked changes are published in the WebHelp output. The default value is <code>no</code> .
webhelp.sitemap.base.url	Base URL for all the <code>loc</code> elements in the generated <code>sitemap.xml</code> file. The value of a <code>loc</code> element is computed as the relative file path from the <code>href</code> attribute of the <code>topicref</code> element from the DITA map, appended to this base URL value. The <code>loc</code> element is mandatory in <code>sitemap.xml</code> . If you leave this parameter set to its default empty value, then the <code>sitemap.xml</code> file is not generated.
webhelp.sitemap.change.frequency	The value of the <code>changefreq</code> element in the generated <code>sitemap.xml</code> file. The <code>changefreq</code> element is optional in <code>sitemap.xml</code> . If you leave this parameter set to its default empty value, then the <code>changefreq</code> element is not added in <code>sitemap.xml</code> . Allowed values: <code><empty string></code> (default), <code>always</code> , <code>hourly</code> , <code>daily</code> , <code>weekly</code> , <code>monthly</code> , <code>yearly</code> , <code>never</code> .
webhelp.sitemap.priority	The value of the <code>priority</code> element in the generated <code>sitemap.xml</code> 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 <code>priority</code> element is optional in <code>sitemap.xml</code> . If you leave this parameter set to its default empty value, then the <code>priority</code> element is not added in <code>sitemap.xml</code> .

Parameters Specific to WebHelp Responsive with Feedback Output

webhelp.fragment.after.body	Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the body in the output.
webhelp.fragment.after.logo_and_title	Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the logo and title in the output.
webhelp.fragment.after.main.page.search	Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the search field in the output.
webhelp.fragment.after.toc_or_tiles	Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the table of contents or tiles in the main page of the output.
webhelp.fragment.after.top_menu	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the top menu in the output.
webhelp.fragment.before.body	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the body in the output.
webhelp.fragment.before.logo_and_title	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the logo and title in the output.

<code>webhelp.fragment.before.main.page.search</code>	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the search field in the output.
<code>webhelp.fragment.before.toc_or_tiles</code>	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the table of contents or tiles in the main page of the output.
<code>webhelp.fragment.before.top_menu</code>	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the top menu in the output.
<code>webhelp.fragment.footer</code>	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed as the footer in the output.
<code>webhelp.fragment.head</code>	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed as the header in the output.
<code>webhelp.fragment.welcome</code>	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed as a welcome message (or title) in the output.
<code>webhelp.show.breadcrumb</code>	Specifies if the breadcrumb component will be presented in the output. The default value is yes .
<code>webhelp.show.indexterms.link</code>	Specifies if an icon that links to the index will be presented in the output. The default value is yes .
<code>webhelp.show.main.page.tiles</code>	Specifies if the tiles component will be presented in the main page of the output. For a <i>tree</i> style layout, this parameter should be set to no .
<code>webhelp.show.main.page.toc</code>	Specifies if the table of contents will be presented in the main page of the output. The default value is yes .
<code>webhelp.show.navigation.links</code>	Specifies if navigation links will be presented in the output. The default value is yes .
<code>webhelp.show.print.link</code>	Specifies if a print link or icon will be presented within each topic in the output. The default value is yes .
<code>webhelp.show.side.toc</code>	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 .
<code>webhelp.show.top.menu</code>	Specifies if a menu will be presented at the topic of the main page in the output. The default value is yes .
<code>webhelp.top.menu.depth</code>	Specifies the maximum depth level of the topics that will be included in the top menu. The default value is 2.

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 (or not). `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 Developer)

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 parameter has no impact on your edited DITA content. Allowed values: `true` or `false` (default is `false`).

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 page. 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 XML; 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 excerpt 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 are `yes`, `no`. If set to `no`, no footer is added to the WebHelp pages. If set to `yes` and `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 Developer footer is inserted in each WebHelp page.

webhelp.google.search.results

URL value 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 recommended to set the `linkTarget` attribute to `frm` for frameless (*iframe*) version of WebHelp or to `contentWindow` for the frameset version of WebHelp. The default value for this attribute is `_blank`, which means 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

Specifies the location of a well-formed XHTML file containing the Custom Search Engine script from Google. The value must be a URL.

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 head.

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.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Developer uses for indexing Japanese content in the WebHelp pages. This indexer does not come bundled with Oxygen XML Developer or the Oxygen XML WebHelp plugin. To use it, you need to download it from <http://mvnrepository.com/artifact/org.apache.lucene/lucene-analyzers-kuromoji/4.0.0> and add it in the `DITA_OT_DIR/plugins/com.oxygenxml.webhelp/` directory.

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 the `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.

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 <code>en-us</code> .
clean.output	Deletes all files from the output folder before the transformation is performed (only <code>no</code> and <code>yes</code> values are valid and the default value is <code>no</code>).
fix.external.refs.com.oxygenxml (Only supported when the DITA OT transformation process is started from Oxygen XML Developer)	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 parameter has no impact on your edited DITA content. Allowed values: <code>true</code> or <code>false</code> (default is <code>false</code>).
use.stemming	Controls whether or not you want to include stemming search algorithms into the published output (default setting is <code>false</code>).
webhelp.copyright	Adds a small copyright text that appears at the end of the <i>Table of Contents</i> 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 <i>favicon</i> in the WebHelp output.
webhelp.footer.file	Path to an XML file that includes the footer content for your WebHelp output page. 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 XML; each widget must be in separate <code>div</code> or <code>span</code> element, and the code for each <code>script</code> element is included in an XML comment (also, the start and end tags for the XML element).

comment must be on a separate line). The following code excerpt 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 are `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 Developer footer is inserted in each WebHelp page.

webhelp.google.search.results

URL value 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 recommended to set the `linkTarget` attribute to `frm` for frameless (*iframe*) version of WebHelp or to `contentWindow` for the frameset version of WebHelp. The default value for this attribute is `_blank`, which means 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

Specifies the location of a well-formed XHTML file containing the Custom Search Engine script from Google. The value must be a URL.

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.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Developer uses for indexing Japanese content in the WebHelp pages. This indexer does not come bundled with Oxygen XML Developer or the Oxygen XML WebHelp plugin. To use it, you need to download it from <http://mvnrepository.com/artifact/org.apache.lucene/lucene-analyzers-kuromoji/4.0.0> and add it in the `DITA_OT_DIR/plugins/com.oxygenxml.webhelp/` directory.

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 the `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.

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 http://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 <code>en-us</code> .
clean.output	Deletes all files from the output folder before the transformation is performed (or no and yes values are valid and the default value is <code>no</code>).
fix.external.refs.com.oxygenxml (Only supported when the DITA OT transformation process is started from Oxygen XML Developer)	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 parameter has no impact on your edited DITA content. Allowed values: <code>true</code> or <code>false</code> (default is <code>false</code>).
use.stemming	Controls whether or not you want to include stemming search algorithms into the published output (default setting is <code>false</code>).
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 <code><body></code> section of every WebHelp page.
webhelp.copyright	Adds a small copyright text that appears at the end of the <i>Table of Contents</i> 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.faviconThe 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 page. 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 XML. Each widget must be in a 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 excerpt 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 are **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 Developer footer is inserted in each WebHelp page.

webhelp.google.search.results

URL value 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 recommended to set the **linkTarget** attribute to **frm** for frameless (*iframe*) version of WebHelp or to **contentWindow** for the frameset version of WebHelp. The default value for this attribute is **_blank**, which means 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

Specifies the location of a well-formed XHTML file containing the Custom Search Engine script from Google. The value must be a URL.

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.japanese.dictionary

The file path of the dictionary that will be used by the *Kuromoji* morphological engine that Oxygen XML Developer uses for indexing Japanese content in the WebHelp pages. This indexer does not come bundled with Oxygen XML Developer or the Oxygen XML WebHelp plugin. To use it, you need to download it from <http://mvnrepository.com/artifact/org.apache.lucene/lucene-analyzers-kuromoji/4.0.0> and add it in the **DITA_OT_DIR/plugins/com.oxygenxml.webhelp/1.0** directory.

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 the 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 .

DITA Map to PDF WYSIWYG Transformation

Oxygen XML Developer comes bundled with a DITA OT plugin that converts DITA maps to PDF using a CSS layout processor. Oxygen XML Developer supports the following processors (not included in the Oxygen XML Developer installation kit):

- **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 Developer, the CSS files located in the **\${frameworks}** directory take precedence.

Creating the Transformation Scenario

To create an experimental 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 - Experimental**.
3. When applied, this new transformation scenario uses the currently selected CSS files for the opened topic files. These CSS files can be selected from the **Styles** drop-down menu from the toolbar.



Important: The author could open the map in the editor and change its style, but this is ignored in the publishing stage. Since authors usually edit topics instead of the map, Oxygen XML Developer uses the styles selected in the opened topics.

4. 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.
 - **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**.

Customizing the Styles (for Output and Editing)

If you need to change the styles in the associated CSS, make sure you install Oxygen XML Developer in a folder in which you have full read and write privileges (for instance, your user home directory). This is due to the fact that all the installed files are usually read-only (for instance, in Windows, Oxygen XML Developer is installed in the **Program Files** folder where the users do not have change rights).



Note: This experimental transformation scenario also allows you to present colored highlights in the PDF output.

Compiled HTML Help (CHM) Output Format

To perform a **Compiled HTML Help (CHM)** transformation Oxygen XML Developer needs Microsoft **HTML Help Workshop** to be installed on your computer. Oxygen XML Developer 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 Developer uses the `htmlhelp.locale` parameter to correctly display specific characters of different languages in the output of the **Compiled HTML Help (CHM)** transformation. The **Compiled HTML Help (CHM)** default scenario that comes bundled with Oxygen XML Developer has the `htmlhelp.locale` parameter set to `en-US`.

The default value of the `htmlhelp.locale` is `en-US`. To customize this parameter, go to **Configure**

Transformation Scenarios and click the **Edit** button. In the parameter tab search for the `htmlhelp.locale` parameter and change its value to the desired language tag.

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 Developer requires KindleGen to generate Kindle output from DITA maps. To install KindleGen for use by Oxygen XML Developer, 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 Developer and open a DITA map in the **DITA Maps Manager** view.
4. In the **DITA Maps Manager** view, open the **Configure Transformation Scenario(s)** dialog box.
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.

Migrating OOXML Documents to DITA

Oxygen XML Developer integrates the entire DITA for Publishers plugins suite that allows you to migrate content from Open Office XML documents to DITA.

You can use any of the following methods to migrate such documents to DITA:

- Open an OOXML document in Oxygen XML Developer. The document is opened in the **Archive Browser** view.
- From the **Archive Browser**, open document.xml.



Note: `document.xml` holds the content of the document.

- Click  **Configure Transformation Scenario(s)** on the toolbar and apply the **DOCX DITA** scenario. If you encounter any issues with the transformation, click the link below for further details about the Word to DITA Transformation Framework.

Related information

<http://dita4publishers.sourceforge.net/>

XHTML Transformation Scenarios

The following default transformation scenarios are available for XHTML:

- XHTML to DITA concept** - Converts an XHTML document to a DITA concept document.
- XHTML to DITA reference** - Converts an XHTML document to a DITA reference document.
- XHTML to DITA task** - Converts an XHTML document to a DITA task document.
- XHTML to DITA topic** - Converts an XHTML document to a DITA topic document.

Related information

[Editing a Transformation Scenario](#) on page 596

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

TEI ODD Transformation Scenarios

The following default transformations are available:

- TEI ODD XHTML** - Transforms a TEI ODD document into an XHTML document
- TEI ODD PDF** - Transforms a TEI ODD document into a PDF document using the Apache FOP engine
- TEI ODD EPUB** - Transforms a TEI ODD document into an EPUB document
- TEI ODD DOCX** - Transforms a TEI ODD document into a DOCX document
- TEI ODD ODT** - Transforms a TEI ODD document into an ODT document
- TEI ODD RelaxNG XML** - Transforms a TEI ODD document into a RelaxNG XML document
- TEI ODD to DTD** - Transforms a TEI ODD document into a DTD document
- TEI ODD to XML Schema** - Transforms a TEI ODD document into an XML Schema document
- TEI ODD to RelaxNG Compact** - Transforms a TEI ODD document into an RelaxNG Compact document

Related information

[Editing a Transformation Scenario](#) on page 596

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

TEI P4 Transformation Scenarios

The following default transformations are available:

- TEI HTML** - Transforms a TEI document into an HTML document.
- TEI P4 - TEI P5 Conversion** - Convert a TEI P4 document into a TEI P5 document.
- TEI PDF** - Transforms a TEI document into a PDF document using the Apache FOP engine.

Related information

[Editing a Transformation Scenario](#) on page 596

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

TEI P5 Transformation Scenarios

The following default transformations are available:

- TEI P5 XHTML** - transforms a TEI P5 document into an XHTML document.
- TEI P5 PDF** - transforms a TEI P5 document into a PDF document using the Apache FOP engine.

- **TEI EPUB** - transforms a TEI P5 document into an EPUB output. The EPUB output will contain any images referenced in the TEI XML document.
- **TEI DOCX** - transforms a TEI P5 document into a DOCX (OOXML) document. The DOCX document will contain any images referenced in the TEI XML document.
- **TEI ODT** - transforms a TEI P5 document into an ODT (ODF) document. The ODT document will contain any images referenced in the TEI XML document.

Related information

[Editing a Transformation Scenario](#) on page 596

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

JATS Transformation Scenarios

The following default transformation scenario is available for JATS documents:

- **JATS Preview (simple HTML)** - Converts a JATS document to a simple HTML document.

Related information

[Editing a Transformation Scenario](#) on page 596

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

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 Developer 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:

- **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.
- **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.

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, 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, 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 enabled, the scenario applies the stylesheet specified explicitly in the XML document with the <code>xml-stylesheet</code> processing instruction. By default, this option is disabled 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 Developer 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 checked, the parameter value is evaluated as an XPath expression before starting the XSLT transformation.

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 <i>editor variable</i> can be inserted in the text box using the Insert Editor Variables button. If the Evaluate as XPath option is enabled, 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 <i>editor variable</i> can be inserted in the text box using the Insert Editor Variables button. If the Evaluate as XPath option is enabled, 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 enabled 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 154

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.

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 <i>editor variable</i> 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.

The advanced options for Saxon 9.6.0.7 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 9.6.0.7 configuration file that is executed for XSLT transformation and validation processes.
Debugger trace into XPath expressions (applies to debugging sessions)	Instructs the <i>XSLT Debugger</i> to <i>step into</i> 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 <code>document()</code> 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 <i>strip whitespaces</i> operation is handled. You can choose one of the following values: • All ("all") - Strips <i>all</i> whitespace text nodes from source documents before any further processing, regardless of any <code>xml:space</code> attributes in the source document. • Ignore ("ignorable") - Strips all <i>ignorable</i> whitespace text nodes from source documents before any further processing, regardless of any <code>xml:space</code> 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 <i>no</i> 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.
Initializer class	Equivalent to the <i>-init</i> Saxon command-line argument. The value is the name of a user-supplied class that implements the <code>net.sf.saxon.lib.Initializer</code> 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 9.6.0.7 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 9.6.0.7 processors.  Note: Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Developer 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 checked, 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 <code>http://[URL]</code>). It ensures that the stylesheet cannot call arbitrary Java methods and thus gain privileged access to resources on your machine.

The advanced options that are specific for Saxon 9.6.0.7 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 <code>document()</code> or similar functions. It can have the following values: • Schema validation ("strict") - This mode requires an XML Schema and enables parsing the source documents with strict schema-validation enabled. • Lax schema validation ("lax") - If an XML Schema is provided, this mode enables 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. Enabling 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 enable 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 .

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 enabled, Oxygen XML Developer automatically opens the result of the transformation in a system application associated with the file type of the result (for example, in Windows <i>PDF</i> files are often opened in <i>Acrobat Reader</i>). Note: To set the web browser that is used for displaying HTML/XHTML pages, open the Preferences dialog box (Options > Preferences), go to 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 enabled, 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 Developer 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 Developer 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 can only be selected if **Open in Browser/System Application** is disabled. If selected, Oxygen XML Developer 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 enabling this feature 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:

- **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.
- **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.

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, 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, 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 Developer 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 checked, the parameter value is evaluated as an XPath expression before starting the XQuery transformation.

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 <i>editor variable</i> can be inserted in the text box using the Insert Editor Variables button. If the Evaluate as XPath option is enabled, 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 <i>editor variable</i> can be inserted in the text box using the Insert Editor Variables button. If the Evaluate as XPath option is enabled, 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 enabled 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 154

XQuery Extensions

The **Extensions** button is used to specify the jars 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 9.6.0.7 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 <i>strip whitespaces</i> operation is handled. You can choose one of the following values: • All ("all") - Strips <i>all</i> whitespace text nodes from source documents before any further processing, regardless of any <code>xml:space</code> attributes in the source document. • Ignore ("ignorable") - Strips all <i>ignorable</i> whitespace text nodes from source documents before any further processing, regardless of any <code>xml:space</code> 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 <i>no</i> 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 enabled (default value), Saxon runs the XQuery transformation with the XQuery 3.0 support.
Initializer class	Equivalent to the <i>-init</i> Saxon command-line argument. The value is the name of a user-supplied class that implements the <code>net.sf.saxon.lib.Initializer</code> 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 9.6.0.7 Professional Edition (PE) and Enterprise Edition (EE) only:

Use a configuration file ("-config")	Sets a Saxon 9.6.0.7 configuration file that is used for XQuery transformation and validation scenarios.
Allow calls on extension functions ("-ext")	If checked, calls on external functions are allowed. Checking 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 9.6.0.7 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 <code>document()</code> or similar functions. It can have the following values: • Schema validation ("strict") - This mode requires an XML Schema and enables parsing the source documents with strict schema-validation enabled.

- **Lax schema validation ("lax")** - If an XML Schema is provided, this mode enables 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. Enabling 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 enable 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 checked, backup versions for any XML files updated with an XQuery Update are generated. This option is available when the Enable XQuery update option is enabled.

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	Enabling 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 enabled, Oxygen XML Developer 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, [open the Preferences dialog box \(Options > Preferences\)](#), go to **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 enabled, 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 Developer 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 Developer 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 can only be selected if **Open in Browser/System Application** is disabled. If selected, Oxygen XML Developer 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 enabling this feature 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 Developer 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.

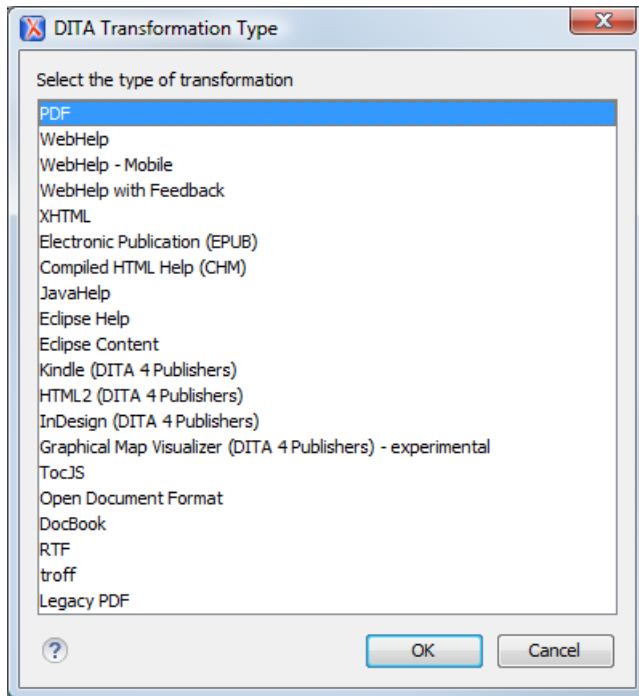


Figure 300: DITA Transformation Type Dialog Box

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:

- 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.
- 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.

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.

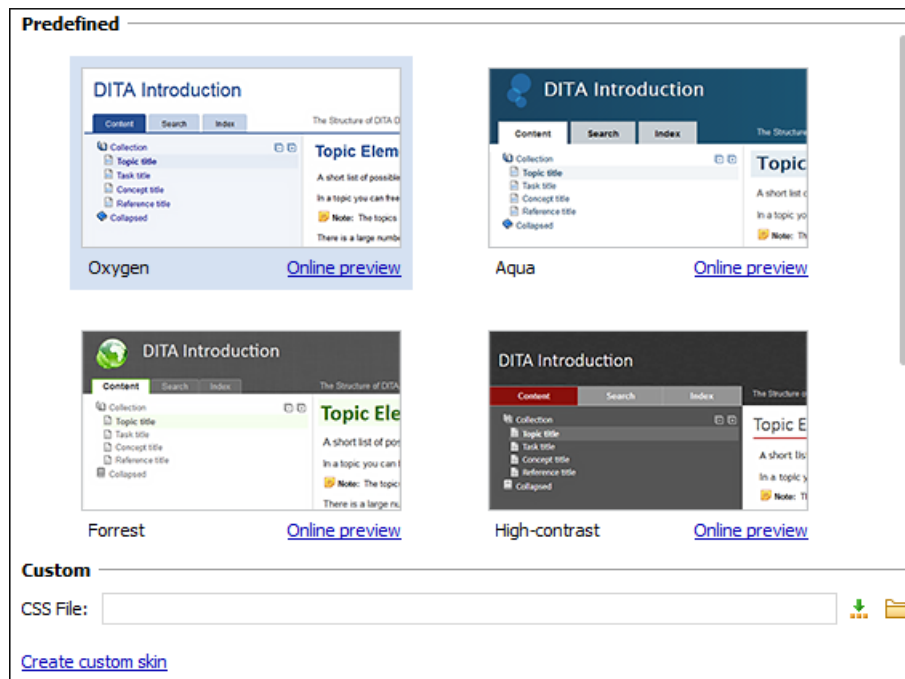


Figure 301: Skins Tab

The **Skins** tab includes the following sections:

Predefined Skins This section presents the predefined skins that are included in Oxygen XML Developer. 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 Developer 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](#).

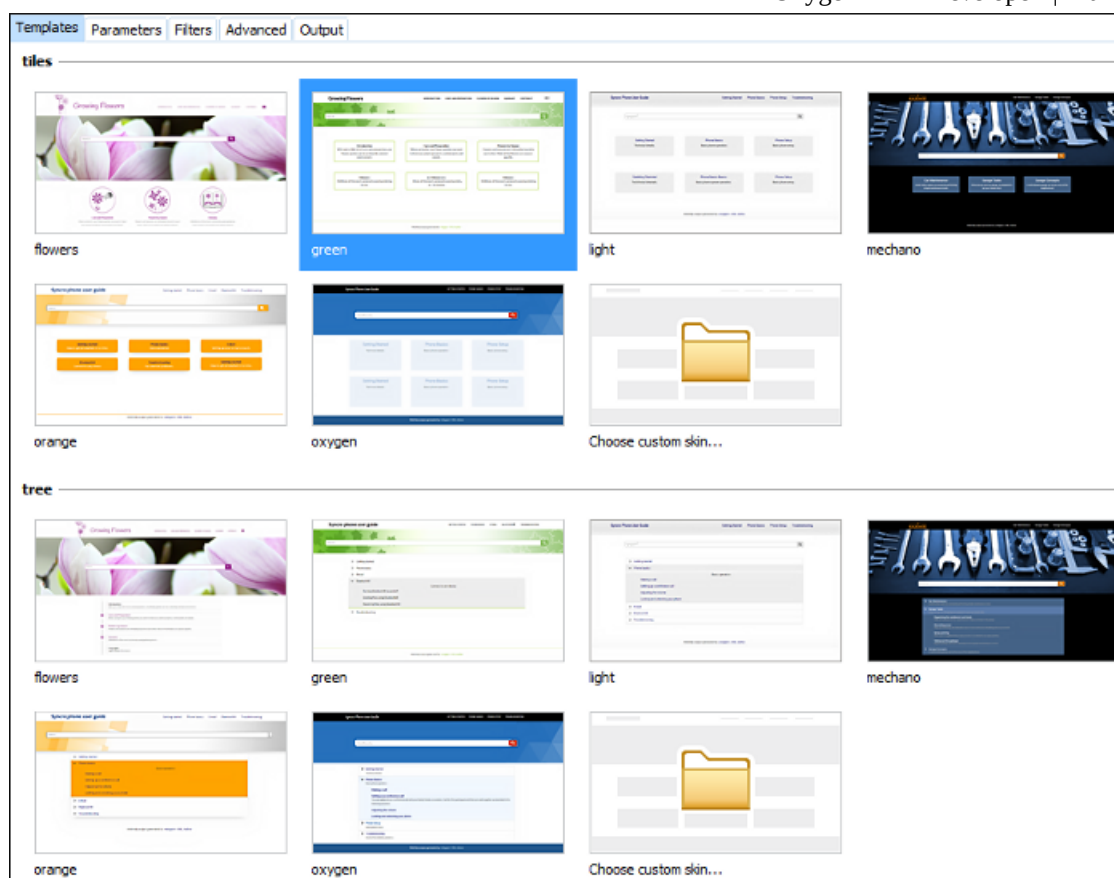


Figure 302: Templates Tab

The **Templates** tab comes by default with the following predefined collections of skins:

- Tiles** This section 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 section 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.

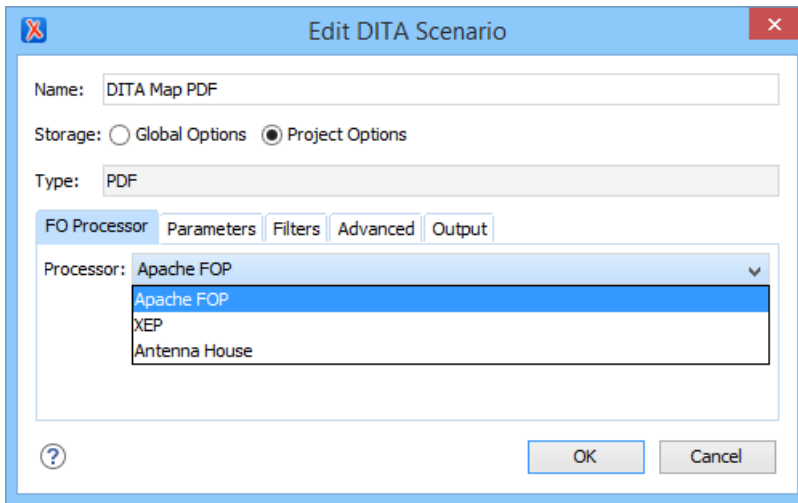


Figure 303: FO Processor Configuration Tab

You can choose one of the following processors:

Apache FOP The default processor that comes bundled with Oxygen XML Developer.

XEP The [RenderX XEP](#) processor. If XEP is already installed, Oxygen XML Developer 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 Developer installation directory.

Antenna House The [Antenna House](#) (AH Formatter) processor. If Antenna House is already installed, Oxygen XML Developer displays 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 Developer 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 115

[XSL-FO Processors](#) on page 607

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 enabled 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.

You can choose one of the following options to define filters:

Use DITAVAL file	If you already have a <i>DITAVAL</i> 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, the history drop-down menu, the  Insert Editor Variables button, or the  Browse button. You can find out more about constructing a <i>DITAVAL</i> file in the DITA OT Documentation.  Attention: If a filter file is specified in the <code>args.filter</code> parameter (in the Parameters tab), that file takes precedence over a <i>DITAVAL</i> file specified here.
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.

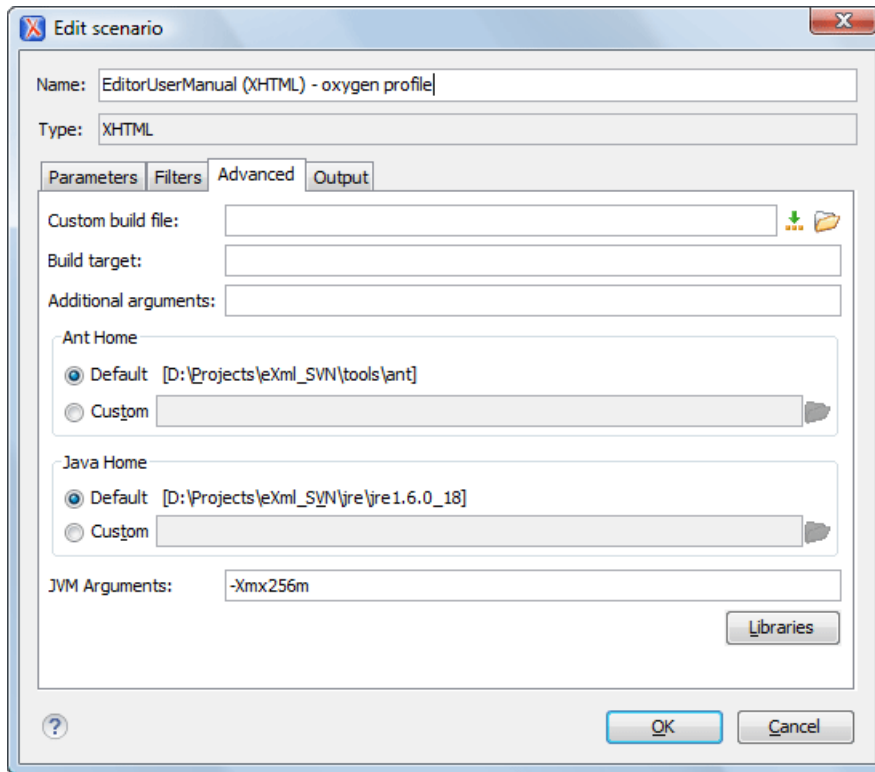


Figure 304: Advanced Settings Tab

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 <code>build.xml</code> file from the <code>dita.dir</code> 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 <code>init</code> target is used.
Additional arguments	You can specify additional command line arguments to be passed to the transformation (such as <code>-verbose</code>).
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 Developer.
	 Note: It may be possible that the used Java version is incompatible with the DITA Open Toolkit engine. For example, DITA OT 1.8.5 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 <code>-Xmx384m</code> , 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 <i>Out of Memory</i> error messages (OutOfMemoryError).

Libraries

By default, Oxygen XML Developer 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.

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.

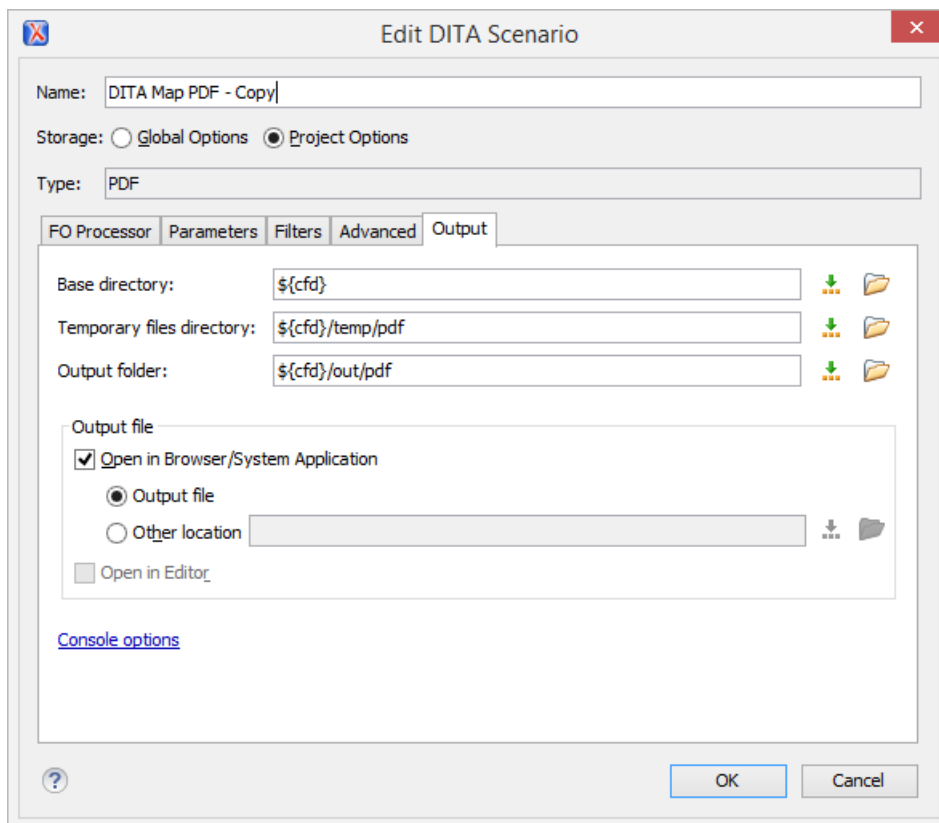


Figure 305: Output Settings Tab

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 enabled, Oxygen XML Developer 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, *open the Preferences dialog box (Options > Preferences)*, go to **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 enabled, 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).

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 your DITA documents by using the  **Validate** action from the  **Validation** toolbar drop-down menu, the **Document > Validate** menu, or from the **Validate** menu when invoking the contextual menu in the **Project** view.
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 **Transformation problems** message 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.

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 Developer 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:

- **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.
- **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.

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 Developer.
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 Developer 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.

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 enabled 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 366

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 <i>PDF</i> files are often opened in <i>Acrobat Reader</i>). • 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:

- **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.
- **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.

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, 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, 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 enabled, the scenario applies the stylesheet specified explicitly in the XML document with the <code>xml-stylesheet</code> processing instruction. By default, this option is disabled 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 Developer 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 checked, the parameter value is evaluated as an XPath expression before starting the XSLT transformation.

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 enabled, 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 enabled, 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 enabled 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 154

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.

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.

The advanced options for Saxon 9.6.0.7 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 9.6.0.7 configuration file that is executed for XSLT transformation and validation processes.
Debugger trace into XPath expressions (applies to debugging sessions)	Instructs the <i>XSLT Debugger</i> to <i>step into</i> 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 <code>document()</code> 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 <i>strip whitespaces</i> operation is handled. You can choose one of the following values: • All ("all") - Strips <i>all</i> whitespace text nodes from source documents before any further processing, regardless of any <code>xml:space</code> attributes in the source document. • Ignore ("ignorable") - Strips all <i>ignorable</i> whitespace text nodes from source documents before any further processing, regardless of any <code>xml:space</code> 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 <i>no</i> 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.
Initializer class	Equivalent to the <i>-init</i> Saxon command-line argument. The value is the name of a user-supplied class that implements the <code>net.sf.saxon.lib.Initializer</code> 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 9.6.0.7 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 9.6.0.7 processors.



Note: Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Developer 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 checked, 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.

The advanced options that are specific for Saxon 9.6.0.7 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 enables parsing the source documents with strict schema-validation enabled.
- **Lax schema validation ("lax")** - If an XML Schema is provided, this mode enables 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. Enabling 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 enable 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>.

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 enabled, Oxygen XML Developer automatically opens the result of the transformation in a system application associated with the file type of the result (for example, in Windows <i>PDF</i> files are often opened in <i>Acrobat Reader</i>).



Note: To set the web browser that is used for displaying HTML/XHTML pages, [open the Preferences dialog box \(Options > Preferences\)](#), go to **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 enabled, 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 Developer 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 Developer 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 can only be selected if **Open in Browser/System Application** is disabled. If selected, Oxygen XML Developer 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 enabling this feature 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:

- **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.
- **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.

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, 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 <i>Calabash</i> 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 <i>built-in editor variables</i> and <i>custom editor variables</i> 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 <i>built-in editor variables</i> and <i>custom editor variables</i> can be used to specify the URL.
Delete	Removes the selected port from the list. It is enabled 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 <i>built-in editor variables</i> and <i>custom editor variables</i> 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 <i>editor variable</i> 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 <i>editor variable</i> 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 enabled 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, the  <i>Insert Editor Variables</i> 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.

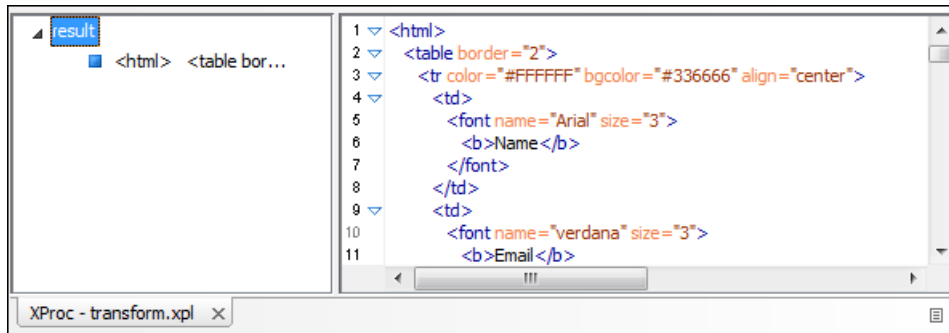


Figure 306: XProc Transformation Results View

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:

- **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.
- **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.

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, 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, 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 Developer 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 checked, the parameter value is evaluated as an XPath expression before starting the XQuery transformation.

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 <i>editor variable</i> can be inserted in the text box using the Insert Editor Variables button. If the Evaluate as XPath option is enabled, 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 <i>editor variable</i> can be inserted in the text box using the Insert Editor Variables button. If the Evaluate as XPath option is enabled, 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 enabled 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 154

XQuery Extensions

The **Extensions** button is used to specify the jars 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 9.6.0.7 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 <i>strip whitespaces</i> operation is handled. You can choose one of the following values: • All ("all") - Strips <i>all</i> whitespace text nodes from source documents before any further processing, regardless of any <code>xml:space</code> attributes in the source document. • Ignore ("ignorable") - Strips all <i>ignorable</i> whitespace text nodes from source documents before any further processing, regardless of any <code>xml:space</code> 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 <i>no</i> 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 enabled (default value), Saxon runs the XQuery transformation with the XQuery 3.0 support.
Initializer class	Equivalent to the <i>-init</i> Saxon command-line argument. The value is the name of a user-supplied class that implements the <code>net.sf.saxon.lib.Initializer</code> 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 9.6.0.7 Professional Edition (PE) and Enterprise Edition (EE) only:

Use a configuration file ("-config")	Sets a Saxon 9.6.0.7 configuration file that is used for XQuery transformation and validation scenarios.
Allow calls on extension functions ("-ext")	If checked, calls on external functions are allowed. Checking 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 9.6.0.7 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 <code>document()</code> or similar functions. It can have the following values: • Schema validation ("strict") - This mode requires an XML Schema and enables parsing the source documents with strict schema-validation enabled.

- **Lax schema validation ("lax")** - If an XML Schema is provided, this mode enables 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. Enabling 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 enable 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 checked, backup versions for any XML files updated with an XQuery Update are generated. This option is available when the Enable XQuery update option is enabled.

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	Enabling 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 enabled, Oxygen XML Developer 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, [open the Preferences dialog box \(Options > Preferences\)](#), go to **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 enabled, 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 Developer 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 Developer 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 can only be selected if **Open in Browser/System Application** is disabled. If selected, Oxygen XML Developer 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 enabling this feature 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: 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. 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.
SQL URL	Allows you to specify the URL of the SQL script. You can specify the path by using the text field, 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: You can edit transformation scenarios that are defined at project level only. To edit a transformation scenario that is associated with a predefined document type, [duplicate it and edit the duplicated scenario](#).

To configure an existing transformation scenario, follow these steps:

- 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.

- 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 560

[Duplicating a Transformation Scenario](#) on page 597

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

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 560

[Editing a Transformation Scenario](#) on page 596

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.

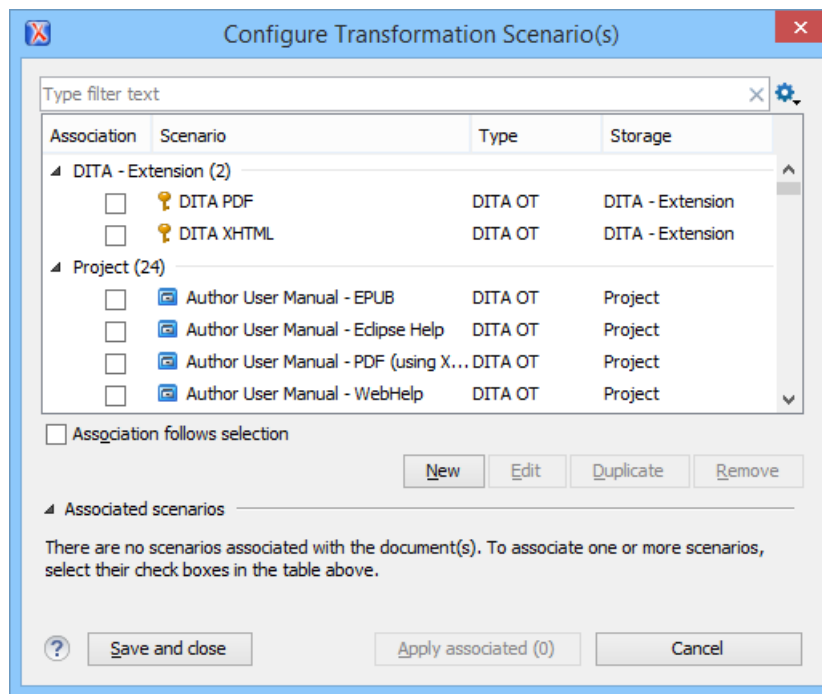


Figure 307: Configure Transformation Scenario(s) Dialog Box

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 Developer 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 Developer 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 Developer adds **imported** to the name of the imported scenario.



Export selected scenarios

Use this option to export selected scenarios individually. Oxygen XML Developer 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 check-boxes 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** - Displays the type of the transformation scenario. For further details about the types of transformation scenarios that are available in Oxygen XML Developer, see the [Transformation Types section](#).
- **Storage** - Displays where a transformation scenario is stored (the [Show Storage option](#) must be enabled.)

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	Enable 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 enabled, 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 Developer 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 **Global Options** or **Project 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 596

[Duplicating a Transformation Scenario](#) on page 597

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. Right-click the newly created logical folder and select **Transform >  Configure Transformation Scenario(s)** to select one or more transformation scenarios to be applied on all the files in the logical folder.



Note: These types of transformation scenarios must be configured with the current file (`${cf}`) or current file URL (`${currentFileURL}`) editor variables for the input file. This ensures that each file becomes the current file when its turn arrives in the batch transformation process. 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.

3. 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.
4. 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.

Sharing the 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 **Global Options** or **Project Options**.



Selecting **Global Options** stores the scenario in the global options that are stored in the user home directory.

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.

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 and Storing Options](#) on page 148

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**.

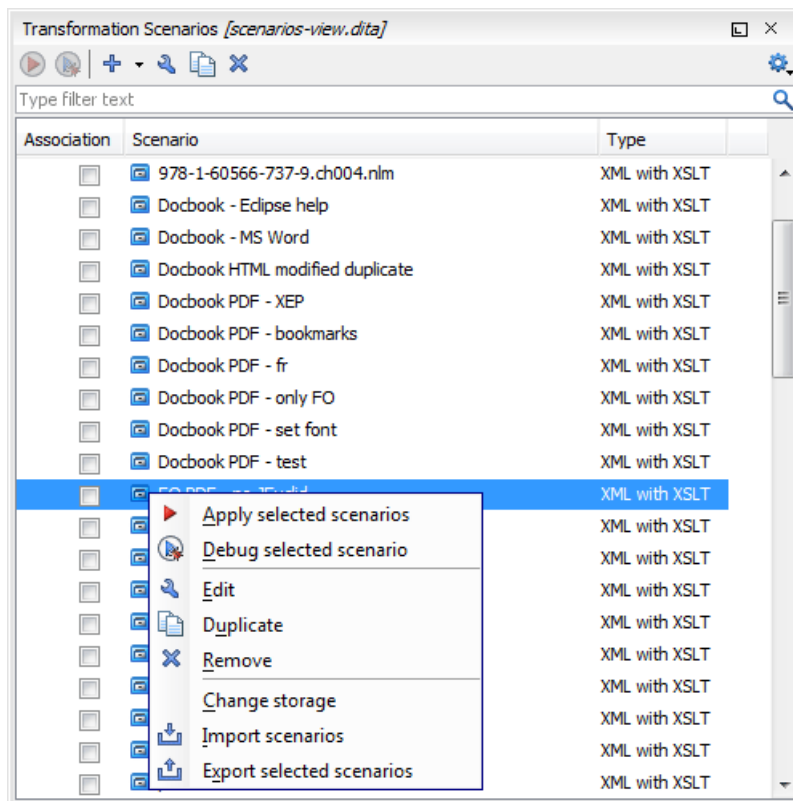


Figure 308: Transformation Scenarios view

Oxygen XML Developer supports multiple scenarios association. To associate multiple scenarios with a document, enable the check-boxes 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 Developer 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 <i>scenarios that you can create</i> . Oxygen XML Developer 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 <i>Transformation Types section</i> .										
 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 Developer 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 Developer adds imported to the name of the imported scenario.										
 Export selected scenarios	Use this option to export transformation and validation scenarios individually. Oxygen XML Developer 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): <table> <tr> <td>Show all scenarios</td><td>Select this option to display all the available scenarios, regardless of the document they are associated with.</td></tr> <tr> <td>Show only the scenarios available for the editor</td><td>Select this option to only display the scenarios that Oxygen XML Developer can apply for the current document type.</td></tr> <tr> <td>Show associated scenarios</td><td>Select this option to only display the scenarios associated with the document you are editing.</td></tr> <tr> <td>Change storage</td><td>Use this option to change the storage location of the selected scenario to Global Options or Project Options. You are also able to keep the original storage location and make a copy of the selected scenario in the new storage location.</td></tr> <tr> <td> Import scenarios</td><td> 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 Developer 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. </td></tr> </table>	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 Developer 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 Global Options or Project 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 Developer 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.
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 Developer 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 Global Options or Project 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 Developer 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 Developer adds `imported` to the name of the imported scenario.

Export selected scenarios

Use this option to export selected scenarios individually. Oxygen XML Developer 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 596

[Creating New Transformation Scenarios](#) on page 560

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.

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 Developer.

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 Developer.

Supported XSLT Processors

Oxygen XML Developer 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 9.6.0.7 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 9.6.0.7 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 Developer. 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 Developer 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 Developer 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 9.6.0.7 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 Developer.

- **Saxon-CE (Client Edition)** is Saxonica's implementation of XSLT 2.0 for use on web browsers. Oxygen XML Developer 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 Developer 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 Developer is 2.7.6 and the `Libxslt` version is 1.1.26.

Oxygen XML Developer 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 Developer 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 Developer 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 Developer performs XSLT transformations and validations using .NET Framework's XSLT implementation (`System.Xml.Xsl.XslTransform` class) through the **nxslt** command line utility. The **nxslt** version included in Oxygen XML Developer 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

You can configure and run XSLT and XQuery transformations with processors other than *the ones which come with the Oxygen XML Developer distribution*.



Note:

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 Developer 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 119

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 9.6.0.7** - <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 Developer.

Built-in XSL-FO Processor

The Oxygen XML Developer 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.

- Open a terminal.
- Change the working directory to the Oxygen XML Developer install directory.
- Create the following script file in the Oxygen XML Developer 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 Developer 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 ttfcConvert.sh` from the command line. On Windows, run the command `ttfcConvert.bat` from the command line or double-click the file `ttfcConvert.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>
      <font>
        <font metrics-url="Arialuni.xml" kerning="yes"
              embed-url="file:/Library/Fonts/Arialuni.ttf">
          <font-triplet name="Arialuni" style="normal"
                    weight="normal"/>
        </font>
      </font>
    </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">
  ...
  <font>
    <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>
  </font>
  ...
</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.

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`.

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`.

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:

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 Developer.

Adding Support for PDF Images

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 Developer.

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 Developer 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.

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 Developer 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.

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.

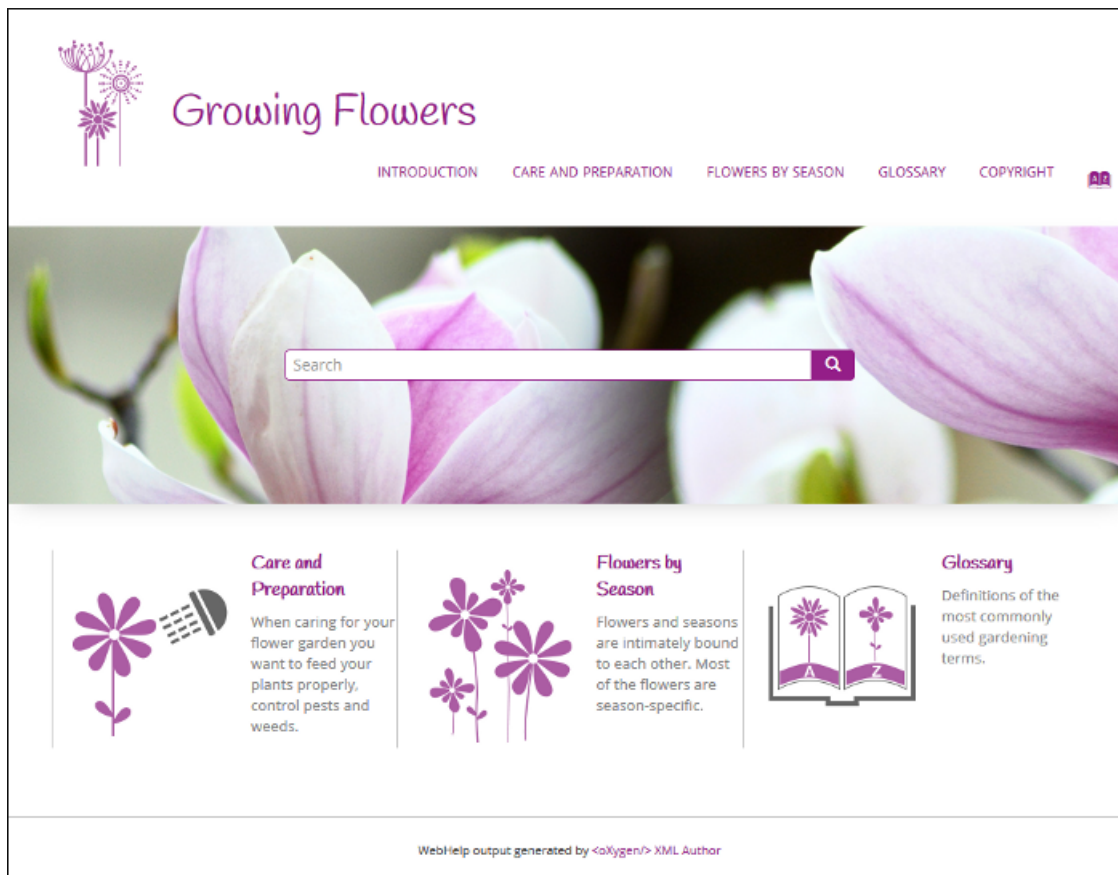


Figure 309: WebHelp Responsive Output on a Normal Screen

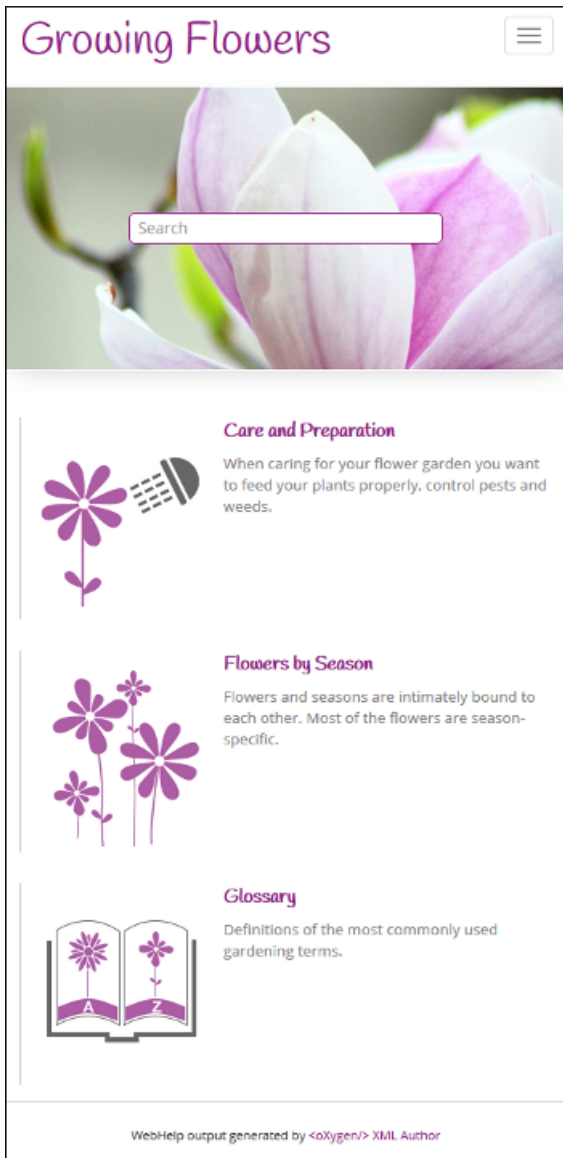


Figure 310: WebHelp Responsive Output on a Narrow Screen

Search Feature

When you enter search terms in the **Search** field, the results are displayed in the results page. When you click on a result, the topic is opened in the main pane and the search results are highlighted.

The **Search** feature is also enhanced with a rating mechanism that computes scores for every page that matches the search criteria. These scores are then translated into a 5-star rating scheme. The search results are sorted depending on the following:

- 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 phrase is included in a meta keyword
 - The search phrase is in the title of the page
 - The search phrase is in bold text in a paragraph
 - The search phrase is in normal text in a paragraph

Rules that are applied during a search include:

- Boolean searches are 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*).
- Do not use quotes to perform an exact search for multiple word expressions (an expression such as "*grow flowers*", returns no results since it searches for two separate words).
- `indexterm` and `keywords` DITA elements are an effective way to increase the ranking of a page (for example, content inside `keywords` elements weighs twice as much as content inside 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.

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 the [Editing Scoring Values of Tag Elements in Search Results](#) on page 638 topic.

This output format is compatible with the most recent versions of the following common browsers:

- Internet Explorer (IE 8 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 Developer 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.

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.

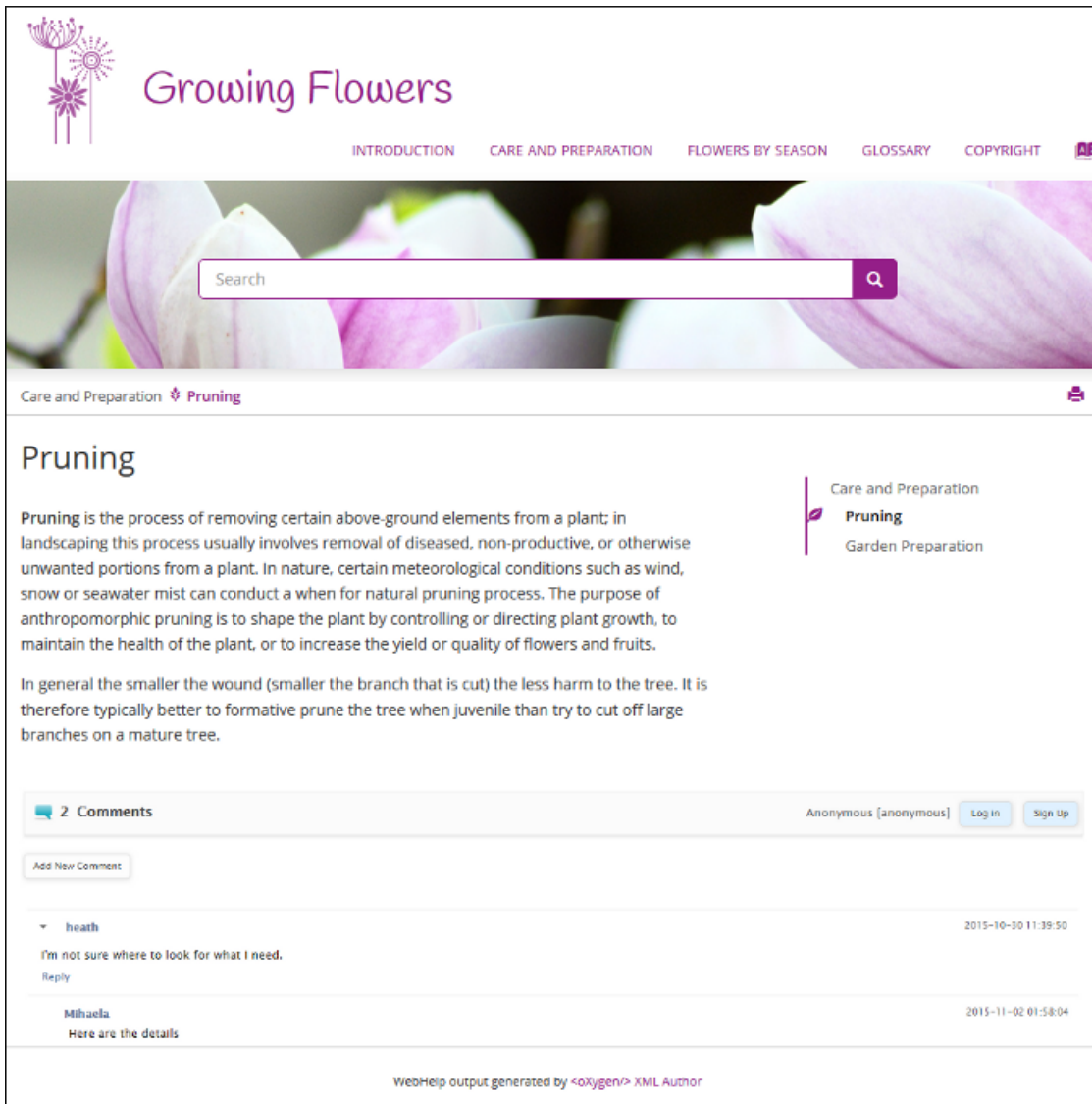


Figure 311: WebHelp Output

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 the [Managing Users and Comments in a WebHelp Responsive with Feedback System](#) on page 618 topic.

Search Feature

When you enter search terms in the **Search** field, the results are displayed in the results page. When you click on a result, the topic is opened in the main pane and the search results are highlighted.

The **Search** feature is also enhanced with a rating mechanism that computes scores for every page that matches the search criteria. These scores are then translated into a 5-star rating scheme. The search results are sorted depending on the following:

- 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 phrase is included in a meta keyword
 - The search phrase is in the title of the page
 - The search phrase is in bold text in a paragraph
 - The search phrase is in normal text in a paragraph

Rules that are applied during a search include:

- Boolean searches are 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*).
- Do not use quotes to perform an exact search for multiple word expressions (an expression such as "*grow flowers*", returns no results since it searches for two separate words).
- `indexterm` and `keywords` DITA elements are an effective way to increase the ranking of a page (for example, content inside `keywords` elements weighs twice as much as content inside 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.

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 the [Editing Scoring Values of Tag Elements in Search Results](#) on page 638 topic.

This output format is compatible with the most recent versions of the following common browsers:

- Internet Explorer (IE 8 or newer)
- Chrome
- Firefox
- Safari
- Opera

Deploying the WebHelp Responsive with Feedback System

System Requirements

The *WebHelp Responsive with Feedback* system of Oxygen XML Developer 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, Internet Explorer, Safari, Opera.

Create WebHelp Responsive with Feedback Database

The *WebHelp Responsive 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.

For 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 Responsive with Feedback Output

If you have a web server configured with PHP and MySQL, you can deploy the *WebHelp Responsive 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 Responsive with Feedback Database section](#) to fill-in the appropriate text boxes.



Warning: Checking 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 Responsive 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, check the **Enable LDAP Authentication** 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 checked, 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 Responsive 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, enable 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 Responsive 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 Responsive with Feedback* system main page.

Testing Your WebHelp Responsive 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 Responsive 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 Responsive 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 WebHelp Responsive with Feedback System](#) on page 618

Refreshing the Content of a WebHelp Responsive with Feedback System

It is common to update the content of an existing installation of a *WebHelp Responsive 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 Responsive 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 Responsive with Feedback* system installation directory, overwriting the existing content.

Managing Users and Comments in a WebHelp Responsive with Feedback System

When you installed the *WebHelp Responsive with Feedback* system the first time (assuming the [Create new database structure option](#) was enabled), 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 Responsive 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.

User Name	Name	Level	Company	E-Mail	Date	WebHelp Notification	Reply Notification	Page Notification	Status	admin.user.type
admin	Administrator	admin	NA	admin@sync.ro	2015-08-04 08:56:55	yes	no	no	validated	Local User
user	user	user	noCompany	user@sync.ro	2015-08-04 08:57:45	no	yes	yes	validated	Local User

Figure 312: 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 enabled, the user receives notifications when comments are posted anywhere in your feedback-enabled WebHelp system.
Reply Notification	When enabled, the user receives notifications when comments are posted as a reply to one of their comments.
Page Notification	When enabled, 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 pages and other additional resources (such as images, CSS, and JavaScript files). Each HTML template file contains 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.

The section provides information about the four types of template pages, the various template components, and the required template directory structure.

WebHelp Responsive Template Pages

The HTML pages that comprise the output of a WebHelp Responsive system are obtained after processing HTML template pages.

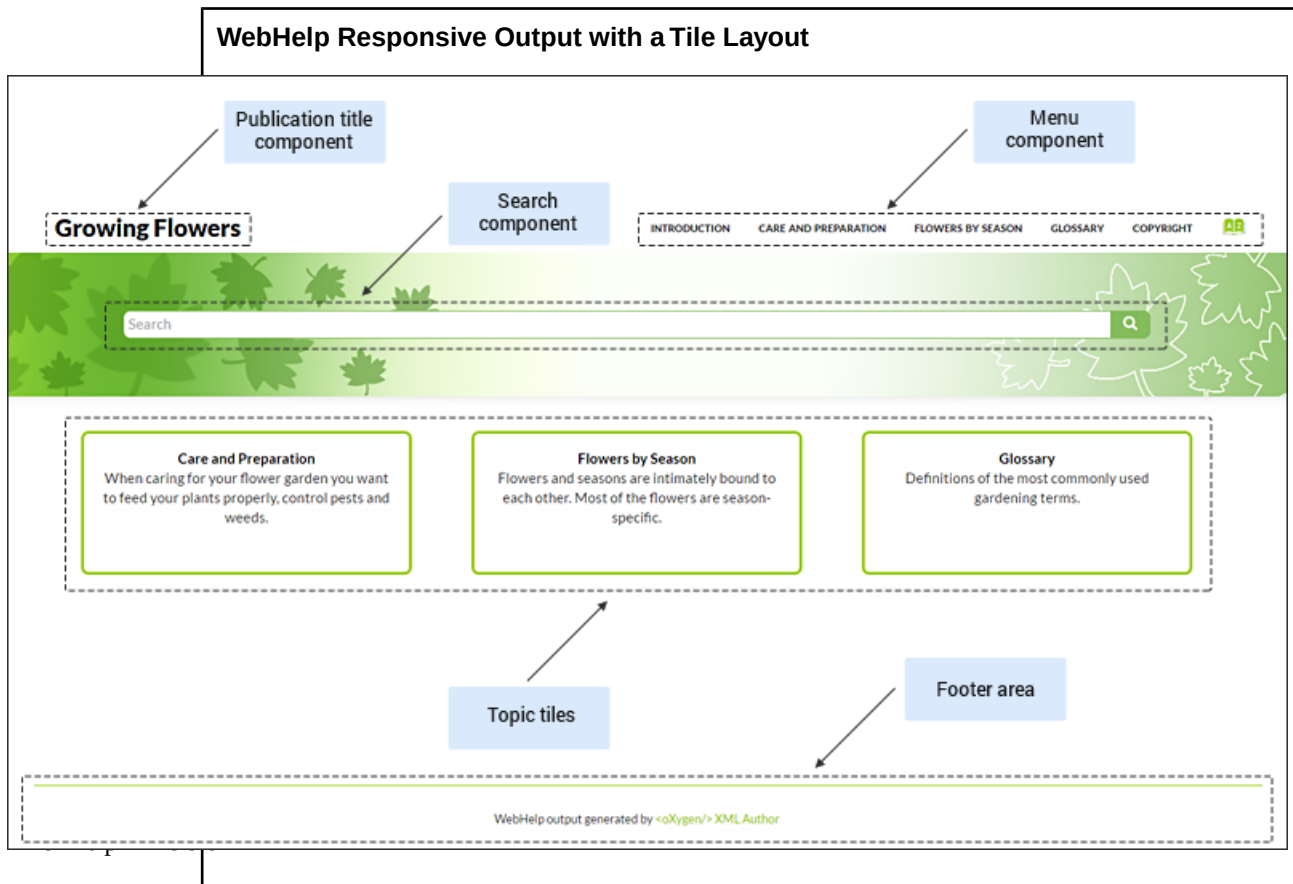
There are four types of template pages:

1. **Main page template** - Used to produce the home page of the WebHelp Responsive output.
2. **Topic template** - Used to generate the HTML pages associated with a topic.
3. **Search results template** - Used to generate the HTML page that presents the search results.
4. **Index terms template** - Used when generating the HTML page that presents the documentation index.

WebHelp Responsive Main Page Template

This template is used to generate the home page of the WebHelp Responsive output.

The main function of the home page is to display links that help you easily reach any of the top level topics of the publication. These links can be rendered either in a *tile* or *tree*-like layout.



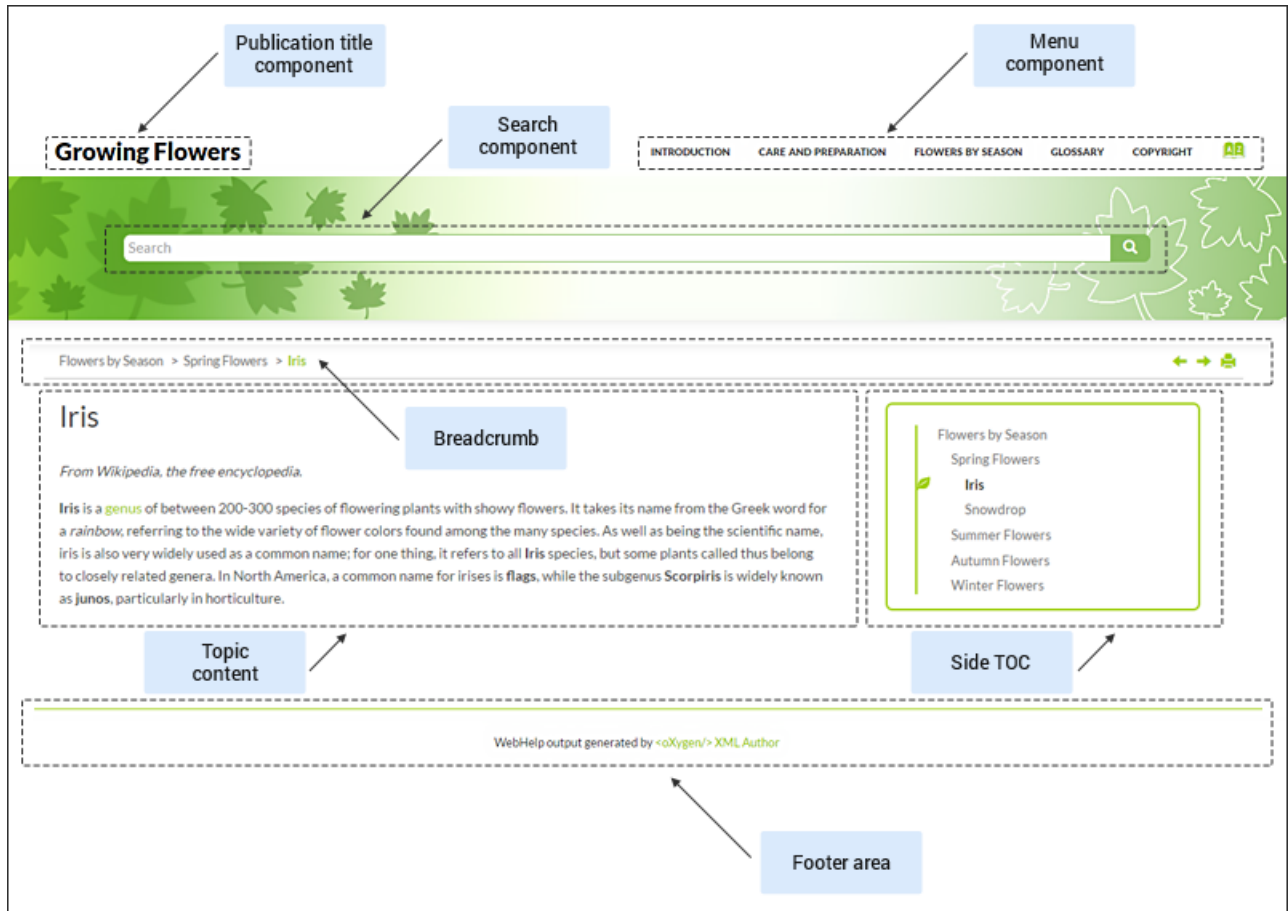
The components that can be referenced from the main page are:

- [Publication Title](#)
- [Publication Logo](#)
- [Search Input](#)
- [Print Link](#)
- [Main Page Menu](#)
- [Main Page Topic Tiles](#)
- [Main Page Table of Contents](#)
- [Index Terms Link](#)

WebHelp Responsive Topic Template

This template is used when generating HTML files for each topic. Basically, the HTML produced for a topic consists of the topic content which is wrapped by additional components such as: *application title*, *navigation breadcrumb*, or a *side TOC*.

A sample HTML topic is:



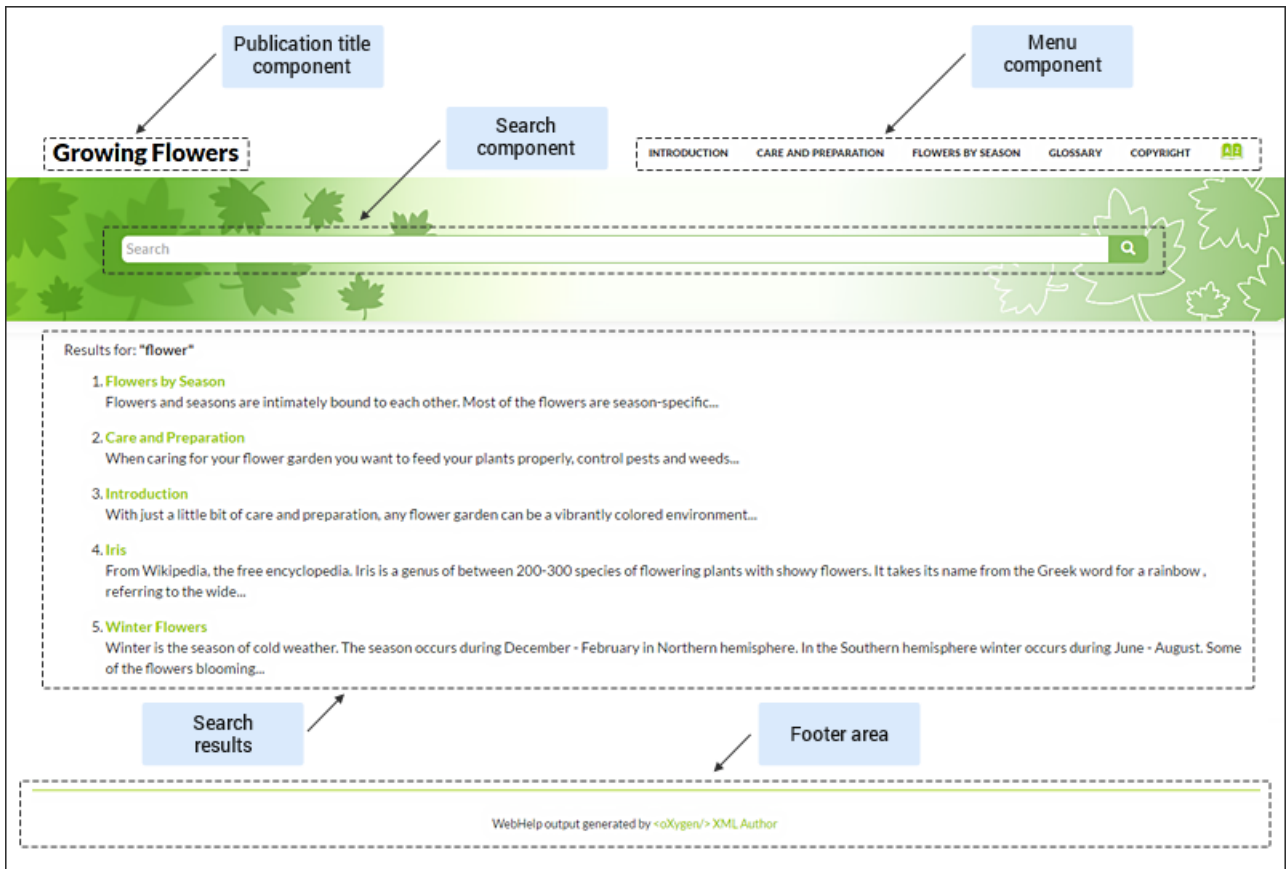
The components that can be referenced from the main page are:

- [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](#)

WebHelp Responsive Search Results Template

This template is used when generating HTML page used to present search results. This template should have a reference to the component that specifies where the search results will be presented.

A sample HTML for search page is:



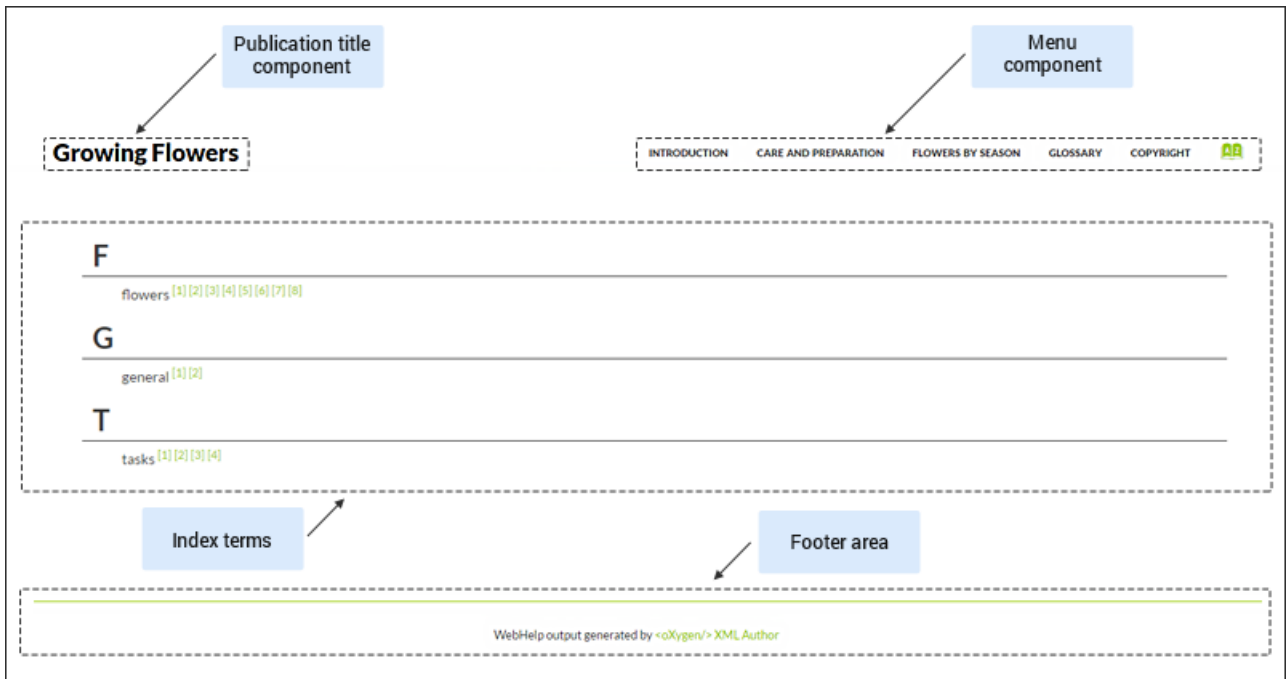
The components that can be referenced from the search page are:

- [Publication Title](#)
- [Publication Logo](#)
- [Search Input](#)
- [Print Link](#)
- [Main Page Menu](#)
- [Index Terms Link](#)

WebHelp Responsive Index Terms Template

This template is used when generating the page used to present index terms. This template should have a reference to the component that specifies where the index terms will be presented.

A sample index term page is:



The components that can be used inside the search results page are:

- [Publication Title](#)
- [Publication Logo](#)
- [Search Input](#)
- [Print Link](#)
- [Main Page Menu](#)
- [Index Terms Link](#)

WebHelp Template Components

A WebHelp Responsive template component adds dynamics to the WebHelp template page. It is expanded differently depending on the context where it is placed or the currently transformed DITA Map.

Some WebHelp Responsive template components could be used in all the template pages and some of them could be used in all pages. For instance, the *publication title* could be used in all pages, but *topic navigation breadcrumb* could be used only when expanding the WebHelp Responsive topic template.

To output such a component you have to refer a specific element in template pages. All the elements associated with a template components should belong to the `http://www.oxygenxml.com/webhelp/components` namespace.

Components can be classified depending on where they can be used:

Publication Title [webhelp_publication_title]

This component is bound to the publication title and it can be used in all template pages.

To output this component in the WebHelp Responsive output you have to specify the element `webhelp_publication_title` in a template page. This component can be specified in all template pages.

Sample

If you specify the next element in a template page:

```
<whc:webhelp_publication_title xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

you will find in the output an element with class `webhelp_publication_title`.

Publication Logo [webhelp_logo]

This component can be used to emit a logo image in WebHelp Responsive output. It will be processed only when a logo image is specified through the `webhelp.logo.image` transformation parameter.

Additionally, you can set the `webhelp.logo.image.target.url` parameter to generate a link to an URL when you hover the logo image. If this parameter is not set, then a link to the home page will be automatically generated.

The element associated with this component is `webhelp_logo` and it can be specified in all template pages.

Sample

If you specify the next element in a template page:

```
<whc:webhelp_logo xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

and the transformation parameters:

```
$webhelp.logo.image = logo.png  
$webhelp.logo.image.target.url = http://www.oxygenxml.com
```

you will find in the output an element with `webhelp_logo` class.

Search Input [webhelp_search_input]

This component is used to emit the input widget associated with search function. The element associated with this component is `webhelp_search_input` and it can be specified in all template pages.

Sample

If you specify the next element in a template page:

```
<whc:webhelp_search_input xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

you will find in the output an element with `webhelp_search_input` class.

Search Results [webhelp_search_results]

This component can be used in the [WebHelp search template page](#) as a placeholder to signal where the search results will be presented.

The element associated with this component is:

```
<whc:webhelp_search_results xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

and it will generate in the output an element with the `webhelp_search_results` class.

Topic Breadcrumb [webhelp_breadcrumb]

This component can be used to display the path in the documentation to the current topic. It can be used only in the [WebHelp Responsive Topic Template](#).

Sample

If you specify the next element in the WebHelp topic template page:

```
<whc:webhelp_breadcrumb xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

you will find in the output an element with `webhelp_breadcrumb` class. This element will contain a list with items to the current topic. The first item in the list has a link to the main page with class `home`. The last item in the list corresponds to the current topic and has set the `active` class.

Navigational Links [webhelp_navigation_links]

This component can be used to generate in the documentation links to the next and previous topics. It is available only in the [WebHelp Responsive Topic Template](#).

Sample

If you specify the next element in the WebHelp topic template page:

```
<whc:webhelp_navigation_links xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

you will find in the output an element with `webhelp_navigation_links` class. This element will contain the links to the next and previous topics.

Topic Content [webhelp_topic_content]

This component can be used in the [WebHelp Responsive Topic Template](#) to generate the raw content of the topic (without any side components like breadcrumb or side TOC). For DITA, it represent the content of the HTML files as they are produced by DITA-OT processor.

The element associated with this WebHelp template component is

```
<whc:webhelp_topic_content xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

It will generate in the output an element with `webhelp_topic_content` class.

Topic Side TOC [webhelp_side_toc]

This component can be used in the [WebHelp Responsive Topic Template](#) to generate a mini TOC for the current topic. The mini TOC will contain links to the current topic's children, its siblings and all of its ancestors.

The element associated with this WebHelp template component is

```
<whc:webhelp_side_toc xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

It will generate in the output an element with `webhelp_side_toc` class and containing links to the topics that are close to the current topic.

Topic Feedback [webhelp_feedback]

This component can be used in the [WebHelp Responsive Topic Template](#) to indicate the place where the feedback part will be presented.

The element associated with this component is

```
<whc:webhelp_feedback xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

Main Page Topic Tiles [webhelp_tiles]

This component can be used in the [WebHelp main page template](#) to generate the tiles section in the main page. This section will contains a tile for each root topic of the published documentation. Each topic tile has three sections that corresponds to topic title, short description and image.

The element associated with this WebHelp template component is

```
<whc:webhelp_tiles xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

It will generate in the output an element with `webhelp_tiles` class.

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. Such a tile template could be:

```
<whc:webhelp_tile class="col-md-4">
  <!-- Place holder for tile's image -->
  <whc:webhelp_tile_image/>

  <div class="webhelp_tile_text">
    <!-- Place holder for tile's title -->
    <whc:webhelp_tile_title/>

    <!-- Place holder for tile's shordesc -->
    <whc:webhelp_tile_shordesc/>
  </div>
</whc:webhelp_tile>
```

Main Page Table of Contents [webhelp_main_page_toc]

This component can be used in the [WebHelp main page template](#) to generate a simplified Table of Content structure. It is simplified because it contains only two levels from the documentation hierarchy.

The element associated with this WebHelp template component is

```
<whc:webhelp_main_page_toc xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

and will generate in the output an element with the `webhelp_main_page_toc` class.

Main Page Menu [webhelp_topics_menu]

This component can be used in the [WebHelp main page template](#) to generate a menu with all the documentation topics.

The user can control the maximum level of topics that will be included in the menu using the `webhelp.top.menu.depth` transformation parameter.

The element associated with this WebHelp template component is

```
<whc:webhelp_topics_menu xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

Print Link [webhelp_print_link]

This component can be used in all WebHelp template pages to generate a button that displays the browser's print dialog.

The element associated with this WebHelp template component is

```
<whc:webhelp_print_link xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

It will generate in the output an element with `webhelp_print_link` class.

Include HTML files [webhelp_include_html]

This component can be used in all the [WebHelp template pages](#) to include custom HTML files.

The element associated with this WebHelp template component is

```
<whc:include_html href="{wh.param}"/>
```

Where the `href` can have the next 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 of which path is specified through a WebHelp transformation scenario parameter. The value of this parameter can be a simple HTML fragment, case when it will be copied to the output.

Index Terms Link [webhelp_indexterms_link]

This component can be used in all the *WebHelp template pages* 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.

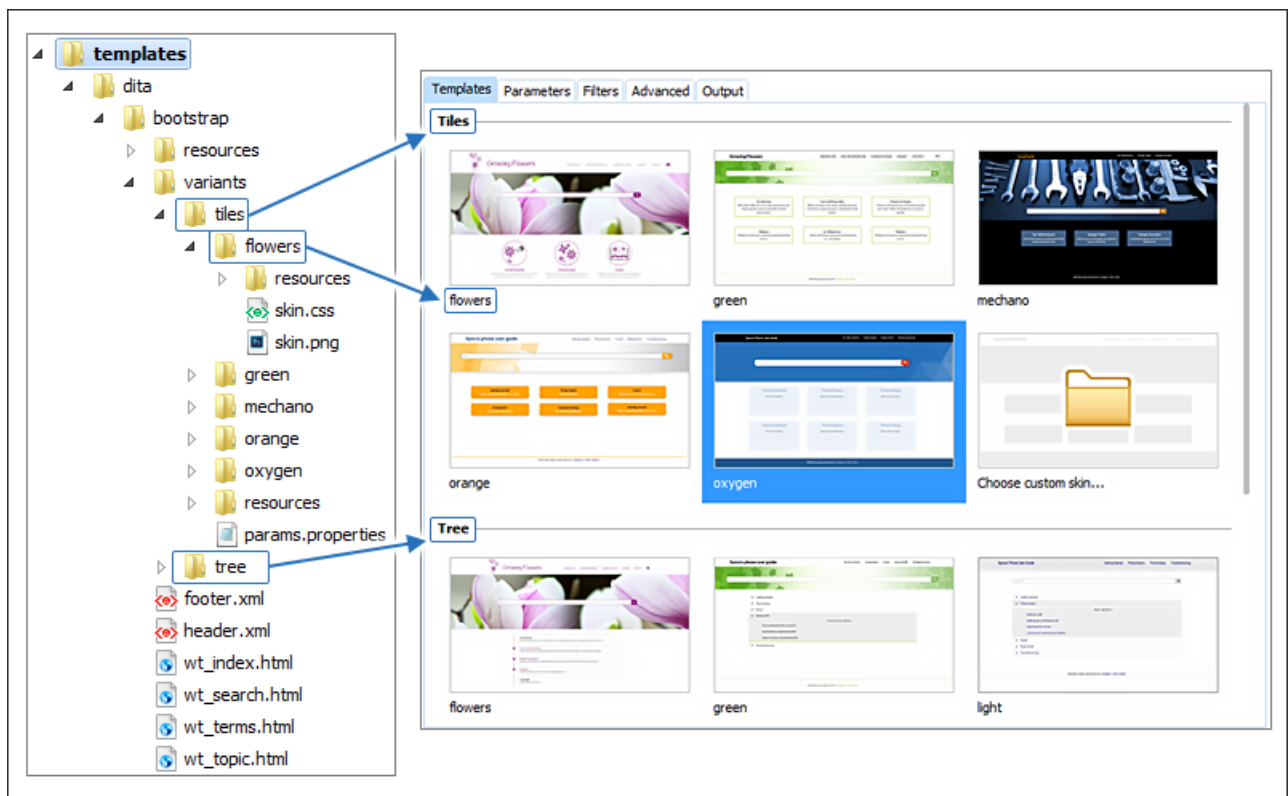
The element associated with this WebHelp template component is

```
<whc:webhelp_indexterms_link xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

It will generate in the output an element with the `webhelp_indexterms_link` class that contains a link to the `indexterms.html` page.

WebHelp Template Resources

All templates are stored in the **templates/dita** folder located in the root folder of the oXygen XML WebHelp plugin. The structure of an WebHelp Responsive template is the following:



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.
- **template files**:
 - `wt_index.html` - Used to generate main page.
 - `wt_topic.html` - Used to generate HTML pages for each topic.
 - `wt_search.html` - Used to generate the HTML page for presenting the search result.
 - `wt_terms.html` - Used to generate the page for index terms.

After the transformation scenario is executed, the resources and variants folders are copied in the output in the `{outDir}/oxygen-webhlep/template` directory.

Template, Variant and Skin Concept

A **template** could be seen as a set of WebHelp components that are put in a predefined HTML layout.

If we want to display one or more template components we could obtain one or more **variants** of the template. So, a WebHelp **template variant** is an instance of the template with a specific set of parameters. For instance, we could have two variants of the WebHelp main page, one that displays the topics as *tiles* and another one that displays the topics in a tree-like fashion.

Each variant has its own directory which gives the name of the variant. The name of the variant is also displayed in the user interface when the template's variants are displayed.

The variant's directory could contain the following resources:

- **params.properties** - This file specifies the values for the parameters imposed by the variant.
- **resources** - Optional directory. It contains resources that are specific to the current variant (images, CSS files, etc). They will be copied to the output directory.
- one or more skin directories that represent skins of the current variant.

A variant's *skin* represents a CSS that allows you to alter the styling of the template. This skin might need additional resources (images, fonts) that must be stored in the **resources** directory in the skin's root folder. The name of the skin directory will give the name of the skin and it will be displayed in the UI when you choose a variant's skin.

The CSS associated with a skin must be named **skin.css** and it must be stored as first child of the skin's directory.

The skin directory can also contain a **skin.png** preview image, that will be displayed in the UI, and a properties file containing an URL for the online preview of the skin.

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. Each method has its own advantages and limitations. This topic provides information on each of the methods so that you can choose the best possible method based upon your needs.

Insert Custom HTML 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 useful 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 313: Predefined Placeholders Diagram

1- webhelp.fragment.head	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed as the header in the output.
2- webhelp.fragment.before.body	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the body in the output.
3- webhelp.fragment.before.logo_and_title	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the logo and title in the output.
4- webhelp.fragment.after.logo_and_title	Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the logo and title in the output.
5- webhelp.fragment.before.top_menu	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the top menu in the output.
6- webhelp.fragment.after.top_menu	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the top menu in the output.
7- webhelp.fragment.before.main.page.search	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the search field in the output.
8- webhelp.fragment.welcome	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed as a welcome message (or title) in the output.
9- webhelp.fragment.after.main.page.search	Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the search field in the output.
10- webhelp.fragment.before.toc_or_tiles	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the table of contents or tiles in the main page of the output.
11- webhelp.fragment.after.toc_or_tiles	Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed

after the table of contents or tiles in the main page of the output.

12- `webhelp.fragment.footer`

Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed as the footer in the output

13- `webhelp.fragment.after.body`

Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the body in the output.

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 useful if you want to use an existing skin and it is very close to what you need, but you want to adjust 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](#).

Create a New WebHelp Responsive Skin

This method is useful 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 project (you could use one of the Oxygen XML Developer 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).
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 Developer sample projects). Also, you can use your web browser CSS inspector tool to lookup the CSS classes you want to modify.

Adding a Logo Image in the Title Area

To customize 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, set the URL in the `webhelp.logo.image.target.url` parameter.
4. Run the transformation scenario.

How to set a Welcome Message in the Home Page

The main page of the Welcome Responsive output contains a set of empty placeholders that can be used to display customized text fragments. These placeholders are available to you through [the 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* that suits you best.
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 our 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.

How to Configure the Main Page Tiles

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 source. 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 these tiles either by setting metadata information in the DITA map or by customizing the transformation scenario DITA map.

Hiding some of the tiles displayed in the main page

When the documentation is very large and the number of topics on the first level is very large, you might want to hide some of them. Also, this might prove useful when you want to display in the first page only the topics that are most relevant to your intended audience.

There are two methods of 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 control which topic on the first level of the DITA map will not be displayed as a tile:

1. Open the DITA map in the Text mode of Oxygen XML Developer.
2. Add the following metadata information in the `topicref` element (or any of its specializations) for each first-level topic 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 control which topic on the first level of the DITA map will not be displayed as a tile:

1. Make sure you set an id to the topic you want to hide. For the purpose of this procedure, we suppose that the id you set to a topic has the value `growing-flowers`.
2. Create a new CSS file that contains a rule that hides the tile generated for our topic (identified by the topic id `growing-flowers`). The CSS file has the following content:

```
.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.

Setting an Image in the Tiles Displayed in the Main Page

There are two methods to set an image in a tile: one of them involves setting *metainformation* in a DITA map and the other one that uses a CSS .

Editing the DITA Map

1. Open the DITA map in the Text mode of Oxygen XML Developer.
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.

Using a Customized CSS

1. Make sure you set an id to the topic for which the tile will be decorated with an image. For the purpose of this procedure, we suppose that the id you set to a topic has the value `growing-flowers`.
2. Create a new CSS file that contains a rule that associates an image with a specific tile. The CSS file has the following content:

```
.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.

Customizing the Menu

By default, the menu component is displayed in all WebHelp Responsive pages. However, for some reasons you might want to hide it completely, or, display only some of its menu entries.

Hiding Some Menu Entries

Editing the DITA map

1. Open the DITA map in the Text mode of Oxygen XML Developer.

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 hide some of the menu entries:

1. Make sure you set an id to the topic for which you do not want to have a menu entry. For the purpose of this procedure, we suppose that the id you set to a topic has the value `growing-flowers`.
2. Create a new CSS file that contains a rule that hides the menu entry generated for our topic (identified by the topic id `growing-flowers`). The CSS file has the following content:

```
.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.

Hiding the Entire Menu

If you do not need 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. Follow this procedure:

1. Edit a WebHelp Responsive transformation scenario.
2. Open the [Templates tab](#) and choose a *skin* that suits you best.
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.

Integrating Social Media and Google Tools in WebHelp Output

Oxygen XML Developer includes support for integrating some of the most popular social media sites in WebHelp output.

How to Integrate Google Analytics in WebHelp Output

See how you can integrate Google Analytics into WebHelp output.

To enable 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-XXXXXXX-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 Developer, 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 545

How to Add a Facebook Like Button in WebHelp Responsive Output

See how you can add a Facebook widget into your 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 Developer, 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 Tweet Button in WebHelp Responsive Output

See how you can add a Twitter widget into your 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 Developer, 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 545

How to Add a Google+ Button in WebHelp Responsive Output

See how you can add a Google+ widget into your 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 Developer, 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 545

How to Localize 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. 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"/>`.
4. 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).
5. Run the transformation scenario to produce the WebHelp output.

Related information

[Creating New Transformation Scenarios](#) on page 560

Localizing the Email Notifications of WebHelp with Feedback Systems

The WebHelp with Feedback system uses 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 with Feedback

1. Create the following directory:
`[OXYGEN_INSTALL_DIR]\frameworks\docbook\xml\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\nl`
2. Copy all English template files from `[OXYGEN_INSTALL_DIR]\frameworks\docbook\xml\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\xml\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\en` directory and translate the content into Dutch.

4. Start Oxygen XML Developer 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 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 Developer 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.

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 Developer 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\xs\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.

Adding Videos in the Output

Videos can be included and played in all HTML5-based output formats (such as *WebHelp*). For example, to add a YouTube video in the WebHelp output generated from DITA or DocBook documents, follow the procedures below.

Adding Videos to WebHelp Generated from DITA Maps

1. Edit the DITA topic to reference the video using an `object` element, as in the following example:

```
<object outputclass="video">
  <param name="src" value="http://www.youtube.com/watch/v/VideoName"/>
</object>
```

2. Apply a *WebHelp* or *WebHelp with Feedback* transformation scenario to obtain the output.

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.

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.

WebHelp Responsive Runtime Additional Parameters

A deployed WebHelp Responsive system can accept the following GET parameters:

- **contextId** - The WebHelp JavaScript engine will look up the value of this parameter in the mapping file and load the corresponding HTML 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`.

- **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`.

Flag DITA Content

Flagging content 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-level 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.

Support for Right-to-Left (RTL) Oriented Languages for DITA WebHelp

To activate support for RTL 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

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** - the date when the page was last modified. The date format is `YYYY-MM-DD`.
- **changefreq** - 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** - 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.



Note: `lastmod`, `changefreq`, and `priority` are optional elements.

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 Developer 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.

Indexing Japanese Content for DITA WebHelp Pages

To optimize the indexing of Japanese content in WebHelp pages generated from DITA map transformations, the Kuromoji analyzer can be used. This analyzer is not included in the Oxygen XML Developer installation kit and must be downloaded and added.

To use the Kuromoji analyzer to index Japanese content in your WebHelp system, follow these steps:

1. Download the analyzer jar file from <http://mvnrepository.com/artifact/org.apache.lucene/lucene-analyzers-kuromoji/4.0.0>.

2. Place the Kuromoji analyzer jar file in the following directory:
`DITA_OT_DIR/plugins/com.oxygenxml.webhelp/lib.`
3. For the analyzer to work properly, search terms that are entered into your WebHelp pages must be separated by spaces.

Optionally a Japanese user dictionary can be set with the `webhelp.search.japanese.dictionary` parameter.

Related information

[DITA Map to WebHelp Output](#) on page 545

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 Developer 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 <i>Table of Contents</i>. 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 <code>indexterm</code> 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.
Main Pane or Frame	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 <code>args.hide.parent.link</code> parameter to hide the Parent, Next, and Previous links.
	The content of the help pages are rendered and displayed in this main section.	

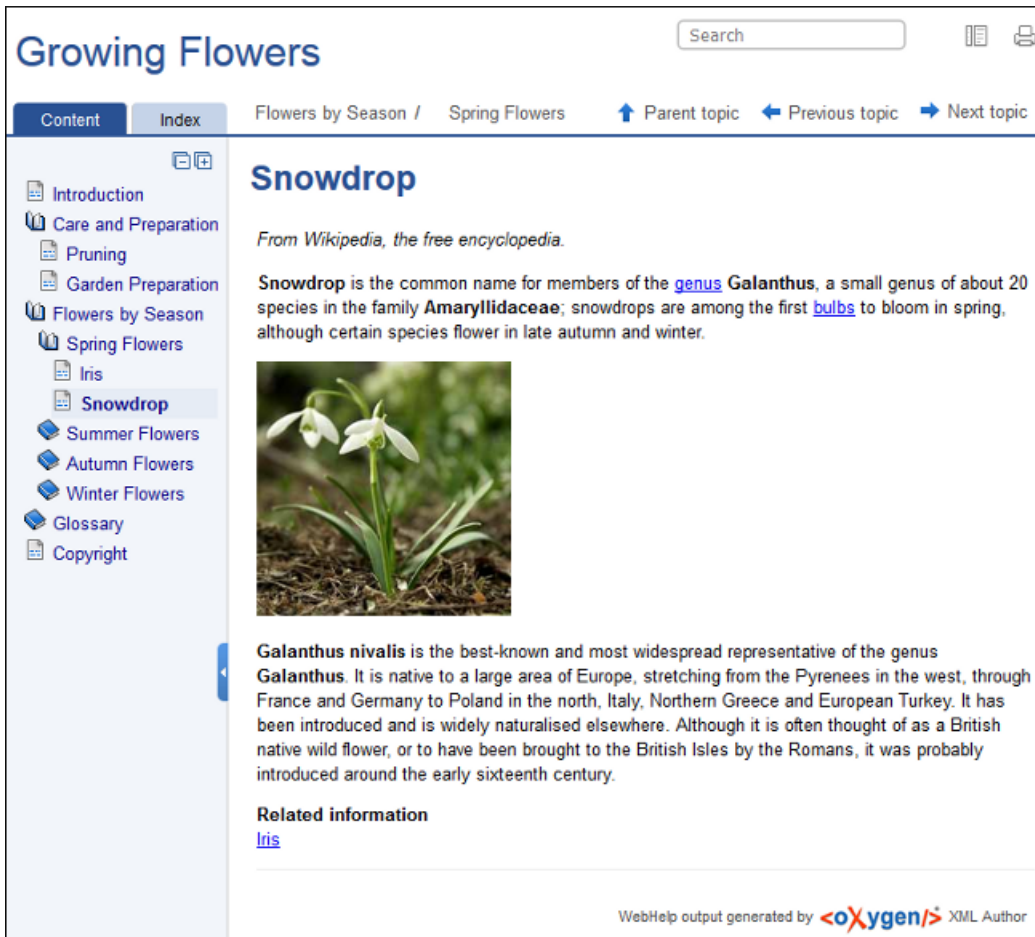


Figure 314: WebHelp Classic Output

Search Feature

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, along with a rating scheme for each result. When you click on a result in the **Search Results** tab, the search terms are highlighted in the main pane. If you click **Enter** with the **Search** field empty, the highlights are removed.

The **Search** feature is also enhanced with a rating mechanism that computes scores for every page that matches the search criteria. These scores are then translated into a 5-star rating scheme. The search results are sorted depending on the following:

- 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 phrase is included in a meta keyword
 - The search phrase is in the title of the page
 - The search phrase is in bold text in a paragraph
 - The search phrase is in normal text in a paragraph

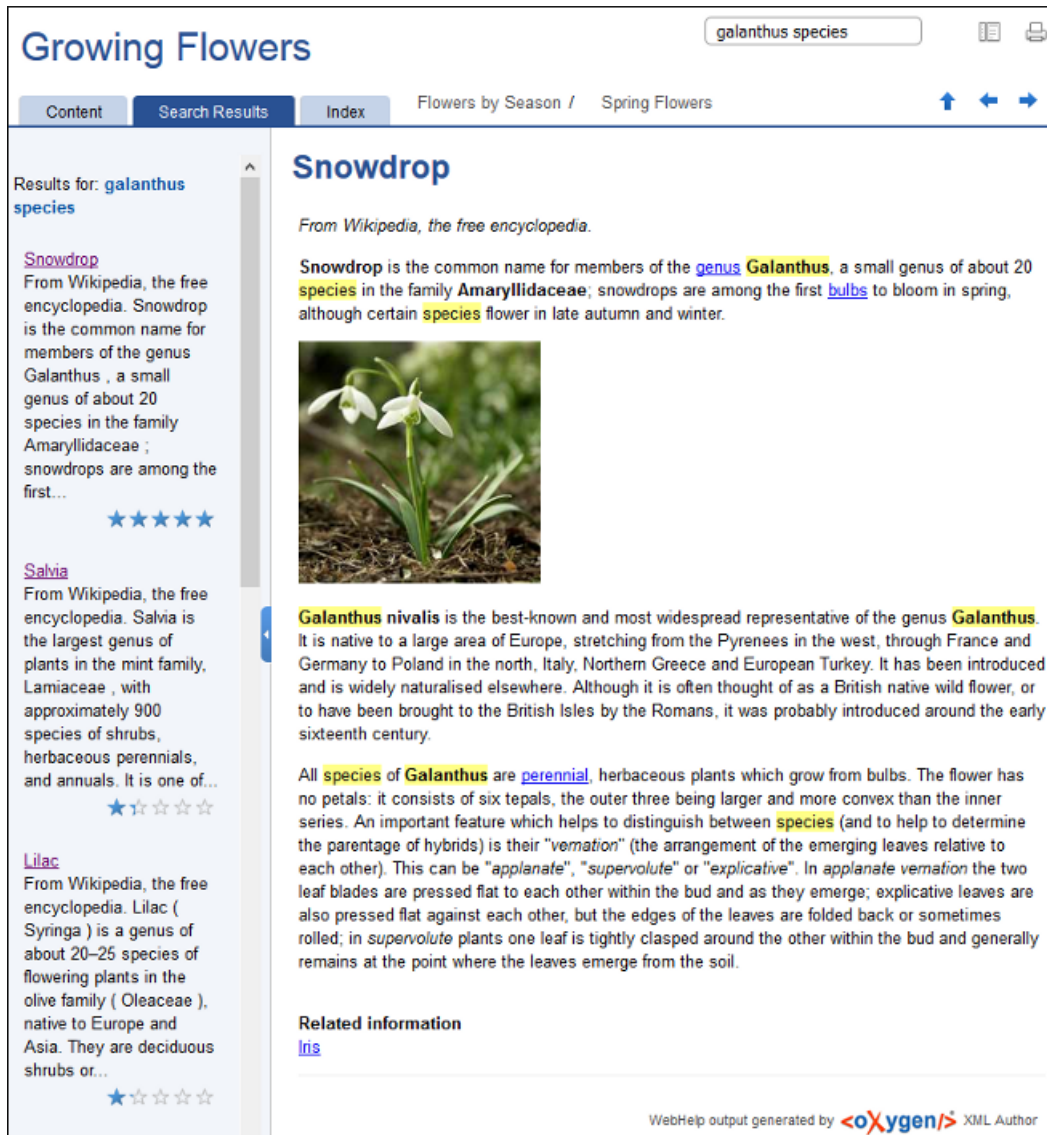


Figure 315: WebHelp Classic Search Feature

Rules that are applied during a search include:

- Boolean searches are 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*).
- Do not use quotes to perform an exact search for multiple word expressions (an expression such as "*grow flowers*", returns no results since it searches for two separate words).
- *indexterm* and *keywords* DITA elements are an effective way to increase the ranking of a page (for example, content inside *keywords* elements weighs twice as much as content inside 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.

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 the [Editing Scoring Values of Tag Elements in Search Results](#) on page 638 topic.

This output format is compatible with the most recent versions of the following common browsers:

- Internet Explorer (IE 8 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 Developer 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 <i>Table of Contents</i> . 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 <code>indexterm</code> 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.

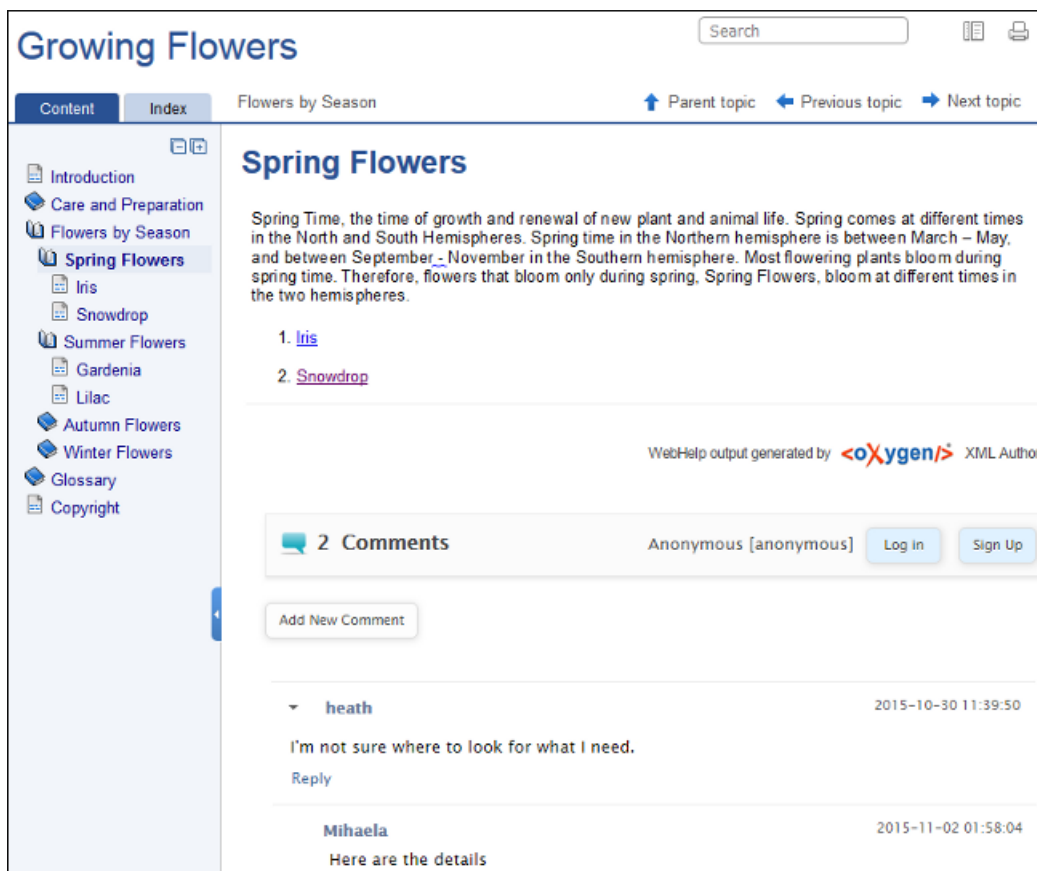


Figure 316: WebHelp Classic with Feedback System

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 the [Managing Users and Comments in a WebHelp Classic with Feedback System](#) on page 651 topic.

Search Feature

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, along with a rating scheme for each result. When you click on a result in the **Search Results** tab, the search terms are highlighted in the main pane. If you click **Enter** with the **Search** field empty, the highlights are removed.

The **Search** feature is also enhanced with a rating mechanism that computes scores for every page that matches the search criteria. These scores are then translated into a 5-star rating scheme. The search results are sorted depending on the following:

- 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 phrase is included in a meta keyword
 - The search phrase is in the title of the page
 - The search phrase is in bold text in a paragraph
 - The search phrase is in normal text in a paragraph

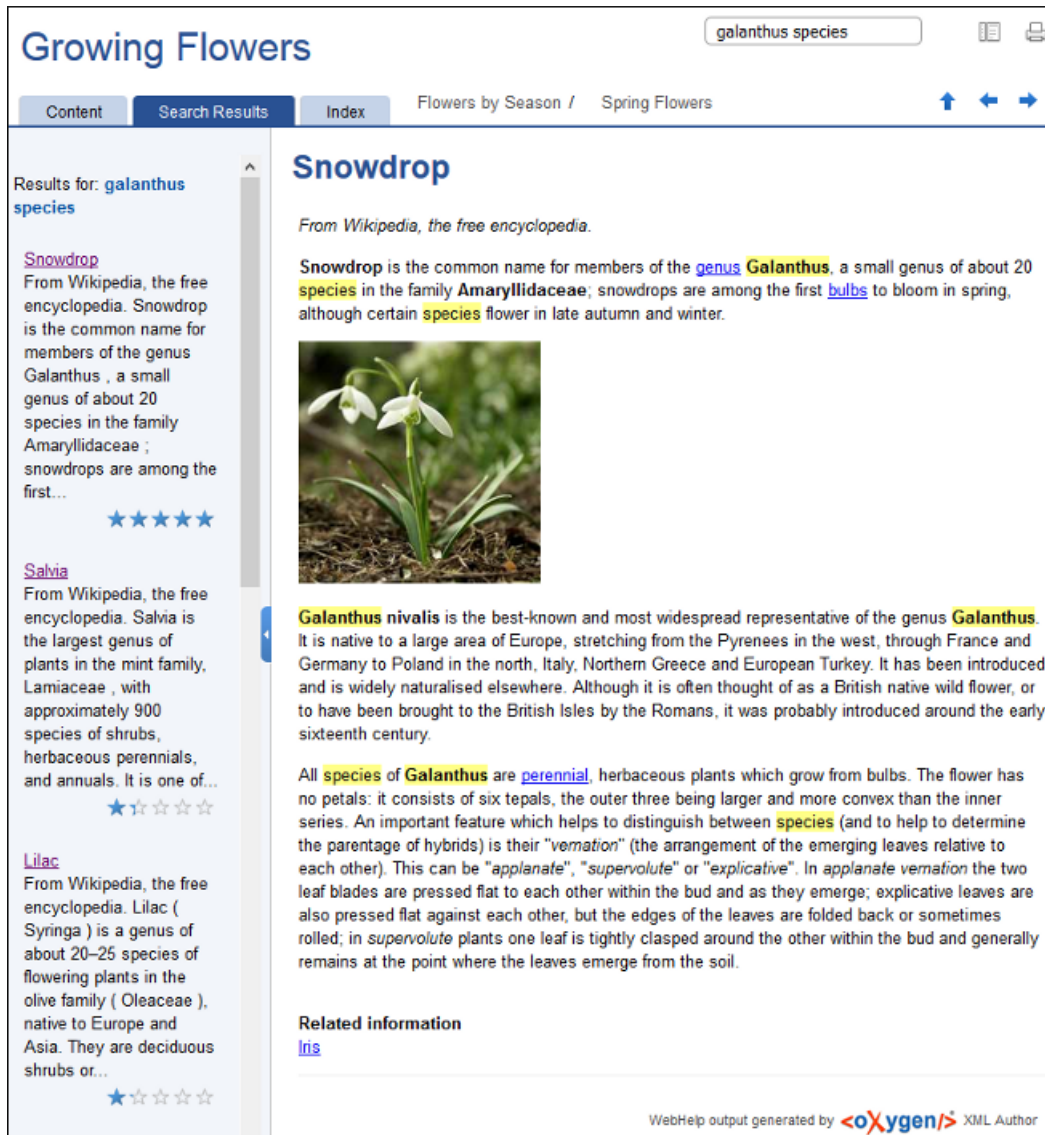


Figure 317: WebHelp Classic Search Feature

Rules that are applied during a search include:

- Boolean searches are 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*).
- Do not use quotes to perform an exact search for multiple word expressions (an expression such as "*grow flowers*", returns no results since it searches for two separate words).
- *indexterm* and *keywords* DITA elements are an effective way to increase the ranking of a page (for example, content inside *keywords* elements weighs twice as much as content inside 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.

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 the [Editing Scoring Values of Tag Elements in Search Results](#) on page 638 topic.

This output format is compatible with the most recent versions of the following common browsers:

- Internet Explorer (IE 8 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 the WebHelp Classic with Feedback System

System Requirements

The *WebHelp Classic with Feedback* system of Oxygen XML Developer 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, Internet Explorer, Safari, Opera.

Create WebHelp Classic with Feedback Database

The *WebHelp Classic 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.

For 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 Classic with Feedback Output

If you have a web server configured with PHP and *MySQL*, you can deploy the *WebHelp Classic 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/](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 Classic with Feedback Database section](#) to fill-in the appropriate text boxes.



Warning: Checking 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 Classic 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, check the **Enable LDAP Authentication** 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 checked, 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 Classic 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, enable 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 Classic 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 Classic with Feedback* system main page.

Testing Your WebHelp Classic 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 Classic 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 Classic 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 WebHelp Classic with Feedback System](#) on page 651

Refreshing the Content of a WebHelp Classic with Feedback System

It is common to update the content of an existing installation of a *WebHelp Classic 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:

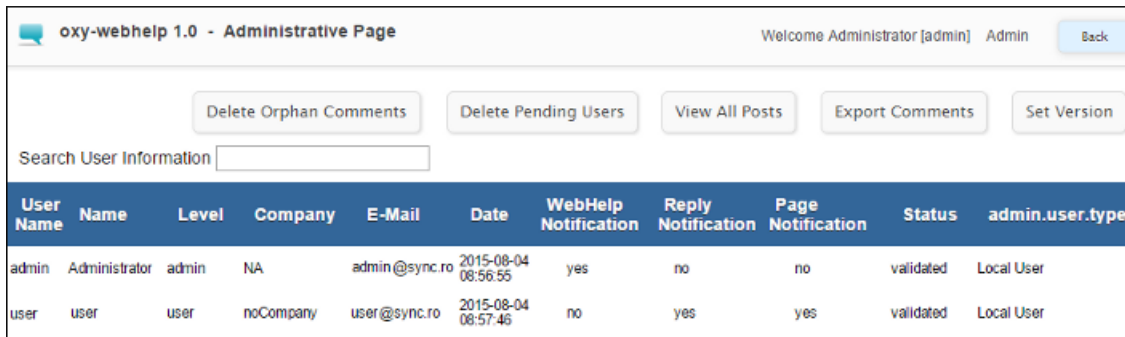
1. Execute the transformation scenario that produces the *WebHelp Classic with Feedback* output directory.
2. Go to the output directory (specified in the **Output** tab of the transformation scenario), locate the `\feedback\resources\php\config\config.php` file, and delete it.
3. Locate the `\feedback\install` directory and delete it.
4. Copy the remaining structure of the output folder and paste it into your *WebHelp Classic with Feedback* system installation directory, overwriting the existing content.

Managing Users and Comments in a WebHelp Classic with Feedback System

When you installed the *WebHelp Classic with Feedback* system the first time (assuming the [Create new database structure option](#) was enabled), 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 Classic 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.



User Name	Name	Level	Company	E-Mail	Date	WebHelp Notification	Reply Notification	Page Notification	Status	admin.user.type
admin	Administrator	admin	NA	admin@sync.ro	2015-08-04 08:56:55	yes	no	no	validated	Local User
user	user	user	noCompany	user@sync.ro	2015-08-04 08:57:45	no	yes	yes	validated	Local User

Figure 318: 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 enabled, the user receives notifications when comments are posted anywhere in your feedback-enabled WebHelp system.

Reply Notification When enabled, the user receives notifications when comments are posted as a reply to one of their comments.

Page Notification When enabled, 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.

Support for Right-to-Left (RTL) Oriented Languages for DITA WebHelp

To activate support for RTL 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

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** - the date when the page was last modified. The date format is `YYYY-MM-DD`.
- **changefreq** - 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** - 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.



Note: `lastmod`, `changefreq`, and `priority` are optional elements.

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 Developer 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.

Indexing Japanese Content for DITA WebHelp Pages

To optimize the indexing of Japanese content in WebHelp pages generated from DITA map transformations, the Kuromoji analyzer can be used. This analyzer is not included in the Oxygen XML Developer installation kit and must be downloaded and added.

To use the Kuromoji analyzer to index Japanese content in your WebHelp system, follow these steps:

1. Download the analyzer jar file from <http://mvnrepository.com/artifact/org.apache.lucene/lucene-analyzers-kuromoji/4.0.0>.
2. Place the Kuromoji analyzer jar file in the following directory:
`DITA_OT_DIR/plugins/com.oxygenxml.webhelp/lib`.
3. For the analyzer to work properly, search terms that are entered into your WebHelp pages must be separated by spaces.

Optionally a Japanese user dictionary can be set with the `webhelp.search.japanese.dictionary` parameter.

Related information

[DITA Map to WebHelp Output](#) on page 545

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 Developer 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 Developer 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 <http://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 Transformation Scenario

1. Start Oxygen XML Developer.
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 Transformation Scenario

1. Start Oxygen XML Developer.
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.

Related information

[Skins Tab \(DITA OT Transformations\)](#) on page 572

[DITA OT Transformation](#) on page 571

This type of transformation specifies the parameters for an Ant transformation that executes a DITA-OT build script. Oxygen XML Developer includes a built-in version of Ant and a built-in version of DITA-OT, but other versions can be set in the scenario.

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.

Integrating Social Media and Google Tools in WebHelp Output

Oxygen XML Developer includes support for integrating some of the most popular social media sites in WebHelp output.

How to Add a Facebook Like Button in WebHelp Output

See how you can add a Facebook widget into your WebHelp 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 Developer, 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 545

How to Add Tweet Button in WebHelp Output

See how you can add a Twitter widget into your WebHelp 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 Developer, 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 545

How to Add a Google+ Button in WebHelp Output

See how you can add a Google+ widget into your WebHelp 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 Developer, 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 545

How to Integrate Google Search in WebHelp Output

See how you can integrate a Google Custom Search Engine into WebHelp output.

You can integrate Google Search into your WebHelp output.

To enable 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 Developer, 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 that contains the Google Custom Search Engine element `gcse:searchresults-only`. It is recommended that you set the `linkTarget` attribute value to `frm` for *frameless* versions of the WebHelp system or to `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.
 - b) Set the value of the `webhelp.google.search.results` parameter to the location 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 634

How to Integrate Google Analytics in WebHelp Output

See how you can integrate Google Analytics into WebHelp output.

To enable 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-XXXXXXX-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 Developer, 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 545

Localizing the Interface of WebHelp Output

You can localize the interface of WebHelp output for DITA or DocBook transformations.

How to Localize 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. 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"/>`.
4. 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).
5. Run the transformation scenario to produce the WebHelp output.

Related information

[Creating New Transformation Scenarios](#) on page 560

How to Localize 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.

Localizing the Email Notifications of WebHelp with Feedback Systems

The WebHelp with Feedback system uses 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 with Feedback

1. Create the following directory:

```
[OXYGEN_INSTALL_DIR]\frameworks\docbook\xml\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\nl
```

2. Copy all English template files from `[OXYGEN_INSTALL_DIR]\frameworks\docbook\xml\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\xml\com.oxygenxml.webhelp\oxygen-webhelp\resources\php\templates\nl` directory and translate the content into Dutch.
4. Start Oxygen XML Developer 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 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 Developer 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.

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 Developer 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.

Adding Videos in the Output

Videos can be included and played in all HTML5-based output formats (such as *WebHelp*). For example, to add a YouTube video in the WebHelp output generated from DITA or DocBook documents, follow the procedures below.

Adding Videos to WebHelp Generated from DITA Maps

1. Edit the DITA topic to reference the video using an `object` element, as in the following example:

```
<object outputclass="video">
  <param name="src" value="http://www.youtube.com/watch/v/VideoName"/>
</object>
```

2. Apply a *WebHelp* or *WebHelp with Feedback* transformation scenario to obtain the output.

Adding Videos to WebHelp Generated from DocBook

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 Icons in a WebHelp Table of Contents

You can change the icons that appear in a WebHelp table of contents by assigning new image files in a custom CSS file. By default, the icons for the WebHelp 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\xml\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.

Customize the Appearance of Selected Items in the Table of Contents

The appearance of selected items in the Table of Contents 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\xml\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 Table of Contents](#) topic).

Adding a Logo Image in the Title Area

To customize 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, set the URL in the `webhelp.logo.image.target.url` parameter.
4. Run the transformation scenario.

Removing the *Previous/Next* Links from WebHelp Pages

The **Previous** and **Next** links that are created at the top area of each WebHelp page can be hidden with a CSS code.

To remove these links from WebHelp pages, 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.

Customizing the Header and Footer

In the transformation scenario, you can use the `args.hdr` and `args.ftr` parameters to point to resources that contain your custom HTML `<div>` blocks. These are included in the header and footer of each generated topic.

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.

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.

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.

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 up the value of this parameter in the mapping file and load the corresponding HTML 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`.

- `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`.

Disable Caching in WebHelp Pages

In cases where a set of WebHelp 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 pages, 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.

Adding a Button in Code Snippet Areas

This task will get you started on how to add an action (such as a button or link) in code snippet areas that are displayed in WebHelp output. 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>
```

Flag DITA Content

Flagging content 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-level 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.

WebHelp Classic Mobile 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 Developer 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, is organized in an intuitive layout, and offers table of contents, search capabilities, and index navigation.



Figure 319: 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.

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.

Context-Sensitive WebHelp 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.

When WebHelp output is generated by Oxygen XML Developer, 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, as in the following example:

```
<map productID="oxy-webhelp" productVersion="1.1">
  <appContext helpID="annotations-view" path="topics/annotations-view.html"/>
  <appContext helpID="button-editor" path="concepts/button-editor.html"/>
  .
</map>
```

- `helpID` - Unique ID provided by a topic from two possible sources:
 - The `resourceid` element set to it in the `prolog` section:

```
<prolog>
  <resourceid id="context-sensitive-help-system"/>
</prolog>
```



Note: If you need multiple parts of the application (for instance, dialog boxes, views, or editing modes) to open the same contextual help topic, all of the context ID values should be included in the same DITA topic file. For example, if you need both a dialog box and a view to open the same WebHelp page, you can assign both resource IDs in the same DITA topic.

```
<prolog>
  <resourceid id="dialog1"/>
  <resourceid id="view1"/>
</prolog>
```

- The `id` attribute set on the topic root 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. In this case, the help system will not work properly.
- `path` - Path to a corresponding WebHelp page. This path is relative to the location of the `context-help-map.xml` mapping file.
 - `productID` - ID of the product for which you are writing documentation. Applicable only if you are using the feedback variants of WebHelp transformations.
 - `productVersion` - Version of the product for which you are writing documentation. Applicable only if you are using the feedback variants of WebHelp transformations.

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
```

Related information

[WebHelp Classic Runtime Additional Parameters](#) on page 665

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 Developer 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 Developer.

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 http://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

This topic includes the procedure for 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 later
- DITA Open toolkit 1.8 or 2.0 (Full Easy Install, includes Saxon 9.1.0.8 libraries)
- Saxon 9.1.0.8

To integrate the Oxygen XML WebHelp plugin with the DITA Open Toolkit, follow these steps:

1. Download and install a [Java Virtual Machine](#) 1.6 or later.
2. Download and install [DITA Open Toolkit](#) 1.8.x or 2.x.
3. Go to [Oxygen XML WebHelp website](#), download the latest installation kit, and unzip it.

4. Copy the `com.oxygenxml.webhelp` and `com.oxygenxml.highlight` directories inside the `plugins` directory of the DITA OT distribution. The `com.oxygenxml.highlight` directory adds syntax highlight capabilities to your WebHelp output for codeblock sections that contain source code snippets (XML, Java, JavaScript etc.).
5. If you are using DITA-OT version 2.x, the WebHelp plugin contains a `plugin_2.x.xml` that needs to be renamed to `plugin.xml`.
6. In the home directory of the DITA Open Toolkit, run `ant -f integrator.xml`.

Licensing the Oxygen XML WebHelp Plugin for DITA OT

This topic explains how to register the license for the Oxygen XML WebHelp plugin for the DITA Open Toolkit.

To register the license for the Oxygen XML WebHelp plugin for the DITA Open Toolkit, follow these steps:

1. Open the `[OXYGEN_INSTALL_DIR]/frameworks/dita/DIT-OT/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 Developer license key from this file. In case the file does not exist, or it contains an invalid license, an error message will be displayed.

Upgrading the Oxygen XML WebHelp Plugin for DITA OT

This topic describes the procedure for upgrading a Oxygen XML WebHelp plugin for the DITA Open Toolkit.

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.webhelp`).
2. Go to [Oxygen XML WebHelp website](#), download the latest installation kit, and unzip it.
3. Copy the `com.oxygenxml.webhelp` and `com.oxygenxml.highlight` directories inside the `plugins` directory of the DITA OT distribution. The `com.oxygenxml.highlight` directory adds syntax highlight capabilities to your WebHelp output for codeblock sections that contain source code snippets (XML, Java, JavaScript etc.).
4. If you are using DITA-OT version 2.x, the WebHelp plugin contains a `plugin_2.x.xml` that needs to be renamed to `plugin.xml`.
5. In the home directory of the DITA Open Toolkit, run `ant -f integrator.xml`.

Running an External DITA Transformation Using the Oxygen XML WebHelp Plugin

This topic explains how to run an external DITA to WebHelp transformation using the Oxygen XML WebHelp plugin.

To run a DITA to WebHelp ([webhelp-responsive](#), [webhelp](#), [webhelp-feedback](#), [webhelp-mobile](#)) transformation using the Oxygen XML WebHelp plugin, use the following:

DITA-OT 2.x versions

- `DITA_OT_DIR\bin\dita.bat -i path_to_input.ditamap -f webhelp` (Windows based systems)
- `DITA_OT_DIR\bin\dita -i path_to_input.ditamap -f webhelp` (Unix/Linux based systems)

DITA-OT 1.x versions

- `WEBHELP_PLUGIN_HOME_DIR\dita.bat` script file (Windows based systems)
- `WEBHELP_PLUGIN_HOME_DIR\dita.sh` script file (Unix/Linux based systems)



Warning: You can also invoke the DITA OT WebHelp publishing using a startup script but you will lose certain small fixes and patches that Oxygen XML Developer added to the automated DITA OT processing.

The `dita.bat` and the `dita.sh` files are located in the home directory of the Oxygen XML WebHelp Plugin. Before using them to generate a WebHelp system, customize them to match the paths to the JVM, DITA Open Toolkit, and Saxon engine, 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 `dita.bat` and `dita.sh` scripts reference the `dost-patches-DITA-1.8.jar` JAR file containing DITA OT 1.8-specific patches. If you use DITA OT 1.7, update that reference to `dost-patches-DITA-1.7.jar`. If you use DITA OT 2.0, 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`, `webhelp-feedback` and `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.

The optional parameter `-Dargs.filter` can be used to specify a filter file to be used to include, exclude, or flag content.

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* page.

-Dwebhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory are included in 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. To integrate social media features (such as widgets for Facebook™, Twitter™, Google+™), the XML must be well-formed, each widget must be in separate `div` or `span` element, and the widget is included in an XML comment (also, the start and end tags for the XML comment). The following code excerpt 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)) {
      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-status="share">
  </div>
```

-Dwebhelp.footer.include (not available for WebHelp Responsive systems)

Specifies whether or not to include footer in each WebHelp page. Possible values are `yes` and `no`. If set to `yes` and the `webhelp.footer.file` is specified, the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default footer is inserted in each WebHelp page.

-Dwebhelp.google.search.results

URL value that specifies the location of a well-formed XHTML file containing the search results. The element `gcse:searchresults-only`. You can use all supported attributes. To set the `linkTarget` attribute to `frm` for frameless (*iframe*) version of WebHelp, use `linkTarget="frm"`. The default value for this attribute is `_blank` and the search results are displayed in a new window. If this parameter is not specified, the following code will be used `<gcse:searchresults-only linkTarget="frm">`

-Dwebhelp.google.search.script

Specifies the location of a well-formed XHTML file containing the Custom Search script. The value must be a URL.

-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 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 h
-Dwebhelp.product.id (available only for Feedback-enabled systems)	This parameter specifies a short name for the documentation target, or product (mobile-phone-user-guide, hvac-installation-guide).  Note: You can deploy documentation for multiple products on the same  Restriction: The following characters are not allowed in the value of } = ; * % + &.
-Dwebhelp.product.version (available only for Feedback-enabled systems)	Specifies the documentation version number (for example, 1.0, 2.5, etc.). New  Note: Multiple documentation versions can be deployed on the same  Restriction: The following characters are not allowed in the value of } = ; * % + &.
-Dwebhelp.search.japanese.dictionary	The file path of the dictionary that will be used by the <i>Kuromoji</i> morphological uses for indexing Japanese content in the WebHelp pages. This indexer does not use the Oxygen XML Developer or the Oxygen XML WebHelp plugin. To use it, you need to download the http://mvnrepository.com/artifact/org.apache.lucene/lucene-analyzers-kuromoji from the <code>DITA_OT_DIR/plugins/com.oxygenxml.webhelp/lib</code> directory.
-Dwebhelp.search.ranking	If this parameter is set to <code>false</code> then the 5-star rating mechanism is no longer displayed on the Search tab (default setting is <code>true</code>).
-Dwebhelp.show.changes.and.comments	When set to <code>yes</code> , user comments, replies to comments, and tracked changes are displayed. The default value is <code>no</code> .
-Dwebhelp.sitemap.base.url	Base URL for all the <code>loc</code> elements in the generated <code>sitemap.xml</code> file. The value is the relative file path from the <code>href</code> attribute of a <code>topicref</code> element from the <code>topic</code> file. The <code>loc</code> element is mandatory in <code>sitemap.xml</code> . If you leave this parameter empty, the <code>sitemap.xml</code> file is not generated.
-Dwebhelp.sitemap.change.frequency	The value of the <code>changefreq</code> element in the generated <code>sitemap.xml</code> file. The value is in <code>sitemap.xml</code> . If you leave this parameter set to its default empty value, the <code>changefreq</code> element is not added in <code>sitemap.xml</code> . Allowed values: <code><empty string></code> (default), <code>always</code> , <code>hourly</code> , <code>daily</code> , <code>weekly</code> , <code>monthly</code> , <code>yearly</code> , <code>never</code> .
-Dwebhelp.sitemap.priority	The value of the <code>priority</code> element in the generated <code>sitemap.xml</code> file. It can be a value from 0.0 (least important priority) and 1.0 (most important priority). For example, <code>0.5</code> . The <code>priority</code> element is optional in <code>sitemap.xml</code> . If you leave this parameter set to its default empty value, the <code>priority</code> element is not added in <code>sitemap.xml</code> .
-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 the <code>OXYGEN_INSTALL_DIR\frameworks\docbook\xsl\com.oxygenxml.docbook\css</code> directory, or it can be a custom skin CSS generated with the Oxygen Skin Builder .
Parameters Specific to WebHelp Responsive Output (available only for WebHelp Responsive and WebHelp Responsive with Feedback systems)	
-Dwebhelp.fragment.after.body	Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the body in the output.
-Dwebhelp.fragment.after.logo_and_title	Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the logo and title in the output.

-Dwebhelp.fragment.after.main.page.search	Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the search field in the output.
-Dwebhelp.fragment.after.toc_or_tiles	Specifies an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the table of contents or tiles in the main page of the output.
-Dwebhelp.fragment.after.top_menu	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed after the top menu in the output.
-Dwebhelp.fragment.before.body	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the body in the output.
-Dwebhelp.fragment.before.logo_and_title	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the logo and title in the output.
-Dwebhelp.fragment.before.main.page.search	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the search field in the output.
-Dwebhelp.fragment.before.toc_or_tiles	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the table of contents or tiles in the main page of the output.
-Dwebhelp.fragment.before.top_menu	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed before the top menu in the output.
-Dwebhelp.fragment.footer	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed as the footer in the output.
-Dwebhelp.fragment.head	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed as the header in the output.
-Dwebhelp.fragment.welcome	Specify an XML fragment (or a path to a file that contains well formed XML content) that will be displayed as a welcome message (or title) in the output.
-Dwebhelp.show.breadcrumb	Specifies if the breadcrumb component will be presented in the output. The default value is yes .
-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 <i>tree</i> 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.

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.hdr` parameter, use:

```
-Dargs.hdr=[HEADER_FILE_DIR]
```

where `[HEADER_FILE_DIR]` is the location of the directory that contains the header file.

Database Configuration for DITA WebHelp Systems with Feedback

This topic explains where to find instructions for configuring the database that contains the user comments for a **DITA WebHelp Classic with Feedback** or **DITA WebHelp Responsive with Feedback** system.

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.

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.

Integrating the Oxygen XML WebHelp Plugin with the DocBook XSL Distribution

This topic includes the procedure for 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 later
- DocBook XSL 1.78.1 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 8.0](#) or later.
3. Download and unzip on your computer the DocBook XSL distribution.
4. Unzip the Oxygen XML WebHelp distribution package in the DocBook XSL installation directory.
The DocBook XSL directory now contains a new subdirectory named `com.oxygenxml.webhelp` and two new files, `oxygen_custom.xsl` and `oxygen_custom_html.xsl`.

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

This topic explains how to register the license for the Oxygen XML WebHelp plugin for the DocBook XSL distribution.

To register the license for the Oxygen XML WebHelp plugin for the DocBook XSL distribution, follow these steps:

1. Create a .txt file named `license` 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 Developer license key from this file. If the file does not exist, or it contains an invalid license, an error message is displayed.

Upgrading the Oxygen XML WebHelp Plugin for DocBook

This topic describes the procedure for upgrading a 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 installation kit, and unzip it.
3. Copy the `com.oxygenxml.webhelp` directory in `[OXYGEN_INSTALL_DIR]/frameworks/docbook/xsl`.

Running an External DocBook Transformation Using the WebHelp Plugin

This topic explains how to run an external DocBook to WebHelp transformation using the Oxygen XML 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 excerpt 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 Developer 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/stylesheets.css
```

Database Configuration for DocBook WebHelp Classic with Feedback

This topic explains where to find instructions for configuring the database that contains the user comments for a **DocBook WebHelp Classic with Feedback** system.

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.

Chapter

9

Querying Documents

Topics:

- [Running XPath Expressions](#)
- [Working with XQuery](#)

This chapter shows how to query XML documents in Oxygen XML Developer with XPath expressions and the XQuery language.

This chapter shows how to query XML documents in Oxygen XML Developer with XPath expressions and the XQuery language.

Running XPath Expressions

This section covers the views, toolbars, and dialog boxes in Oxygen XML Developer that are dedicated to running XPath expressions.

What is XPath

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.

- `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>.

Oxygen XML Developer 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 Developer provides an XPath toolbar to let you query XML documents fast and easy using XPath expressions.



Figure 320: XPath Toolbar

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 Developer 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 Developer can return a compilation error.

XPath scope menu - Oxygen XML Developer 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.

-  **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.

The following actions are available in the **Settings** menu:

-  **Update on cursor move** - When enabled 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](#) on page 683

XPath/XQuery Builder View

The **XPath/XQuery Builder** view allows you to compose complex XPath and XQuery expressions and execute them over the currently edited XML document. For XPath 2.0 / 3.0, or XQuery expressions, 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 Developer executes the expressions over the XML document in the associated scenario.

If the view is not displayed, it can be opened from the **Window > Show View** menu.

The upper part of the view contains the following actions:

- 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 Developer 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 Developer returns a compilation error.

-  **Execute XPath** button - Press 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](#) in a separate tab
- **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 Developer 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 enabled 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 Developer 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.
 -  **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.

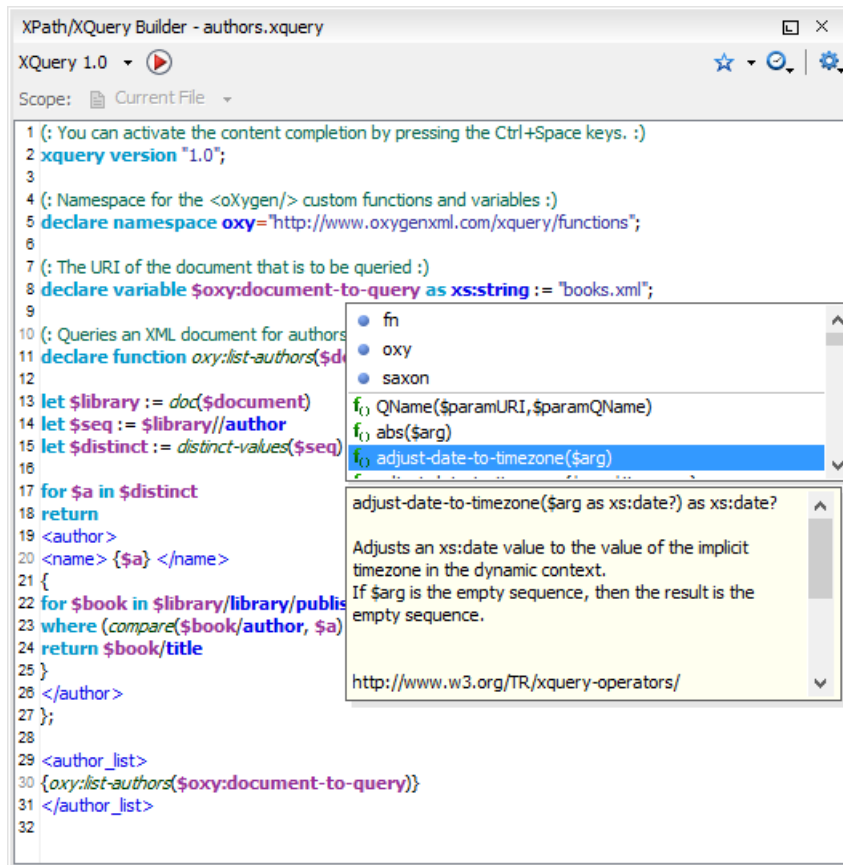


Figure 321: XPath/XQuery Builder View

While you edit an XPath or XQuery expression, Oxygen XML Developer 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 Highlight** - 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](#) on page 683

XPath Expression Results

When you run an XPath expression, Oxygen XML Developer displays the results of its execution in the [Results view](#).

This view contains the following columns:

- **Description** - The result that Oxygen XML Developer displays when you run an XPath expression.
- **Resource** - The name of the document that you run the XPath expression on.
- **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 Developer displays **Not available** in the **Location** column. Oxygen XML Developer displays the results in a valid XPath expression format.

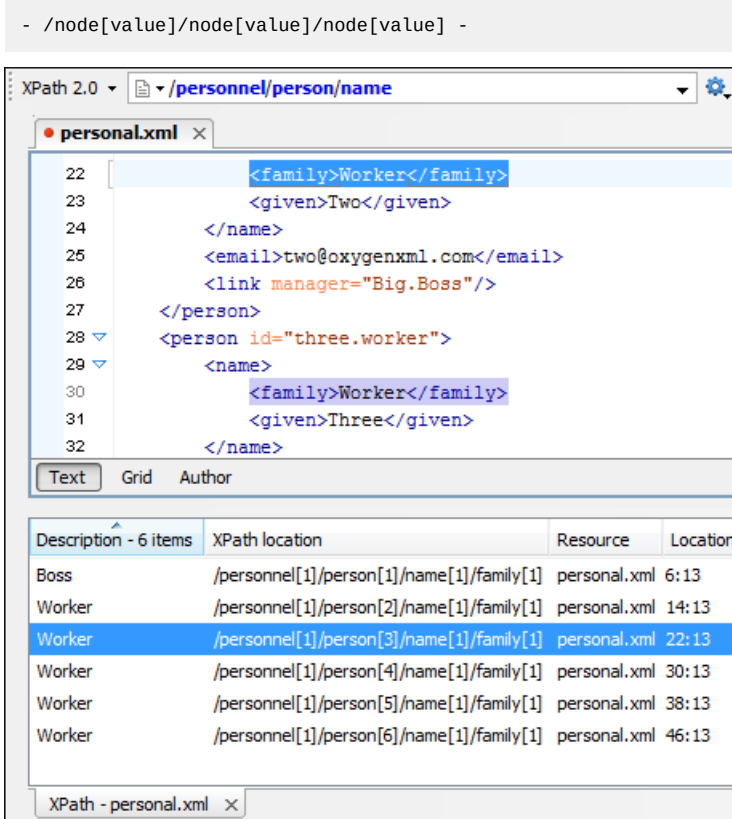


Figure 322: XPath Results Highlighted in Editor Panel with Character Precision

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 Developer 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.

Catalogs

The evaluation of the XPath expression tries to resolve the locations of documents referenced in the expression through the *XML catalogs*. These catalogs are *configured in the Preferences* pages and *the current XInclude preferences*.

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 XML 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/mydocsdire?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 Developer 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 XQuery

This section explains how to edit and run XQuery queries in Oxygen XML Developer.

What is XQuery

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.

Syntax Highlight and Content Completion

To [create an XQuery document](#), select **File > New (Ctrl (Meta on Mac OS)+N)** and when the **New** document wizard appears, select XQuery entry.

Oxygen XML Developer provides syntax highlight for keywords and all known XQuery functions and operators. A **Content Completion Assistant** is also available and can be 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. For some supported database engines such as 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.

The extension functions included in the Saxon product 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 9.6.0.7 PE or Saxon 9.6.0.7 EE.
- The edited file has a validation scenario associated that use as validation engine Saxon 9.6.0.7 PE or Saxon 9.6.0.7 EE.
- The validation engine specified in [Preferences](#) is Saxon 9.6.0.7 PE or Saxon 9.6.0.7 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.

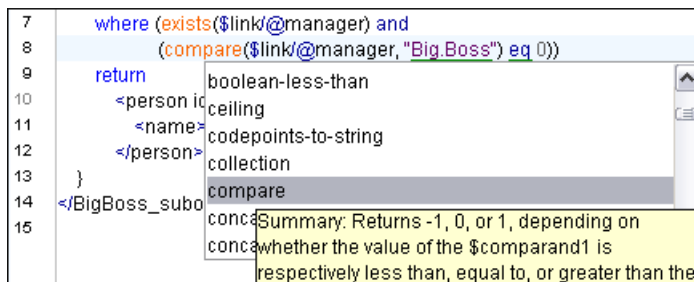


Figure 323: XQuery Content Completion

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 from the **Window > Show View** menu.

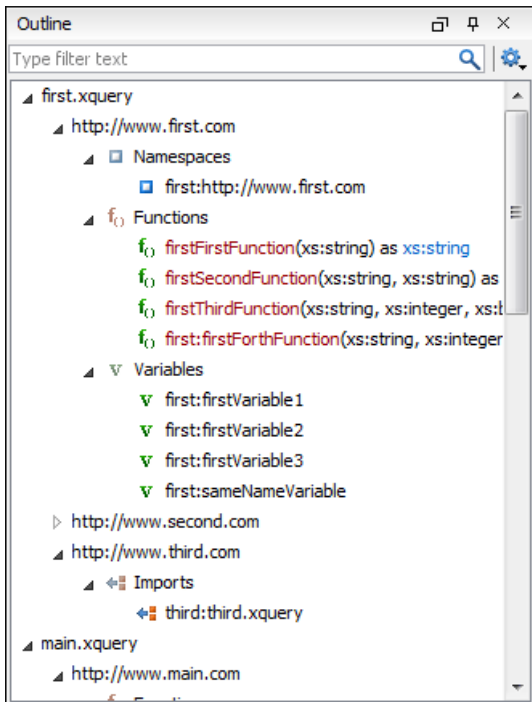


Figure 324: XQuery Outline View

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.

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 **XSLT/XQuery Input**. If the view is not displayed, it can be opened from the **Window > Show View** menu. The tree nodes represent the elements of the documents.

XQuery Input View

You use the **XQuery Input** view to drag and drop a node into the editing area to quickly insert XQuery expressions.

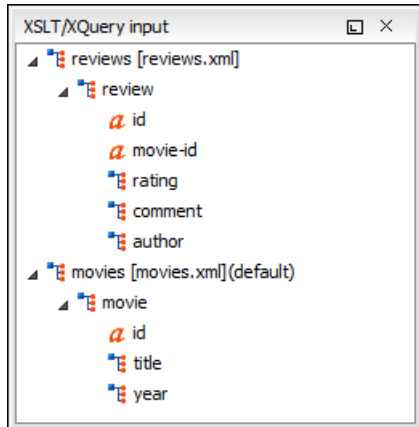


Figure 325: XQuery Input View

Create FLWOR by Drag and Drop

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>
```

and

```
<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, a pop-up menu will be displayed.

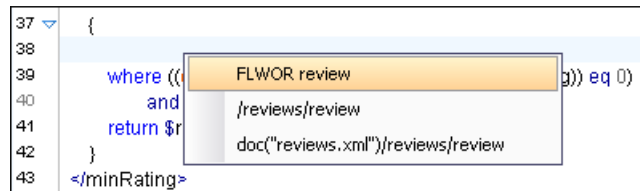


Figure 326: XQuery Input Drag and Drop Pop-up Menu

Select **FLWOR rating** and the result document will be:

```

37 {
38   for $review in doc("reviews.xml")/reviews/review
39   return
40   where ((compare($rew/rating/text(), string($minRating)) eq 0)
41     and ($rev/@movie-id = $movie/@id))
42   return $rew/author
43 }
44 </minRating>

```

Figure 327: XQuery Input Drag and Drop Result

XQuery Validation

With Oxygen XML Developer, you can validate your documents before using them in your transformation scenarios. The validation uses the Saxon 9.6.0.7 PE processor or the 9.6.0.7 EE, IBM DB2, eXist, Berkeley DB XML, Documentum xDB (X-Hive/DB) 10, or MarkLogic (version 5 or newer) 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.

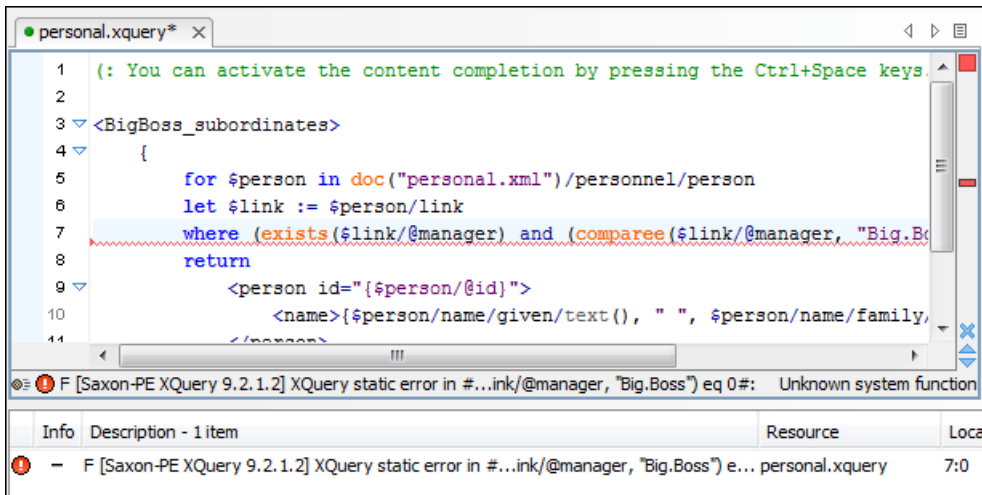


Figure 328: XQuery Validation



Note: If you choose a processor that does not support XQuery validation, Oxygen XML Developer 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 Developer preferences.

When you open an XQuery document from a connection that supports validation (for example, MarkLogic, or eXist), by default Oxygen XML Developer uses this connection for validation. If you open an XQuery file using a MarkLogic connection, the validation better resolves imports.

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 9.6.0.7 HE, Saxon 9.6.0.7 PE, Saxon 9.6.0.7 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 9.6.0.7 EE processor also supports XQuery 3.0 transformations.

XQJ Transformers

This section describes the necessary procedures before running an XQJ transformation.

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 Developer 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 Developer 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.

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 this 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.

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.

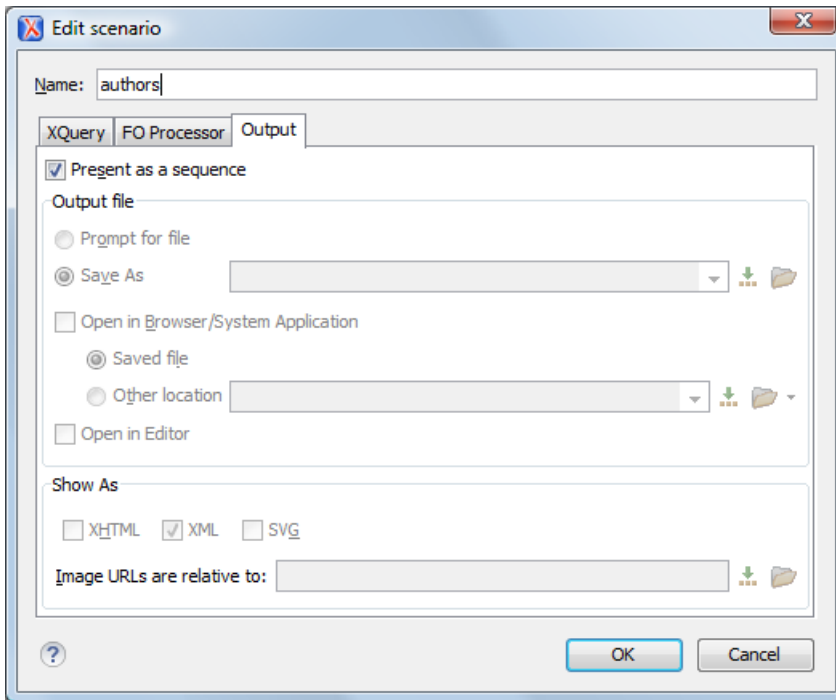


Figure 329: XQuery transformation result displayed in Sequence view

A chunk of the XQuery transformation result is displayed in the **Sequence** view.

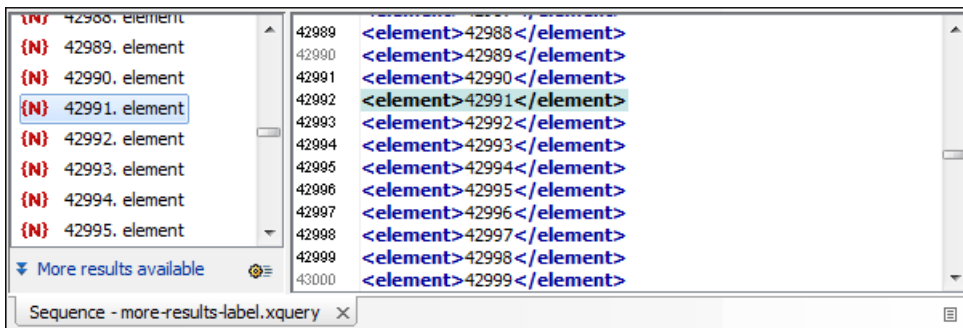


Figure 330: XQuery transformation result displayed in Sequence view

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 9.6.0.7 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 <i>strip whitespaces</i> operation is handled. You can choose one of the following values: • All ("all") - Strips <i>all</i> whitespace text nodes from source documents before any further processing, regardless of any <code>xml:space</code> attributes in the source document. • Ignore ("ignorable") - Strips all <i>ignorable</i> whitespace text nodes from source documents before any further processing, regardless of any <code>xml:space</code> 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 <i>no</i> 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 enabled (default value), Saxon runs the XQuery transformation with the XQuery 3.0 support.
Initializer class	Equivalent to the <i>-init</i> Saxon command-line argument. The value is the name of a user-supplied class that implements the <code>net.sf.saxon.lib.Initializer</code> 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 9.6.0.7 Professional Edition (PE) and Enterprise Edition (EE) only:

Use a configuration file ("-config")	Sets a Saxon 9.6.0.7 configuration file that is used for XQuery transformation and validation scenarios.
Allow calls on extension functions ("-ext")	If checked, calls on external functions are allowed. Checking 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 9.6.0.7 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 <code>document()</code> or similar functions. It can have the following values: • Schema validation ("strict") - This mode requires an XML Schema and enables parsing the source documents with strict schema-validation enabled. • Lax schema validation ("lax") - If an XML Schema is provided, this mode enables 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. Enabling 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 enable 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#!javadococ .
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 checked, backup versions for any XML files updated with an XQuery Update are generated. This option is available when the Enable XQuery update option is enabled.

Updating XML Documents using XQuery

Using the bundled Saxon 9.6.0.7 EE XQuery processor Oxygen XML Developer 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 9.6.0.7 EE as a transformer in the scenario associated with the XQuery files containing update statements and Oxygen XML Developer will notify you if the update was successful.

Using XQuery Update to modify a tag name in an XML file

```
rename node doc("books.xml")//publisher[1]//book[1] as "firstBook"
```

Chapter

10

Working with Archives

Topics:

- [Browsing and Modifying Archives](#)
- [Working with EPUB](#)
- [Editing Files From Archives](#)

Describes the archive support available in Oxygen XML Developer.

Oxygen XML Developer offers the means to manipulate files directly from ZIP type 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

This allows you to modify, transform, or validate files directly from OOXML or ODF packages. The structure and content of an EPUB book, OOXML file, or ODF file [can be opened, edited, and saved](#), like any other ZIP archive.

You can transform, validate, and perform many other operations on files directly from an archive. When selecting a URL for a specific operation (such as a transformation or validation), you can click the  **Browse for archived file** button to navigate and choose the file from a particular archive.

Browsing and Modifying Archives

You can open an archive in the **Archive Browser** view by doing one of the following:

- Open an archive from the [Project view](#).
- Choose an archive in the Oxygen XML Developer file chooser dialog box.
- Drag an archive from the 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.

 **Important:** If a file is not recognized by Oxygen XML Developer as a supported archive type, you can add it from the [Archive preferences page](#).

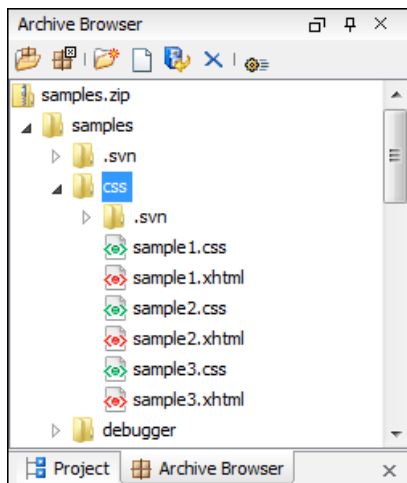


Figure 331: Browsing an Archive

Archive Browser Toolbar

The following actions are available on the **Archive Browser** toolbar:



Reopen

You can use this drop-down to reopen recently edited archives. Apart from the history of the recently edited archives, the drop-down also contains the **Clear history** and



Open Archive actions.



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 **Project view** and dropping them in the **Archive Browser** view.



Delete

Deletes the selected resource in the browsed archive.



Archive Options

Opens the [Archive preferences page](#).

The following additional operations are available from the **Archive Browser** contextual menu:



Open

Opens a resource from the archive in the editor.



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.

 **Find/Replace in Files**

Allows you to search for and replace specific pieces of text inside the archive.

Cut

Cut the selected archive resource.

Copy

Copy the selected archive resource.

Paste

Paste a file or folder into the archive.



Note: You can add files in the archive by copying the files from the **Project view** and paste them into the **Archive view**.

Delete

Remove a file or folder from archive.

Copy location

Copies the URL location of the selected resource.

 **Refresh**

Refreshes the selected resource.

Properties

Views properties for the selected resource.

Working with EPUB

EPUB is a free and open electronic book standard by the International Digital Publishing Forum (IDPF). It was designed for *reflowable content*, meaning that the text display can be optimized for the particular display device used by the reader of the EPUB-formatted book. Oxygen XML Developer supports both EPUB 2.0 and EPUB 3.0.

EPUB files are opened in the **Archive Browser** view, exposing all their internal components:

- Document content (XHTML and image files).
- Packaging files.
- Container files.

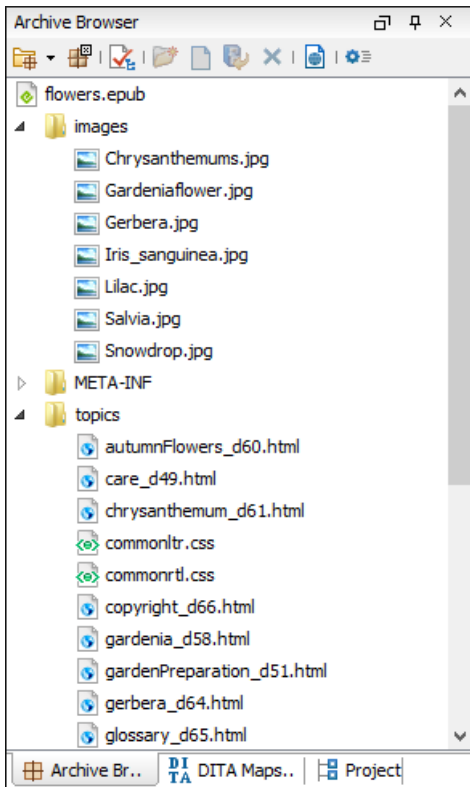


Figure 332: EPUB file displayed in the Archive Browser view

Here you can edit, delete and add files that compose the EPUB structure. To check that the EPUB file you are editing is valid, invoke the **Validate and Check for Completeness** action. Oxygen XML Developer 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. All errors found during validation are displayed in a separate tab in the **Errors** view.



Note: Invoke the **Open in System Application** action to see how the EPUB is rendered in your system default EPUB reader application.



Note: All changes made to the structure of an EPUB, or to the contents of the files inside an EPUB are immediately saved.

To watch our video demonstration about the EPUB support in Oxygen XML Developer, go to <http://oxygenxml.com/demo/Epub.html>.

Related information

[The Archive Browser View](#) on page 696

Create an EPUB

To begin writing an EPUB file from scratch, do the following:

1. Go to **File > New**, press **Ctrl + N (Command + N on OS X)** on your keyboard. or click **New** on the main toolbar.
2. Choose **EPUB Book** template. Click **Create**. Choose the name and location of the file. Click **Save**.
A skeleton EPUB file is saved on disk and open in the **Archive Browser** view.
3. Use the **Archive Browser** view specific actions to edit, add and remove resources from the archive.
4. Use the **Validate and Check for Completeness** action to verify the integrity of the EPUB archive.

Publish to EPUB

Oxygen XML Developer includes built-in support for publishing DocBook and DITA XML documents directly to EPUB.

1. Open the **Configure Transformation Scenario(s)** dialog box and select a predefined transformation scenario. To publish from DITA, select the **DITA Map EPUB** transformation scenario. To publish from DocBook select the **DocBook EPUB** transformation scenario.
2. Click **Apply associated** to run the transformation scenario.

Editing Files From Archives

You can open and edit files directly from an archive using the **Archive Browser** view. When saving the file back to 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](#).



Note: All changes made to the structure of an archive, or to the contents of the files inside an archive are immediately saved.

Chapter 11

Databases and CMS Integration

Topics:

- [*Working with Databases*](#)
- [*Content Management System \(CMS\) Integration*](#)

This chapter provides information about the support for various databases and Content Management Systems.

Oxygen XML Developer provides support for various databases and content management systems.

Working with Databases

Details about working with various database types in Oxygen XML Developer.

XML is a storage and interchange format for structured data and is supported by all major database systems. Oxygen XML Developer 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 Developer helps users with browsing, content editing, importing from databases, using XQuery with databases, SQL execution, and generating XML Schema from a database structure.

Related information

[Integration with Microsoft SharePoint](#) on page 754

Data Source Explorer View

The **Data Source Explorer** view displays your database connections. 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 Developer supports multiple simultaneous database connections and the connection tree in the **Data Source Explorer** view provides an easy method for browsing them.

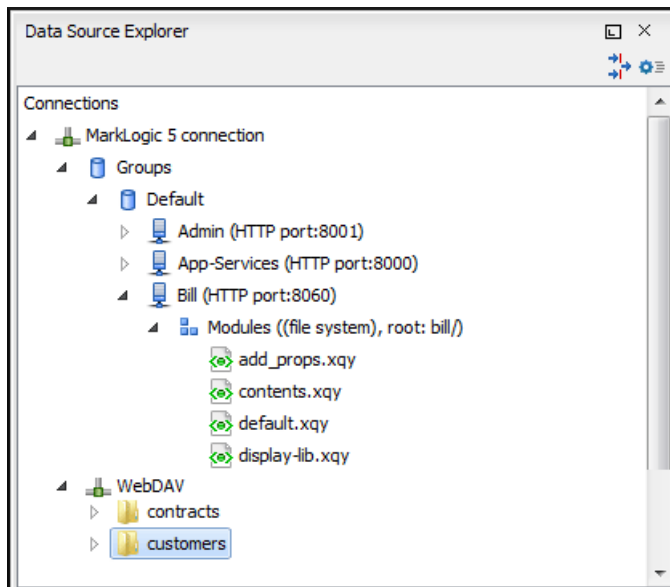


Figure 333: Data Source Explorer View

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.

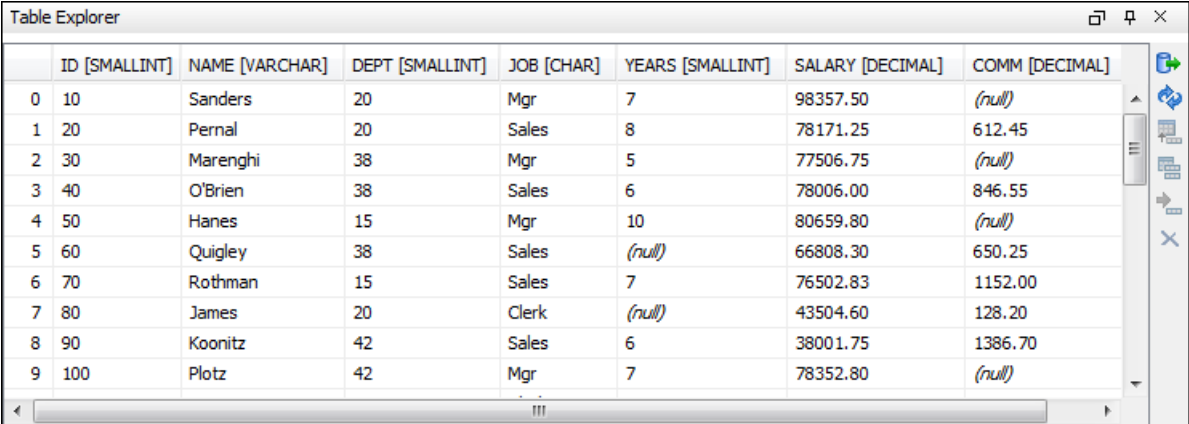
For more information on the specific actions that are available, see the topics in this section for each specific type of database.

Related information

[Data Sources Preferences](#) on page 124

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 Developer attempts to update the database with the new cell content.



	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 334: Table Explorer View

You can sort the content of a table by one of its columns by clicking its column header.

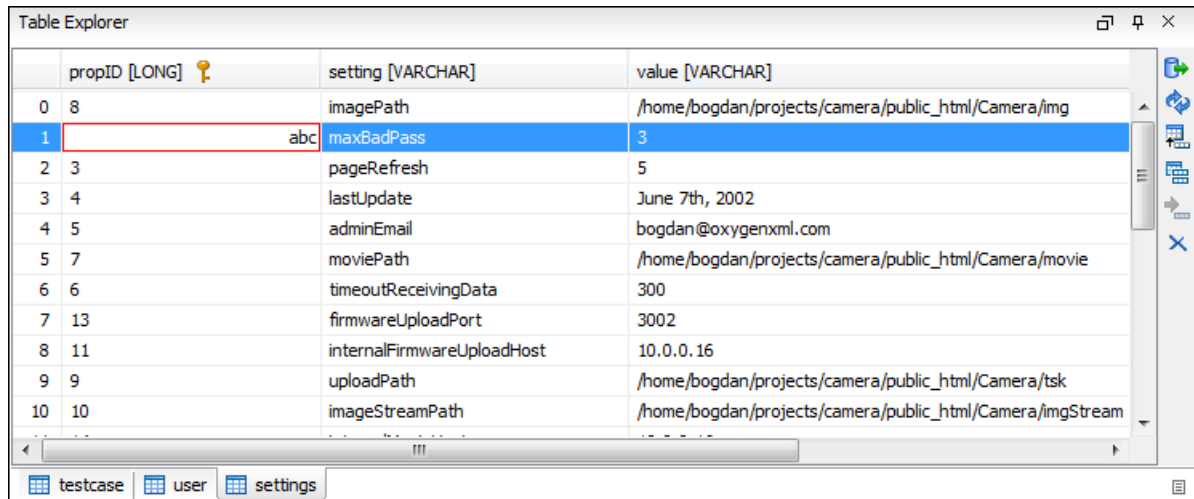
Note the following:

- 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).

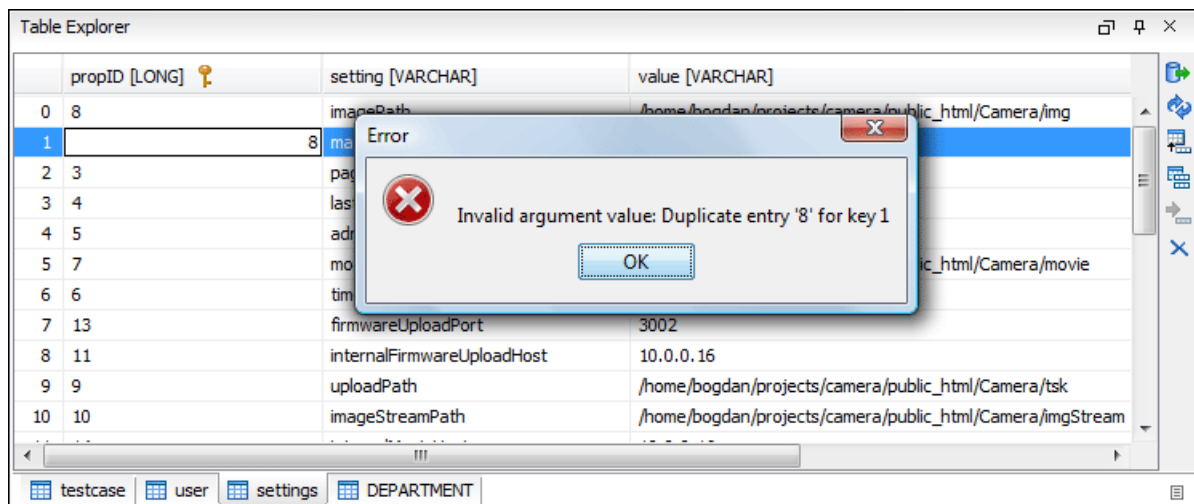
- If the content of the edited cell does not belong to the data type of the column, the cell is marked by a red square and remains in an editing state until a correct value is inserted. For example, in the following figure **propID** contains **LONG** values. If a character or string is inserted, the cell will look like this:



	propID [LONG] 🔑	setting [VARCHAR]	value [VARCHAR]
0	8	imagePath	/home/bogdan/projects/camera/public_html/Camera/img
1		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

Figure 335: 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:



	propID [LONG] 🔑	setting [VARCHAR]	value [VARCHAR]
0	8	imagePath	/home/bogdan/projects/camera/public_html/Camera/img
1	8	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

Figure 336: 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 disabled 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 702

Database Connection Support

Oxygen XML Developer 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 Developer also includes a [Database perspective](#) that helps you to manage databases.

The database support in Oxygen XML Developer 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 Developer offers support for the most commonly used relational databases, including:

- IBM DB2
- Oracle 11g
- Microsoft SQL Server
- PostgreSQL
- MySQL

Oxygen XML Developer 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 Developer, go to http://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 Developer 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 Developer, go to http://www.oxygenxml.com/demo/Author_Database_XML_Native.html.

Related information

[WebDAV Connections](#) on page 741

[Integration with Microsoft SharePoint](#) on page 754

IBM DB2 Database Connections

Oxygen XML Developer includes support for IBM DB2 database connections. Oxygen XML Developer 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.

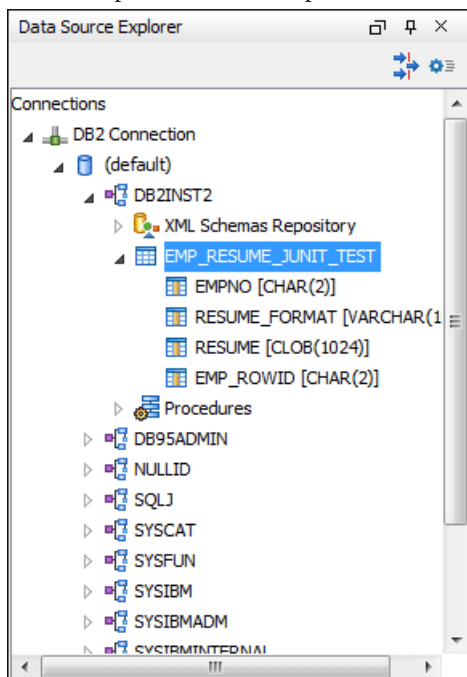


Figure 337: IBM DB2 Database Connection

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 Developer 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 from the **Window > Show View** menu) or switch to the **Database Perspective** (available from **Window > Open Perspective > Database**).

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. [Open the Preferences dialog box \(Options > Preferences\)](#) and go to **Data Sources**.
2. Click the **+** **New** button in the **Data Sources** panel.

The dialog box for configuring a data source is opened.

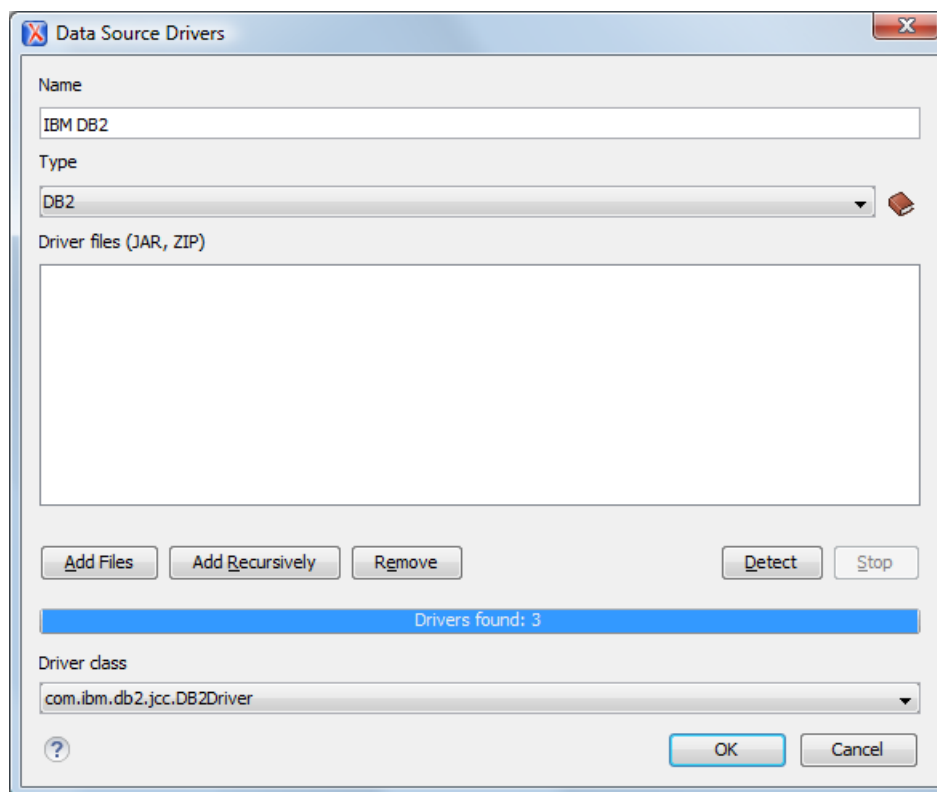


Figure 338: Data Source Drivers Configuration Dialog Box

3. Enter a unique name for the data source.
4. Select **DB2** in the driver **Type** drop-down menu.
5. Add the driver files for IBM DB2 using the **Add Files** button.

The IBM DB2 driver files are:

- db2jcc.jar
- db2jcc_license_cisuz.jar
- db2jcc_license_cu.jar

The **Driver files** section lists [download links for database drivers](#) that are necessary for accessing IBM DB2 databases in Oxygen XML Developer.

6. Select the most appropriate **Driver class**.
7. Click the **OK** button to finish the data source configuration.
8. 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 <http://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. In the **Connections** panel, click the **+ New** button.

The dialog box for configuring a database connection is displayed.

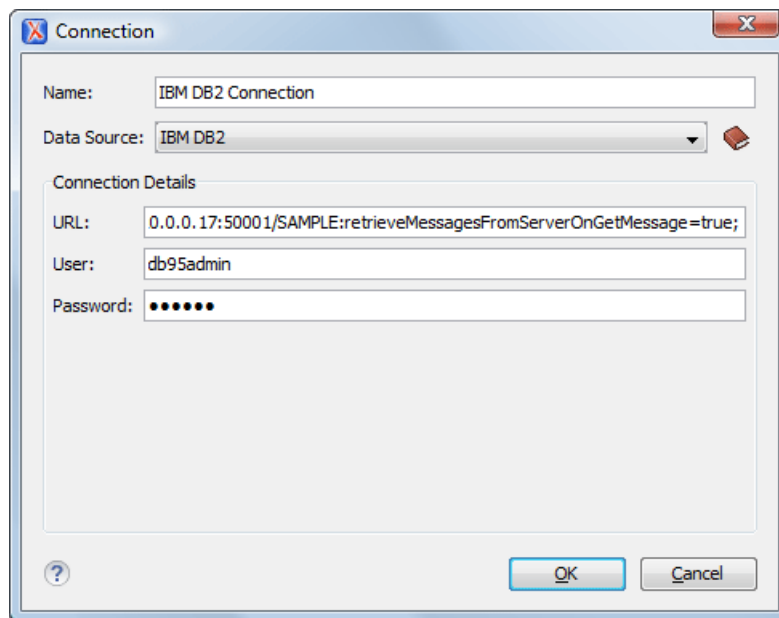


Figure 339: Connection Configuration Dialog Box

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 configuration of the database connection.

To watch our video demonstration about running XQuery against an IBM DB2 Pure XML database, go to <http://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 Developer.

Microsoft SQL Server Database Connections

Oxygen XML Developer includes support for Microsoft SQL Server database connections. Oxygen XML Developer 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 from the **Window > Show View** menu) or switch to the **Database Perspective** (available from **Window > Open Perspective > Database**).

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. [Open the Preferences dialog box \(Options > Preferences\)](#) and go to **Data Sources**.
2. Click the  **New** button in the **Data Sources** panel.

The dialog box for configuring a data source is opened.

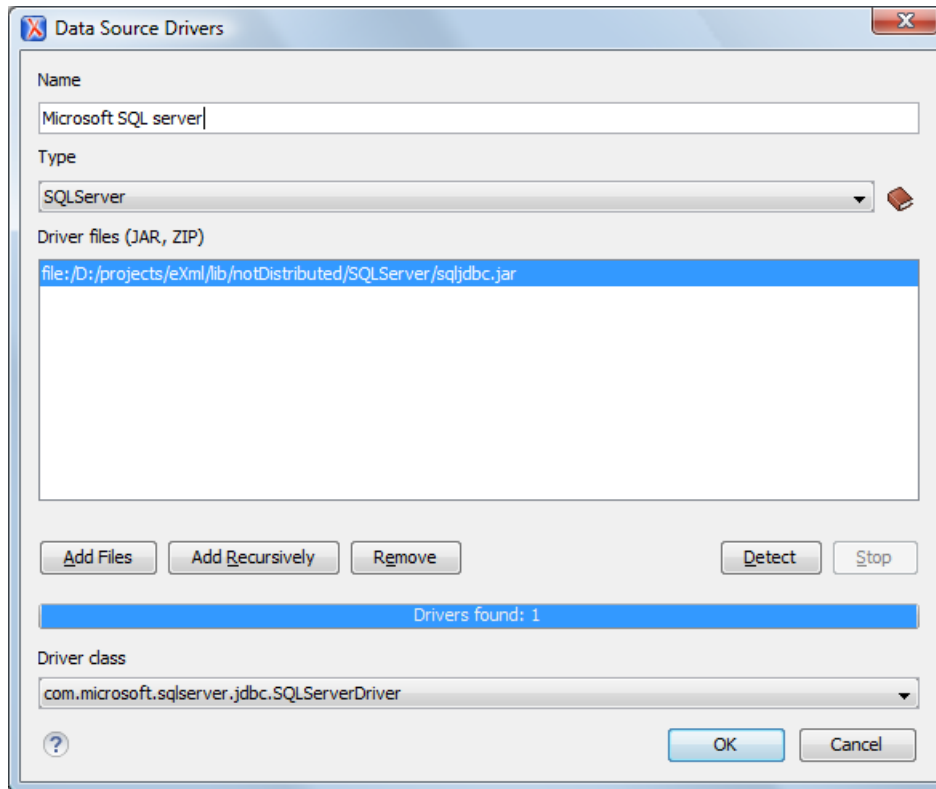


Figure 340: Data Source Drivers Configuration Dialog Box

3. Enter a unique name for the data source.
4. Select *SQLServer* in the driver **Type** drop-down menu.
5. Add the Microsoft SQL Server driver file using the **Add Files** button.

The SQL Server driver file is called `sqljdbc.jar`. In the **The Driver files** section lists [download links for database drivers](#) that are necessary for accessing Microsoft SQL Server databases in Oxygen XML Developer.

6. Select the most appropriate **Driver class**.
7. Click the **OK** button to finish the data source configuration.
8. 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. In the **Connections** panel, click the **+ New** button.

The dialog box for configuring a database connection is displayed.

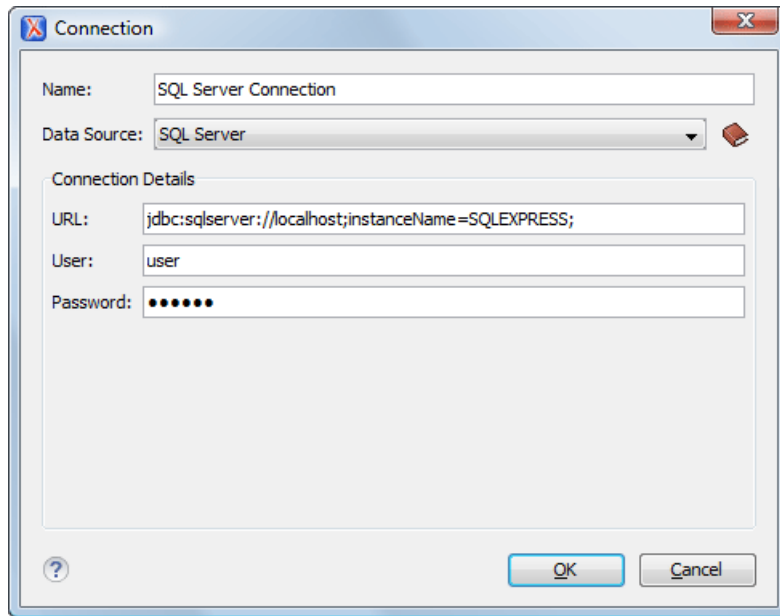


Figure 341: Connection Configuration Dialog Box

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 configuration of the database connection.

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.
	Add	Adds a new schema to the XML Schema files.
	Unregister	Removes the selected schema from the XML Schema Repository.
 Schema Level Nodes	View	Opens the selected schema in Oxygen XML Developer.

Oracle Database Connections

The Oracle database is a common relational type of database system. Oxygen XML Developer 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 Developer 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.

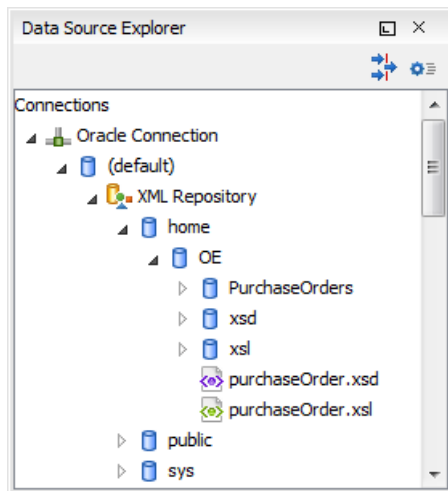


Figure 342: Oracle Database Connection

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 from the **Window > Show View** menu) or switch to the **Database Perspective** (available from **Window > Open Perspective > Database**).

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. Open the **Preferences** dialog box (**Options > Preferences**) and go to **Data Sources**.
2. Click the **+ New** button in the **Data Sources** panel.

The dialog box for configuring a data source is opened.

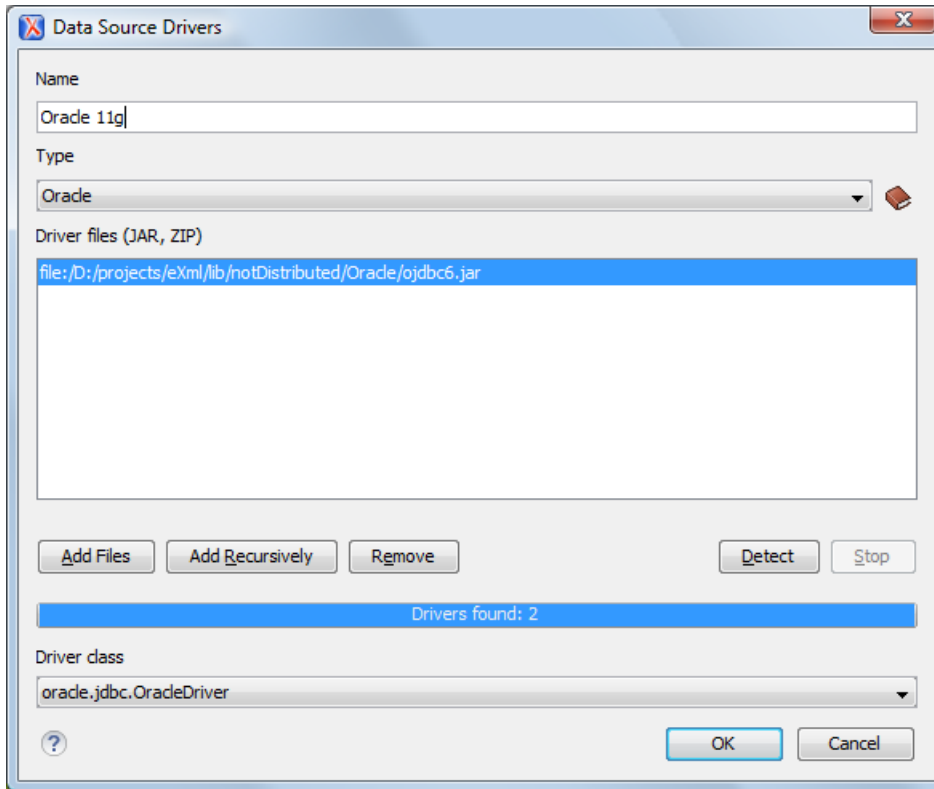


Figure 343: Data Source Drivers Configuration Dialog Box

3. Enter a unique name for the data source.
4. Select **Oracle** in the driver **Type** drop-down menu.
5. Add the Oracle driver file using the **Add Files** button.

The Oracle driver file is called `ojdbc5.jar`. The **Driver files** section lists [download links for database drivers](#) that are necessary for accessing Oracle databases in Oxygen XML Developer.

6. Select the most appropriate **Driver class**.
7. Click the **OK** button to finish the data source configuration.
8. 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. In the **Connections** panel, click the **+ New** button.

The dialog box for configuring a database connection is displayed.

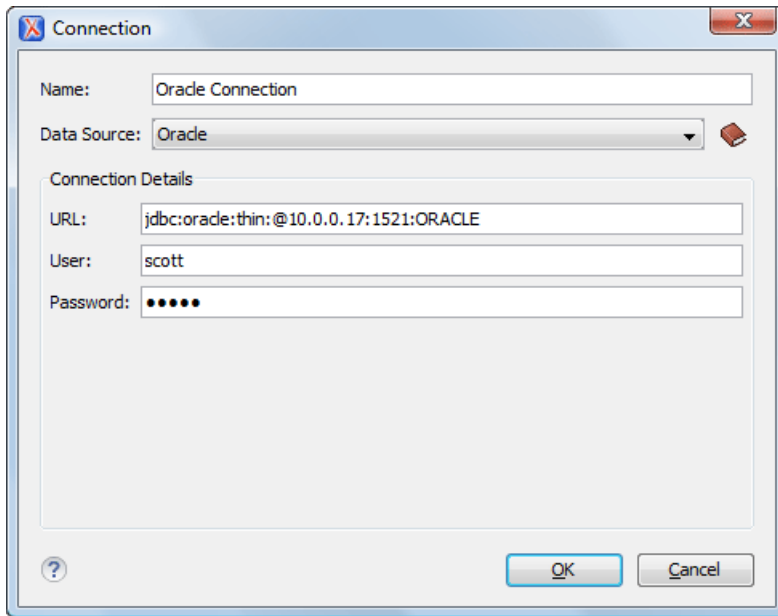


Figure 344: Connection Configuration Dialog Box

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 configuration of the database connection.

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.

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 Developer includes support for PostgreSQL database connections. Oxygen XML Developer 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.

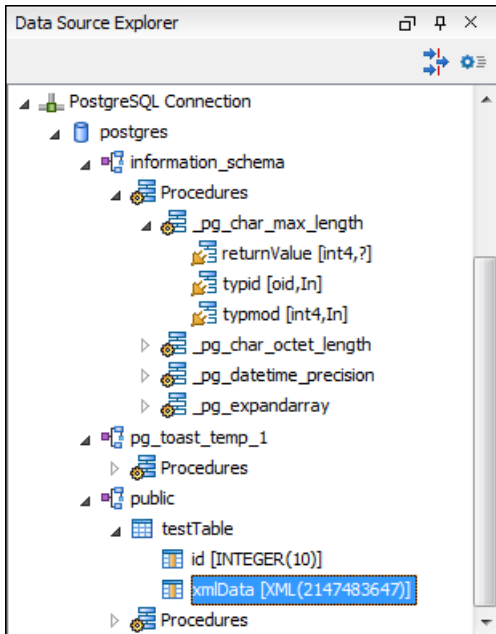


Figure 345: PostgreSQL Database Connection

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 from the **Window > Show View** menu) or switch to the **Database Perspective** (available from **Window > Open Perspective > Database**).

How to Configure PostgreSQL 8.3 Data Source Drivers

To configure a data source for connecting to a PostgreSQL server, 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.
The dialog box for configuring a data source is opened.
3. Enter a unique name for the data source.
4. Select *PostgreSQL* in the driver **Type** drop-down list.
5. Add the PostgreSQL driver file using the **Add Files** button.

The PostgreSQL driver file is called `postgresql-8.3-603.jdbc3.jar`. The **Driver files** section lists [download links for database drivers](#) that are necessary for accessing PostgreSQL databases in Oxygen XML Developer.

6. Select the most appropriate **Driver class**.
7. Click the **OK** button to finish the data source configuration.

8. 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. In the **Connections** panel, click the **+ New** button.

The dialog box for configuring a database connection is displayed.

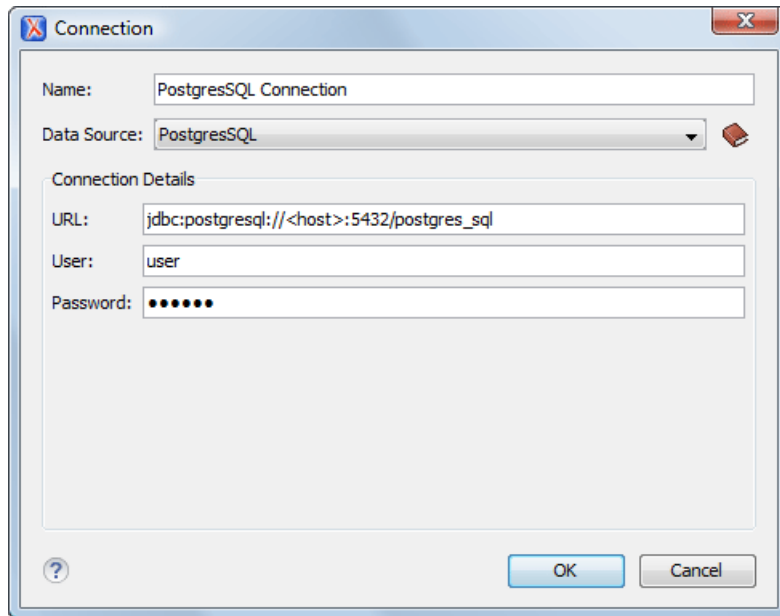


Figure 346: Connection Configuration Dialog Box

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 configuration of the database connection.

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 Developer includes support for Berkeley DB XML database connections. Oxygen XML Developer 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>.

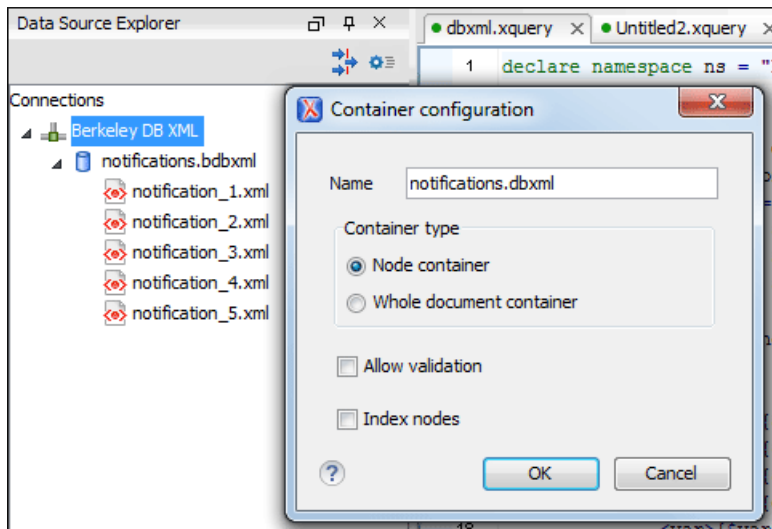


Figure 347: Berkeley DB XML Connection

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 from the **Window > Show View** menu) or switch to the **Database Perspective** (available from **Window > Open Perspective > Database**).

How to Configure Berkeley DB XML Data Source Drivers

Oxygen XML Developer 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** button to add the Berkeley DB driver files.

The driver files for the Berkeley DB database are the following:

- `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 Developer 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 one of the previously configured data sources 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 checked, 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.

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:

	Configure	Opens the Data Sources preferences page where you can configure both data sources and connections.
	Database Sources	
	Disconnect	Stops the connection.

New Collection Opens a **Container configuration** dialog box that allows you to add a new container in the repository.

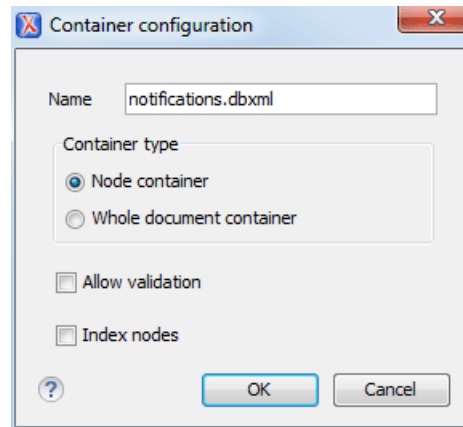


Figure 348: Container Configuration Dialog Box

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 checked, documents will be validated when they are loaded into the container. The default behavior is to not validate documents.
- **Index nodes** - If checked, 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**

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.

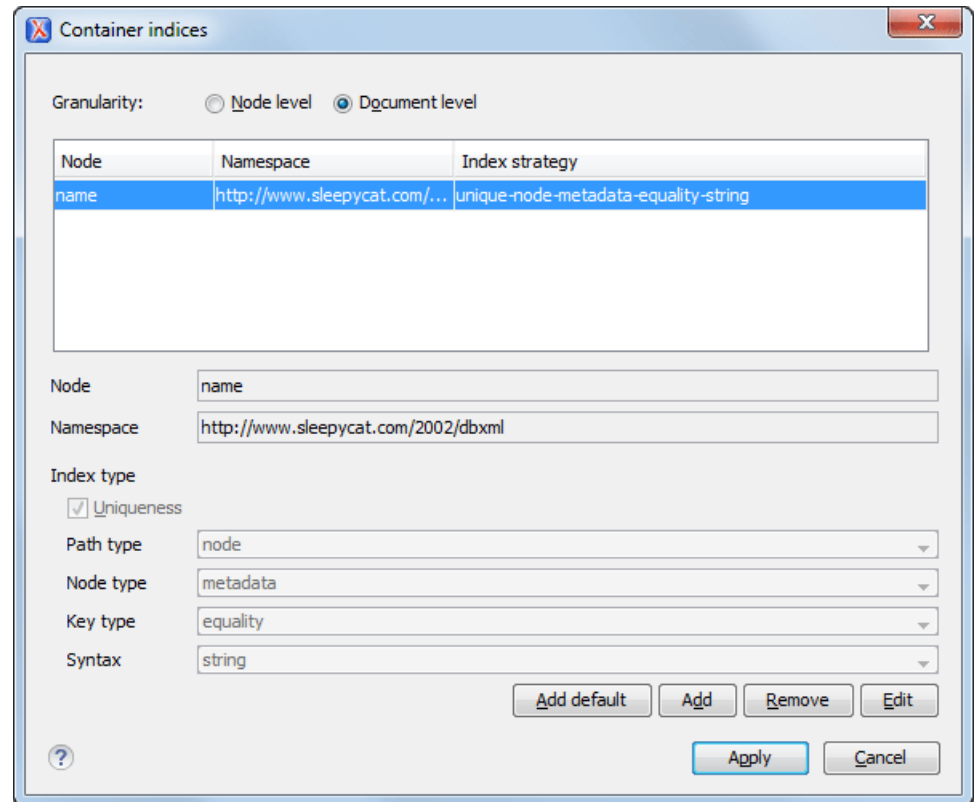


Figure 349: Container Indices Dialog Box

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`.

Resource Level Nodes		Refresh	Performs a refresh on the selected node.
		Properties	Shows various properties of the current container.
		Find/Replace in Files	Allows you to find and replace text in multiple files from the connection.
		Open	Opens the selected resource in the editor.
		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	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 Developer XQuery Debugger. *The same restrictions and peculiarities* apply for the Berkeley debugger as for the MarkLogic debugger.

eXist Database Connections

Oxygen XML Developer includes support for eXist database connections. Oxygen XML Developer 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.

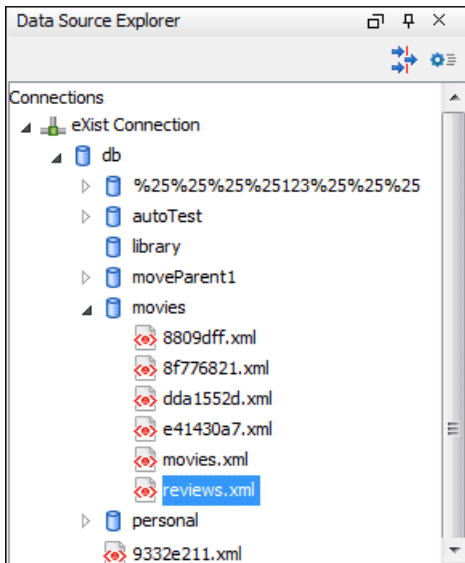


Figure 350: eXist Database Connection

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 Developer 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 from the **Window > Show View** menu) or switch to the **Database Perspective** (available from **Window > Open Perspective > Database**).
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 from the **Window > Show View** menu) or switch to the **Database Perspective** (available from **Window > Open Perspective > Database**).

How to Configure eXist Data Source Drivers

Oxygen XML Developer supports eXist database server versions up to and including version 2.2. 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** button to add the eXist driver files.

The following driver files should be added in the dialog box for setting up the eXist datasource. 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 Developer.

- exist.jar
- lib/core/xmlldb.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 http://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 one of the previously configured data sources 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.
- To watch our video demonstration about running XQuery against an eXist XML database, go to http://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	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.

 Resource Level Nodes	 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	Allows you to find and replace text in multiple files from the connection.
	Open	Opens the selected resource in the editor.
	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	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 Developer 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/blogfile/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 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 Developer 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*.

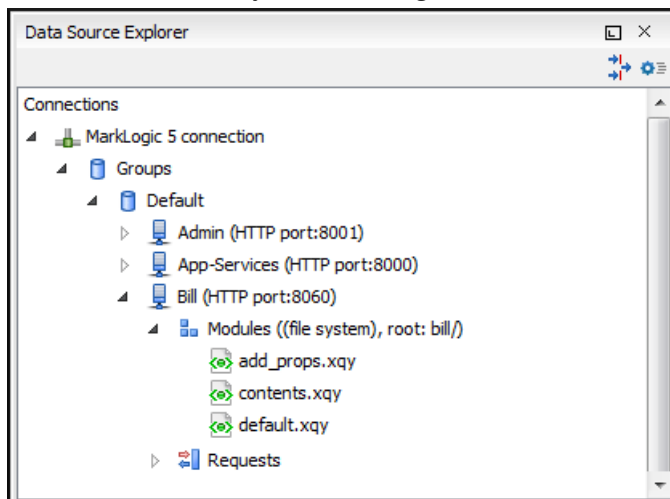


Figure 351: MarkLogic Connection in Data Source Explorer

Configuring a MarkLogic Database Connection

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 from the **Window > Show View** menu) or switch to the **Database Perspective** (available from **Window > Open Perspective > Database**).

Related information

[MarkLogic Development in Oxygen XML Developer](#) on page 727

How to Configure MarkLogic Data Source Drivers

Available in the Enterprise edition only.



Note: Oxygen XML Developer supports MarkLogic version 4.0 or later.

To configure a data source for MarkLogic, follow this procedure:

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 *MarkLogic* from the driver **Type** drop-down list.
5. Click the **Add** button to add the MarkLogic driver file (`marklogic-xcc-{server_version}`, where `{server_version}` is the MarkLogic server version.)

You can download the driver file from: <http://community.marklogic.com/download>.

6. Click the **OK** button to finish the data source configuration.
7. Continue on to [configure your MarkLogic Connection](#).

How to Configure a MarkLogic Connection

Available in the Enterprise edition only.



Note: Oxygen XML Developer 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 one of the previously configured data sources 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 Developer 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 set the URL used for browsing the MarkLogic database in the **Data Source Explorer** view in the **WebDAV URL** field.

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.

MarkLogic Development in Oxygen XML Developer

The Oxygen XML Developer 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 Developer 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 Developer 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 Developer 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 728

[Configuring a MarkLogic Database Connection](#) on page 726

Debugging with MarkLogic

Oxygen XML Developer 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 Developer 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 Developer 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 Developer 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 729.

Remote Debugging with MarkLogic

Oxygen XML Developer allows you to debug remote applications that use XQuery (for example, web applications that trigger XQuery executions). Oxygen XML Developer 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 Developer 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 <http://oxygenxml.com/demo/XQueryDebuggerforMarkLogic.html>.

Related information

MarkLogic Development in Oxygen XML Developer on page 727

Configuring a MarkLogic Database Connection on page 726

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 725

[MarkLogic Development in Oxygen XML Developer](#) on page 727

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	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.
	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 726

[MarkLogic Development in Oxygen XML Developer](#) on page 727

[Debugging with MarkLogic](#) on page 728

Documentum xDB (X-Hive/DB) 10 Database Connections

Oxygen XML Developer includes support for Documentum xDB (X-Hive/DB) 10 database connections. Oxygen XML Developer allows you to browse the structure of a Documentum xDB (X-Hive/DB) 10 database in the [Data Source Explorer](#) and perform various operations on the resources in the repository.

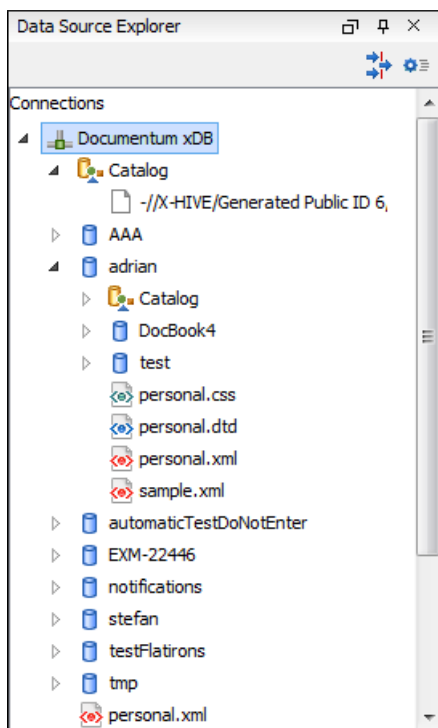


Figure 352: Documentum xDB (X-Hive/DB) 10 Connection

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 from the **Window > Show View** menu) or switch to the **Database Perspective** (available from **Window > Open Perspective > Database**).

How to Configure Documentum xDB (X-Hive/DB) 10 Data Source Drivers

Available in the Enterprise edition only.

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 Developer. 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 one of the previously configured data sources 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) Check 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.

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

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

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

 Resource Level Nodes	 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	Allows you to find and replace text in multiple files from the connection.
	Open	Opens the selected resource in the editor.
	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	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 checked, 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

Cannot save the file. DTD factory class `org.apache.xerces.impl.dv.dtd.DTDDVFactoryImpl` does not extend from `DTDDVFactory`

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-shrunk.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 Developer includes support for MySQL database connections.

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 from the **Window > Show View** menu) or switch to the **Database Perspective** (available from **Window > Open Perspective > Database**).

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. Open the **Preferences dialog box (Options > Preferences)** and go to **Data Sources**.
2. Click the **+** **New** button in the **Data Sources** panel.

The dialog box for configuring a data source is opened.

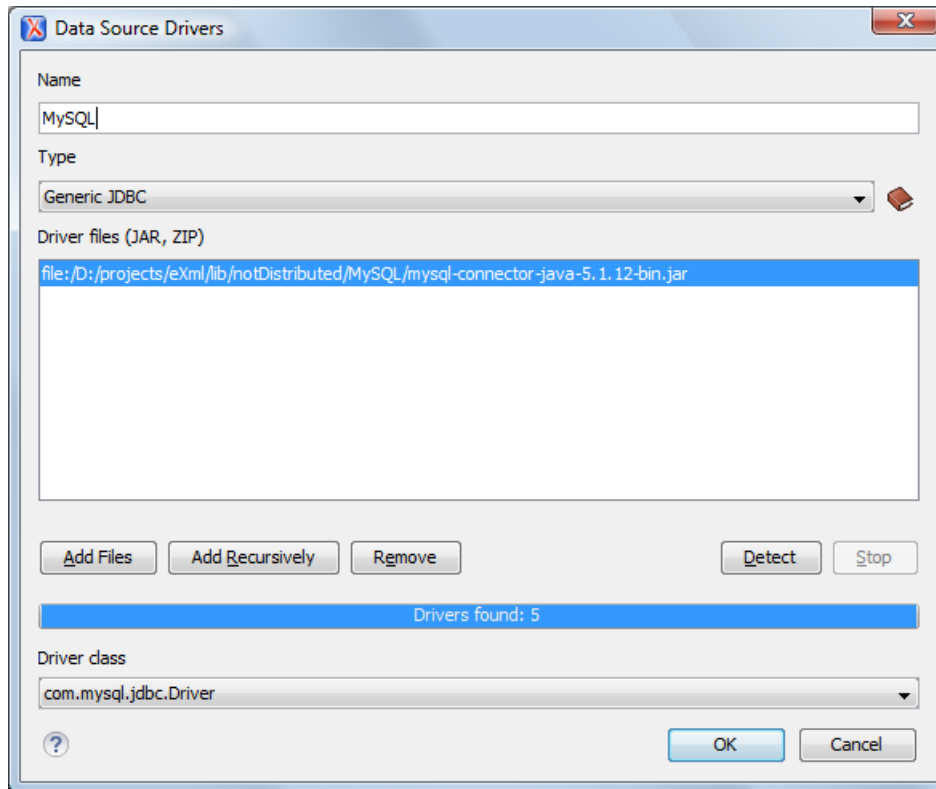


Figure 353: Data Source Drivers Configuration Dialog Box

3. Enter a unique name for the data source.
4. Select *Generic JDBC* in the driver **Type** drop-down list.
5. Add the MySQL driver files using the **Add Files** button.

The driver file for the MySQL server is called `mysql-com.jar`. The **Driver files** section lists [download links for database drivers](#) that are necessary for accessing MySQL databases in Oxygen XML Developer.

6. Select the most appropriate **Driver class**.
7. Click the **OK** button to finish the data source configuration.
8. 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. In the **Connections** panel, click the **+ New** button.

The dialog box for configuring a database connection is displayed.

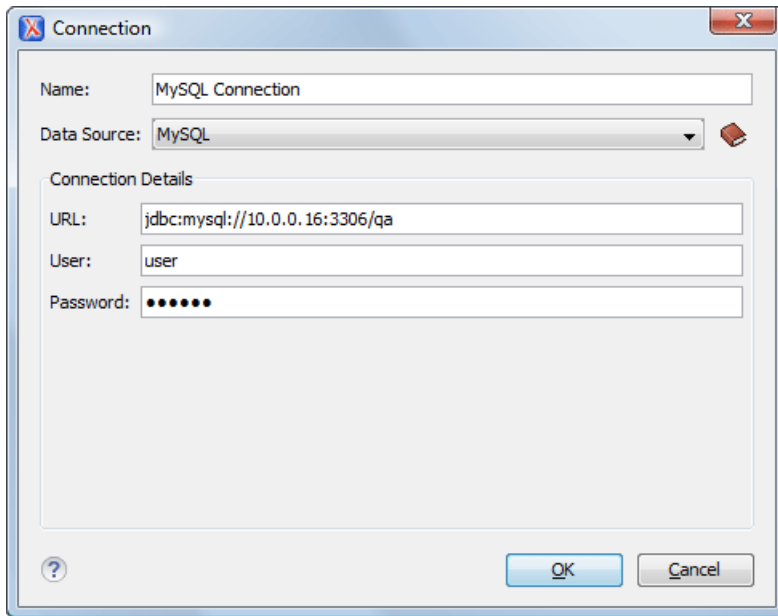


Figure 354: Connection Configuration Dialog Box

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 configuration of the database connection.

Generic JDBC Database Connections

Oxygen XML Developer 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 from the **Window > Show View** menu) or switch to the **Database Perspective** (available from **Window > Open Perspective > Database**).

How to Configure Generic JDBC Data Source Drivers

Starting with version 17, Oxygen XML Developer 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 Developer 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. In the **Connections** panel, click the **+ New** button.
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 configuration of the database connection.

JDBC-ODBC Database Connections

Oxygen XML Developer includes support for JDBC-ODBC database connections.

How to Configure a JDBC-ODBC Connection

Starting with version 17, Oxygen XML Developer 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 Developer 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. In the **Connections** panel, click the **+ New** button.

The dialog box for configuring a database connection is displayed.

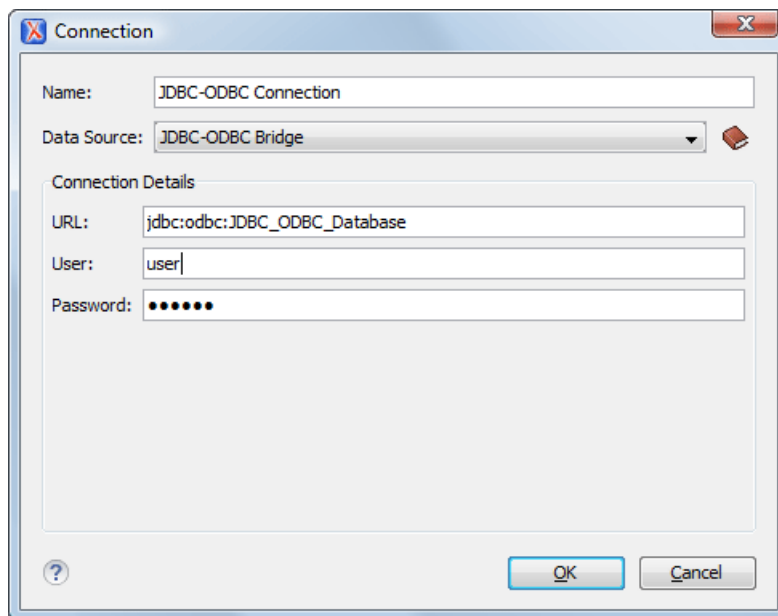


Figure 355: Connection Configuration Dialog Box

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 configuration of the database connection.
7. To view your connection, go to the **Data Source Explorer** view (if the view is not displayed, it can be opened from the **Window > Show View** menu) or switch to the **Database Perspective** (available from **Window > Open Perspective > Database**).

BaseX Database Connections

Oxygen XML Developer includes support for BaseX database connections using a WebDAV connection. BaseX is a light-weight XML database engine and XQuery processor. Oxygen XML Developer 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 Developer as described in [How to Configure a WebDAV Connection](#) on page 741. 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 from the **Window > Show View** menu) or switch to the **Database Perspective** (available from **Window > Open Perspective > Database**).

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	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	Allows you to find and replace text in multiple files from the connection.
 Resource Level Nodes	Open	Opens the selected resource in the editor.
	 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	Allows you to find and replace text in multiple files from the connection.
	Compare	Compares two selected resources using the Compare Files tool .

XQuery Execution

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 690. 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 as described in [How to Configure an XQJ Connection](#) on page 691.

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 Developer and create a *Transformation Scenario* as described in [XQuery Transformation](#) on page 590. 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.

WebDAV Connections

Oxygen XML Developer includes support for WebDAV server connections. Oxygen XML Developer 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 Developer 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. In the **Connections** panel, click the **New** button.
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.
7. To view your connection, go to the **Data Source Explorer** view (if the view is not displayed, it can be opened from the **Window > Show View** menu) or switch to the **Database Perspective** (available from **Window > Open Perspective > Database**).

To watch our video demonstration about the WebDAV support in Oxygen XML Developer, go to http://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	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	Allows you to find and replace text in multiple files from the connection.
 Resource Level Nodes	Open	Opens the selected resource in the editor.
	 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	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 Developer 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](#) on page 705 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.

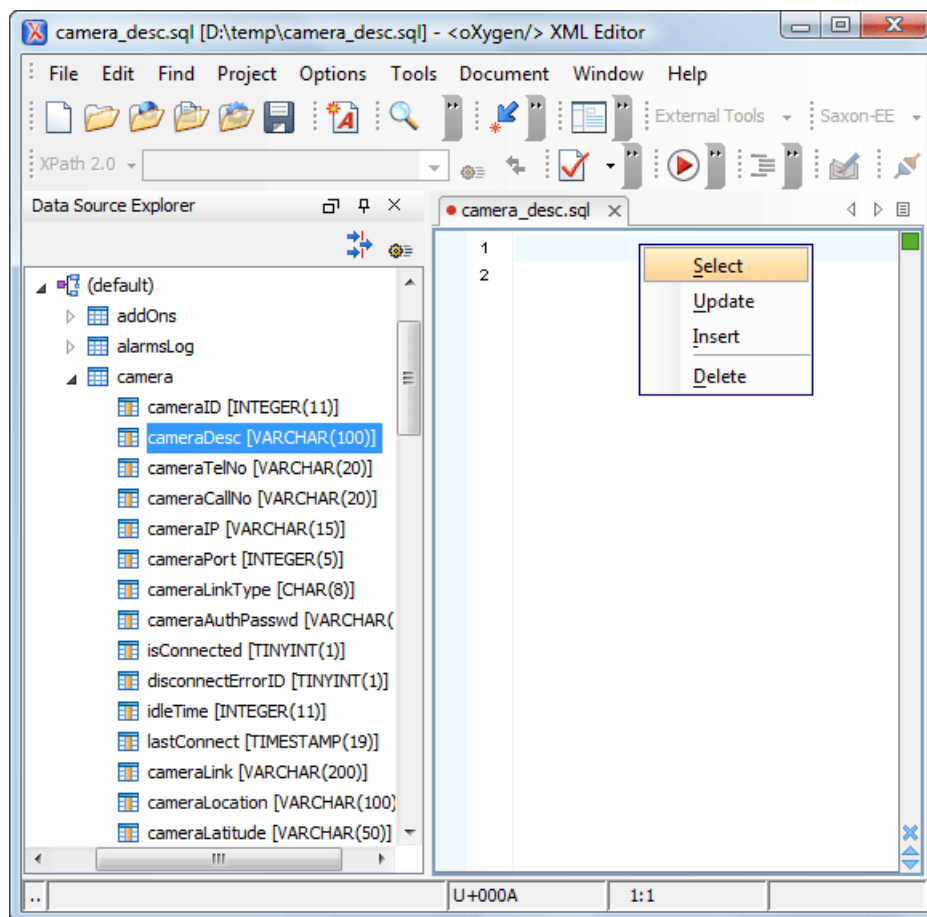


Figure 356: SQL Statement Editing with Drag and Drop

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 in-line 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 Developer 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 Developer.

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 Developer 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)

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 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 357: 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 Developer 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 clipboard.
- Paste the value wherever you need it (for example, in an opened XQuery editor panel of Oxygen XML Developer).

Related information

[XML Transformation with XQuery](#) on page 566

This type of transformation specifies the transform parameters and location of an XQuery file that is applied to the edited XML document.

XQuery Database Debugging

Oxygen XML Developer 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 Developer, see the [Debugging XSLT Stylesheets and XQuery Documents](#) on page 769 section.

Debugging with MarkLogic

Oxygen XML Developer 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 Developer 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 Developer 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 Developer 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 729.

Remote Debugging with MarkLogic

Oxygen XML Developer allows you to debug remote applications that use XQuery (for example, web applications that trigger XQuery executions). Oxygen XML Developer 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 Developer 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

<http://oxygenxml.com/demo/XQueryDebuggerforMarkLogic.html>.

Related information

[MarkLogic Development in Oxygen XML Developer](#) on page 727

[Configuring a MarkLogic Database Connection](#) on page 726

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 725

[MarkLogic Development in Oxygen XML Developer](#) on page 727

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 Developer XQuery Debugger. [The same restrictions and peculiarities](#) apply for the Berkeley debugger as for the MarkLogic debugger.

Content Management System (CMS) Integration

Describes how you can integrate Oxygen XML Developer with content management systems.

This chapter explains how you can use Oxygen XML Developer 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 Developer.

Related concepts

[General Configuration of an Oxygen XML Developer Plugin](#) on page 820

Related information

[Working with Databases](#) on page 702

Details about working with various database types in Oxygen XML Developer.

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 Developer 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 Developer 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.

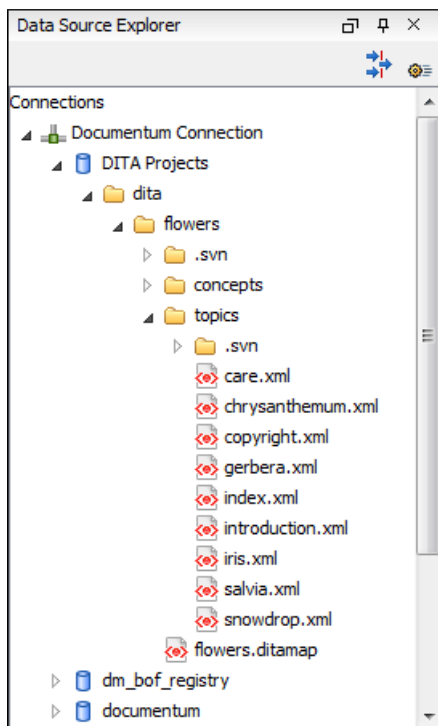


Figure 358: Documentum (CMS) Connection

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 from the **Window > Show View** menu) or switch to the **Database Perspective** (available from **Window > Open Perspective > Database**).

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/utis/*.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 Developer 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

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**

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, enable 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.

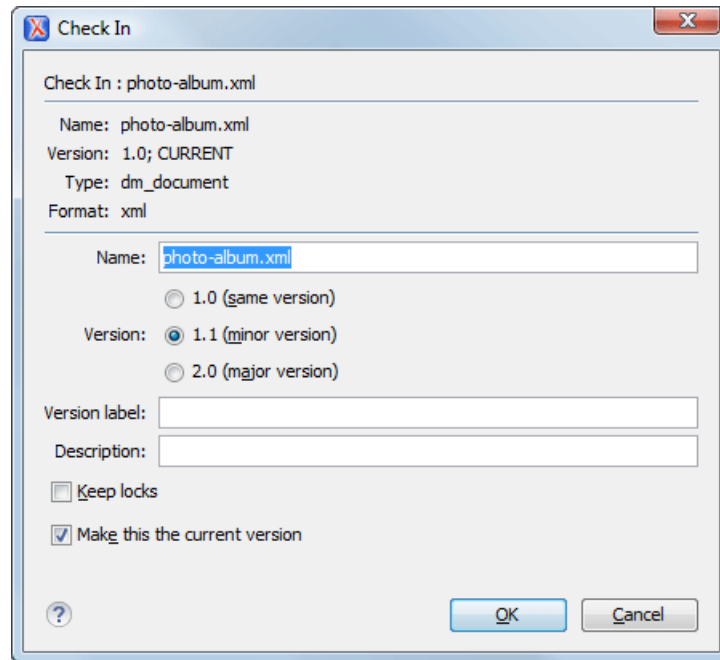


Figure 359: Check In Dialog Box

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 enabled, the updated resource is checked into 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	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 Developer 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 Developer.

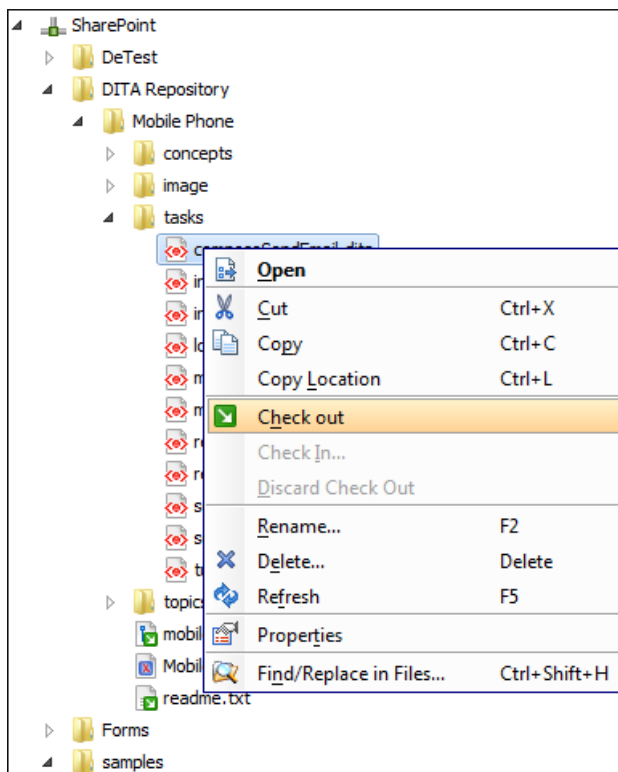


Figure 360: SharePoint Connection

Related information

[Working with Databases](#) on page 702

Details about working with various database types in Oxygen XML Developer.

How to Configure a SharePoint Connection

By default, Oxygen XML Developer 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.

To watch our video demonstration about connecting to repository located on a SharePoint server, go to http://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**.

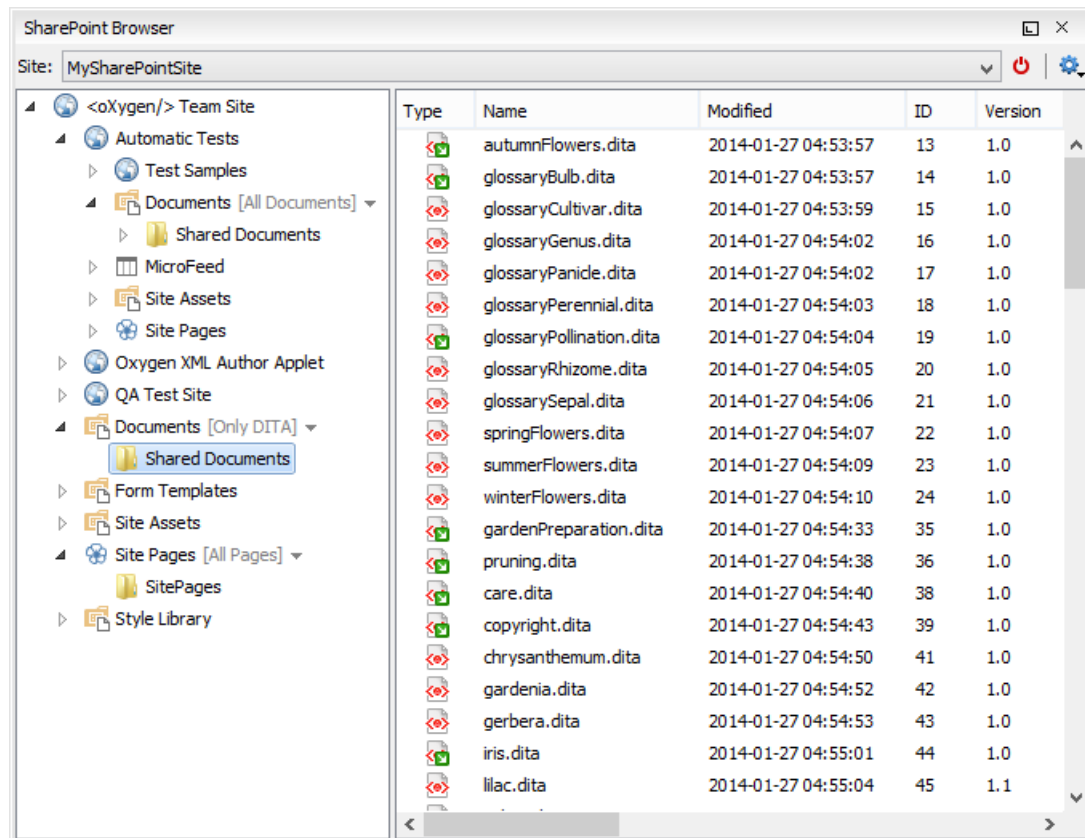


Figure 361: SharePoint Browser View

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.

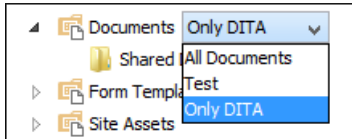


Figure 362: Drop-Down Menu to Select Which Items to Display

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.

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.		

Action	Description	Available for	
		folders	documents
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.

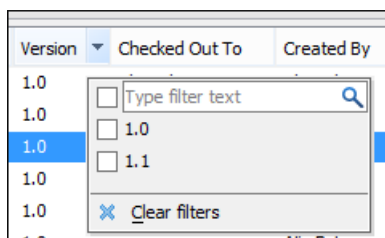


Figure 363: Column Filter

Related tasks

[How to Configure a SharePoint Connection](#) on page 755

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.
 Folder Level Nodes	 Find/Replace in Files	Allows you to find and replace text in multiple files from the connection.
	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	Allows you to find and replace text in multiple files from the connection.
 Resource Level Nodes	Open	Opens the selected resource in the editor.
	 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	Allows you to find and replace text in multiple files from the connection.
	Compare	Compares two selected resources using the Compare Files tool .

Chapter 12

Importing Data

Topics:

- [*Import from Text Files*](#)
- [*Import from MS Excel Files*](#)
- [*Import Database Data as an XML Document*](#)
- [*Import from HTML Files*](#)
- [*Import Content Dynamically*](#)

This chapter explains how to import data from various sources into XML documents.

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 Developer 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.

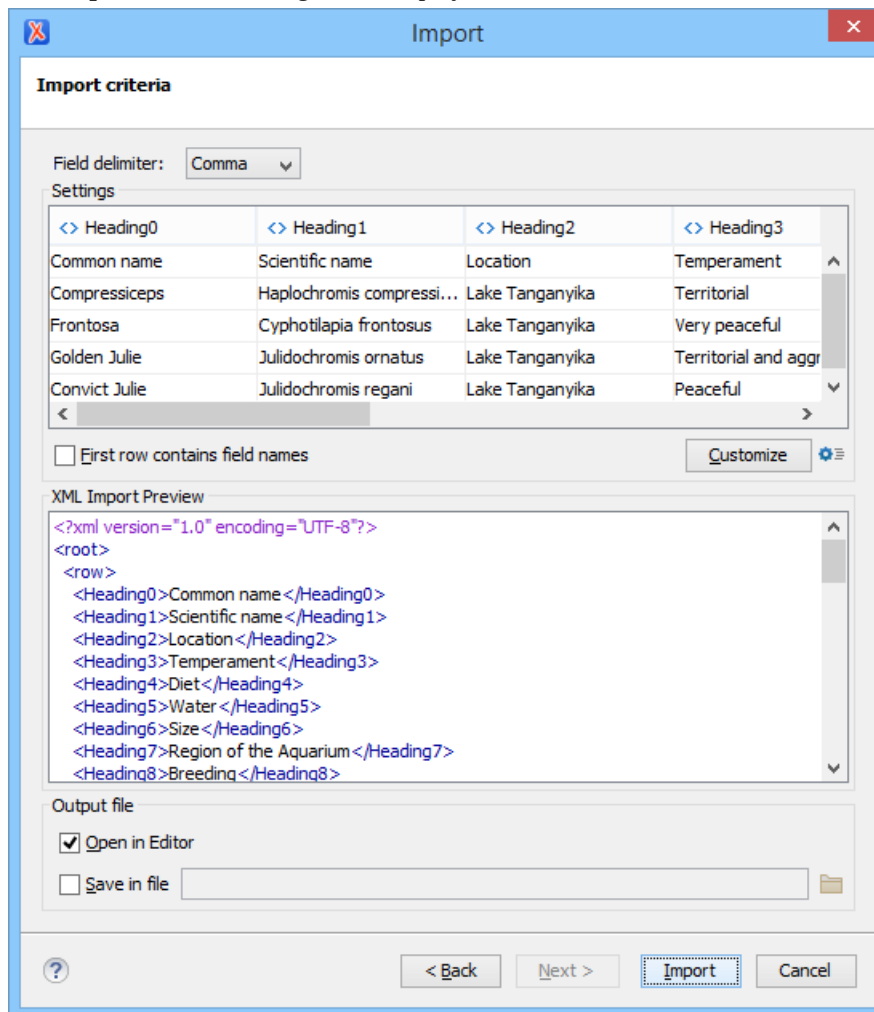


Figure 364: Import Criteria Dialog Box

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 enabled, 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](#) on page 120 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 checked, the new XML document created from the imported text file is opened in the editor.
 - h) **Save in file** - If checked, the new XML document is saved in the specified path.
6. Click **Import** to generate the XML document.

Import from MS Excel Files

By default, importing Excel 97/2000/XP/2003 formats are supported 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 763 for instructions on adding more libraries.

Oxygen XML Developer offers support for importing MS Excel Files.

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.
Opens the **Import criteria** dialog box.

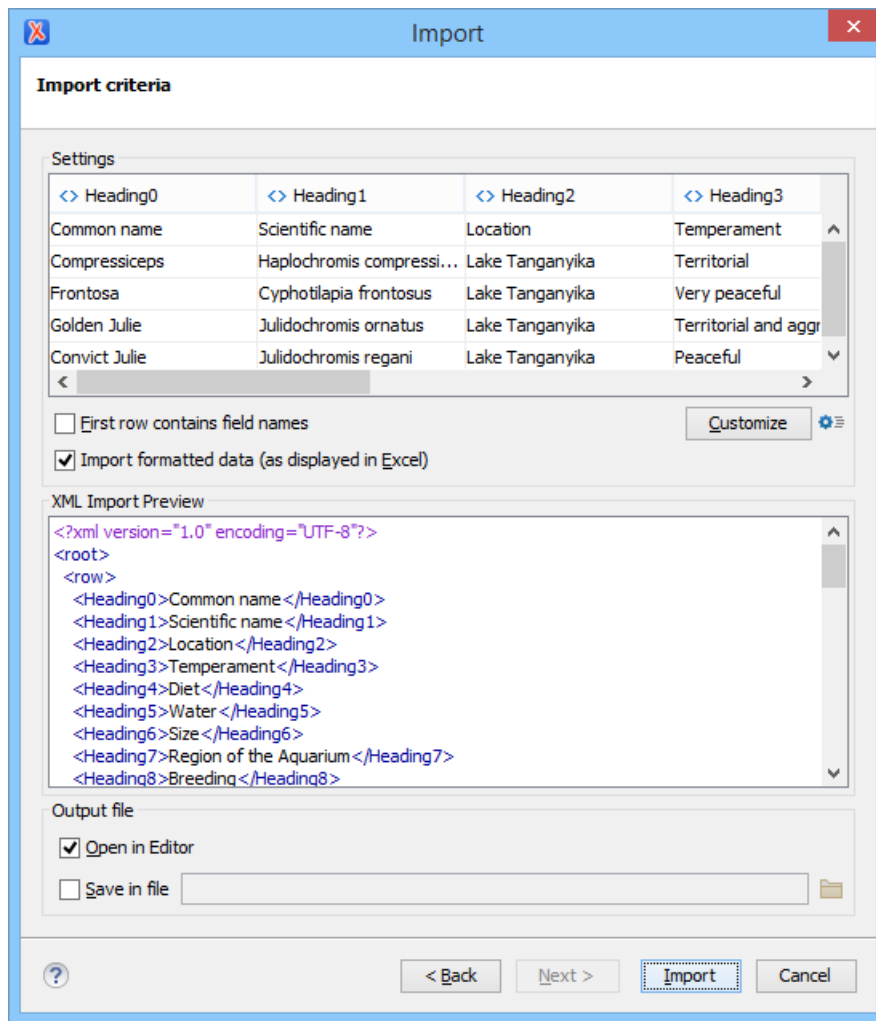


Figure 365: Import Criteria Dialog Box

4. Configure the settings for the conversion.
 - a) 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 (x symbol) when generating the XML file. You can cycle through these three options by continuing to click the column header.
 - b) **First row contains field names** - If this option is enabled, 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.
 - c) **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.
 - d) **Import Settings** - Clicking this button opens the *Import Preferences* on page 120 page that allows you to configure more import options.
 - e) **Import formatted data (as displayed in Excel)** - If this option is selected, the imported data retains the Excel styling. If deselected, the data formatting is not imported.
 - f) The **XML Import Preview** panel contains an example of what the generated XML document looks like.
 - g) **Open in editor** - If checked, the new XML document created from the imported file is opened in the editor.
 - h) **Save in file** - If checked, 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 Developer 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 Developer :
 - 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

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:

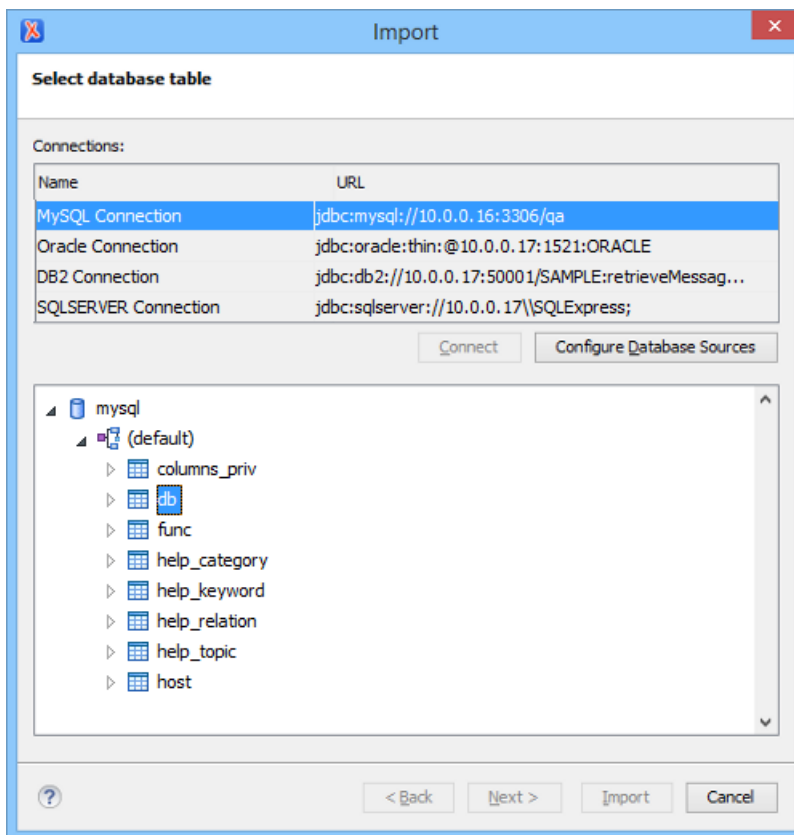


Figure 366: Select Database Table Dialog Box

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.

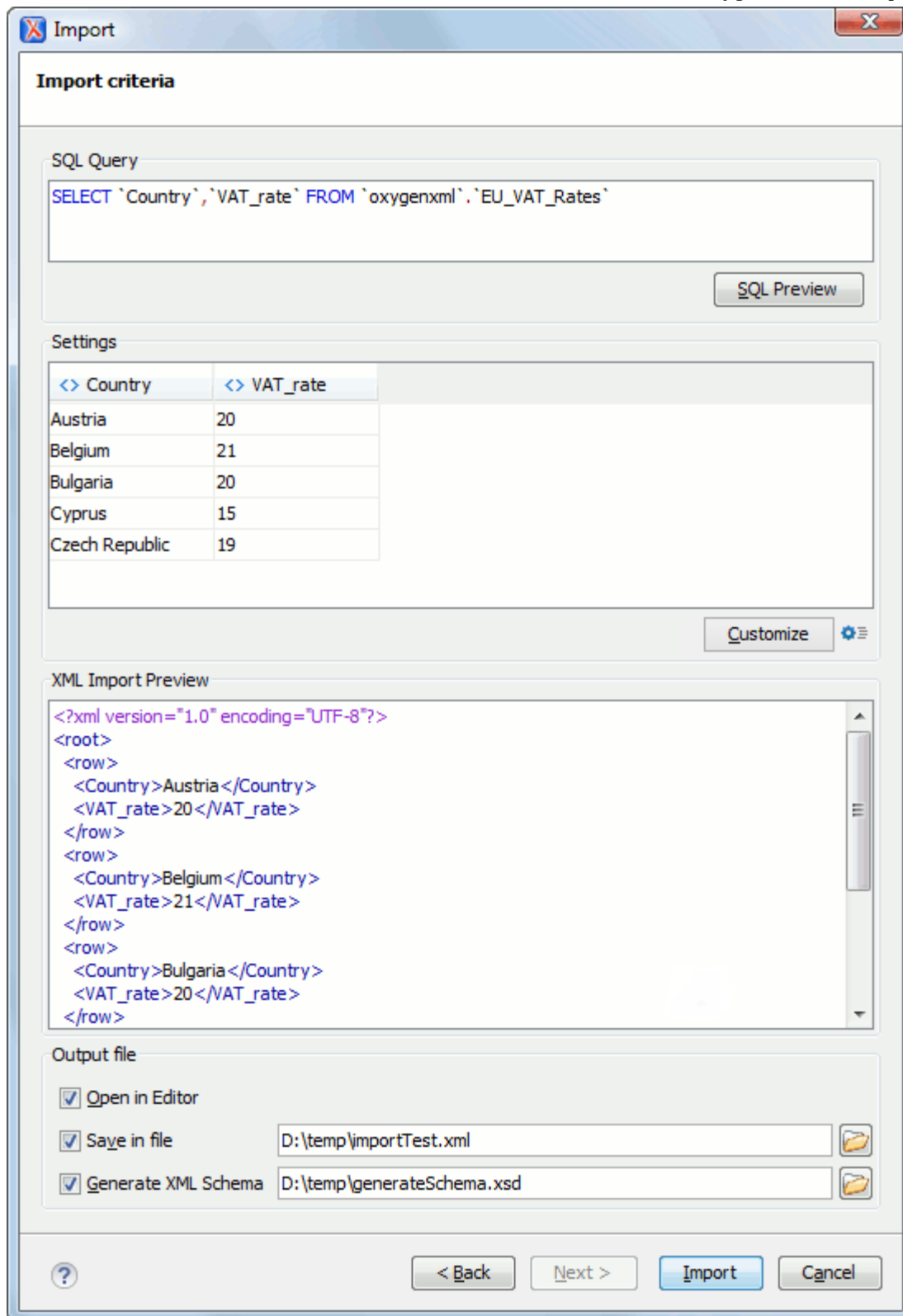


Figure 367: Import from Database Criteria Dialog Box

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* on page 120 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 checked, the new XML document created from the imported file is opened in the editor.
- f) **Save in file** - If checked, 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 Developer offers support for importing HTML files as an XML document.

To import from HTML files, follow these steps:

1. Go to **File > Import > HTML File**.
The **Import HTML** dialog box is displayed.
2. Enter the URL of the HTML document.
3. Select the type of the resulting XHTML document:
 - XHTML 1.0 Transitional
 - XHTML 1.0 Strict
4. Click the **OK** button.

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 XHTML Transitional or XHTML Strict depending on the option you chose when performing the import operation.

Import Content Dynamically

Along with the built-in support for various useful URL protocols (such as HTTP or FTP), Oxygen XML Developer 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 Developer. 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 XSLT 2.0 processing. 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 XQuery processing. 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 Developer 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 **Libraries** button in the **Advanced** tab of the transformation scenario.

Chapter 13

Debugging XSLT Stylesheets and XQuery Documents

Topics:

- [Layout](#)
- [Working with the XSLT / XQuery Debugger](#)
- [Debugging Java Extensions](#)
- [Supported Processors for XSLT / XQuery Debugging](#)
- [Performance Profiling of XSLT Stylesheets and XQuery Documents](#)

Describes the debugging interface that helps you to detect and solve problems with XSLT and XQuery transformations.

Oxygen XML Developer includes a 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 enable 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 9.6.0.7 XSLT engine) and XQuery 1.0 / 3.0 (using Saxon 9.6.0.7 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.

Layout

The XML and XSL files are displayed in *Text mode*. The *Grid mode* is available only in *the Editor perspective*.

The debugger perspective contains four sections:

- **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 document 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. There are two types of output views: a text view (with XML syntax highlight) and an XHTML view. For large outputs, the XHTML view can be disabled (see *Debugger Settings*).
- **Control view** - The control view is used to configure and control the debugging operations. It also provides a set of *Information views* types. This pane has two sections:
 - *Control toolbar*
 - *Information views*

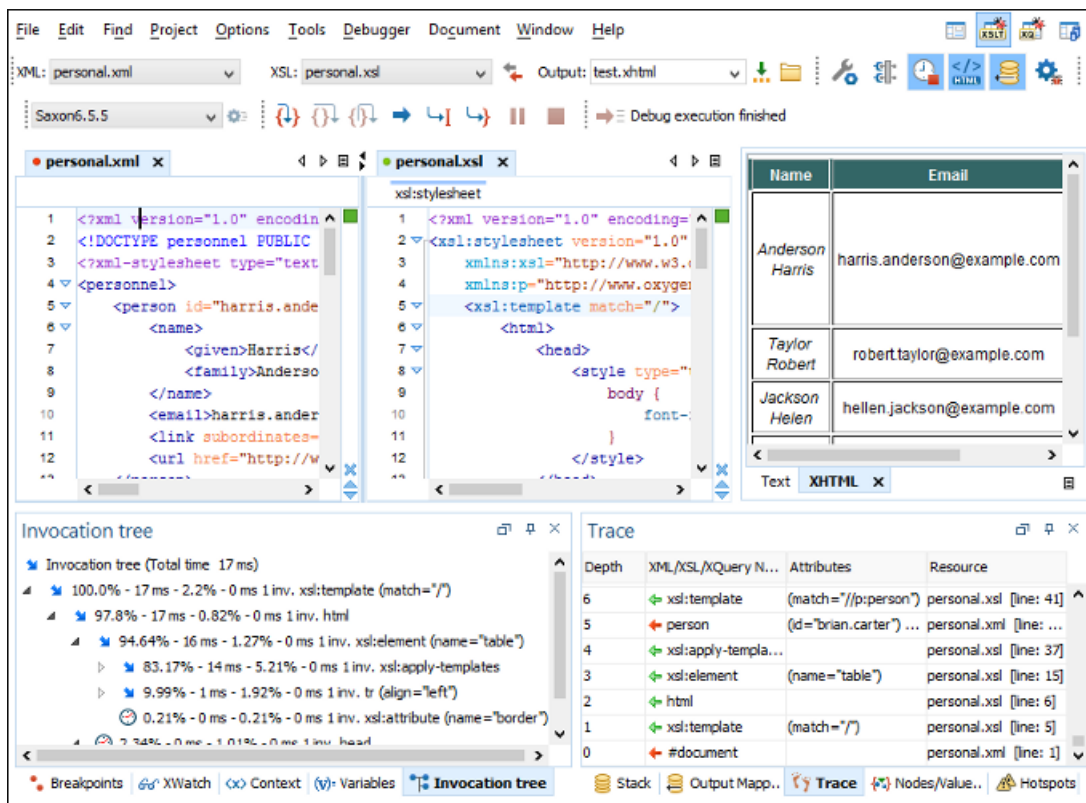


Figure 368: Debugger Mode Interface

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.

When editing in the Editor perspective the editor toolbar is displayed. In Debugger mode this toolbar is not available, however the functions are still accessible from *the Document menu* and *the editors 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.

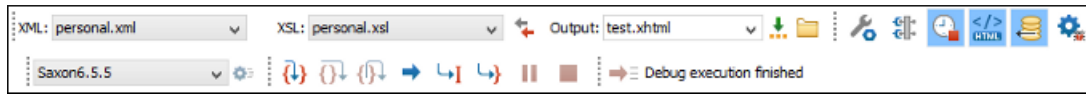


Figure 369: Control Toolbar

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 enabled, the XML and XSLT/XQuery selectors display the names of the files opened in the central editor panels. This button is disabled 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 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 9.6.0.7.
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.



Figure 370: 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.

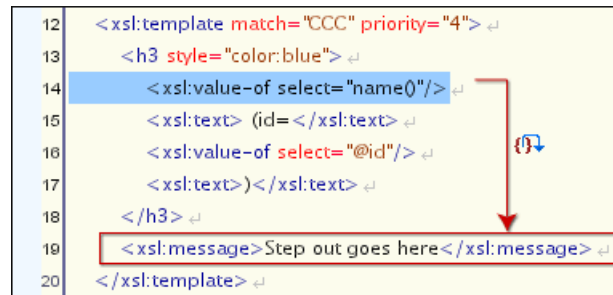


Figure 371: Step out

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 568

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](#)

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 from the **Window > Show View** menu.

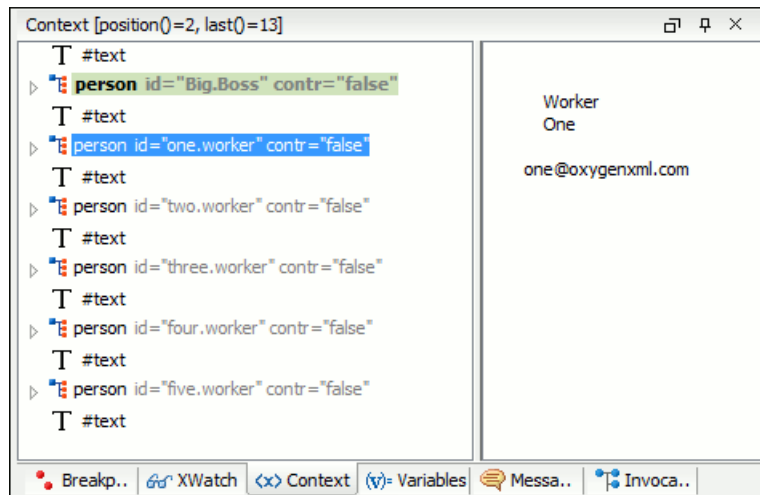


Figure 372: Context node view

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 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 Developer supports you with syntax highlight and [content completion assistance](#).

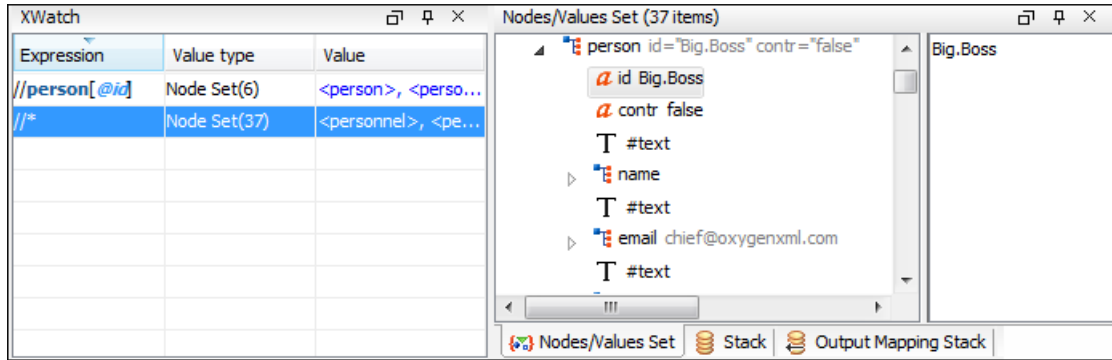


Figure 373: XPath Watch View

Table 7: 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 on page 781). For <i>Node Set</i> 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 from the **Window > Show View** menu.

Once you [insert a breakpoint](#) it is automatically added in this list. Breakpoints can be set in XSLT/XQuery documents and XML documents in XSLT/XQuery debugging sessions. 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**.

Enabled	Resource	Condition
☒	(conditional only)	count(preceding::person)=2
☒	personal.xml [line:16]	local-name()='person'
☒	personal.xml [line:22]	position()>=1 Last evaluated as: false
☒	personal.xml [line:34]	(no condition)
☒	personal.xml [line:44]	(no condition)
☒	personal.xml [line:50]	(no condition)

Figure 374: Breakpoints View

The **Breakpoints** view contains the following columns:

- **Enabled** - If checked, 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.



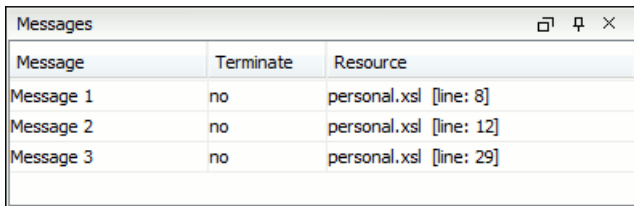
Note: The breakpoints list is not deleted at the end of a transformation (it is preserved between debugging sessions).

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.

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 from the **Window > Show View** menu.



Message	Terminate	Resource
Message 1	no	personal.xml [line: 8]
Message 2	no	personal.xml [line: 12]
Message 3	no	personal.xml [line: 29]

Figure 375: Messages View**Table 8: 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 <code>xsl:message</code> 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).



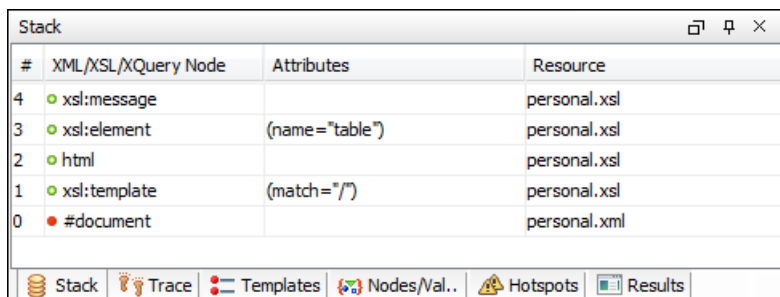
Important: Remarks

- 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 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 Developer 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.



#	XML/XSL/XQuery Node	Attributes	Resource
4	xsl:message		personal.xml
3	xsl:element	(name="table")	personal.xml
2	html		personal.xml
1	xsl:template	(match="/")	personal.xml
0	#document		personal.xml

Figure 376: 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 9: 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 <code>id="value"</code> 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 from the **Window > Show View** menu.

#	XML/XSL/XQuery Node	Attributes	Resource
8	xsl:value-of	(select="./link/@manager")	personal.xml
7	font	(color="black") (name="verdana") (size="3")	personal.xml
6	xsl:element	(name="td")	personal.xml
5	xsl:element	(name="tr")	personal.xml
4	xsl:template	(match="//person")	personal.xml
3	xsl:apply-templates		personal.xml
2	xsl:element	(name="table")	personal.xml
1	html		personal.xml
0	xsl:template	(match="/")	personal.xml

Figure 377: 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 10: 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.

Column	Description
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 tasks

[Determining What XSLT / XQuery Expression Generated Particular Output](#) on page 784

Related information

[Stack View](#) on page 777

[Trace History View](#) on page 779

[Templates View](#) on page 780

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 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.

Trace			
Depth	XML/XSL/XQuery Node	Attributes	Resource
0	→ #document		personal.xml
1	⇒ xsl:template	(match="/")	personal.xsl
2	⇒ html		personal.xsl
3	⇒ xsl:element	(name="table")	personal.xsl
4	⇒ xsl:message		personal.xsl
4	⇐ xsl:message		personal.xsl
4	⇒ xsl:message		personal.xsl

Figure 378: 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 11: 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 <code>id="value"</code> 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 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.

Match	Hits	Priority	Mode	Name	Resource
//person	6				personal.xml
/	1				personal.xml

Figure 379: 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 12: Templates columns

Column	Description
Match	The <code>match</code> attribute of the <code>xsl:template</code> .
Hits	The number of hits for the <code>xsl:template</code> . Shows how many times the XSLT processor used this particular template.
Priority	The template priority as established by XSLT processor.
Mode	The <code>mode</code> attribute of the <code>xsl:template</code> .

Column	Description
Name	The <code>name</code> attribute of the <code>xsl:template</code> .
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 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.

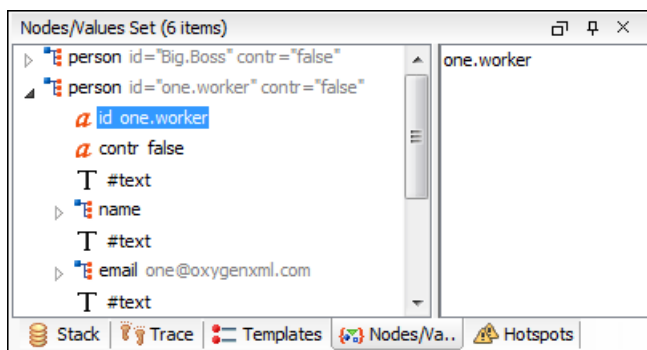


Figure 380: Node Set view

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 from the **Window > Show View** menu.

Variables and parameters play an important role during an XSLT/XQuery transformation. Oxygen XML Developer uses the following icons to differentiate variables and parameters:

-  - 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.

	Name	Value type	Value
{V}	val	String	
{P}	rowval	String	1,2
V	trans	String	
P	level	String	1,2
P	image-path	String	Images/

Figure 381: Variables View

Table 13: 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

This section describes how to work with the debugger in the most common use cases.

To watch our video demonstration about how you can use the **XSLT Debugger**, go to http://oxygenxml.com/demo/XSLT_Debugger.html.

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 tasks

[Determining What XSLT / XQuery Expression Generated Particular Output](#) on page 784

Using Breakpoints

The Oxygen XML Developer 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.

Inserting Breakpoints

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.

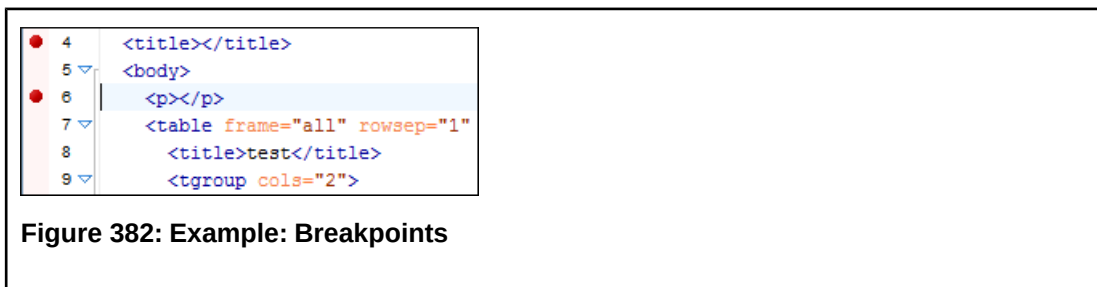


Figure 382: Example: Breakpoints

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 select **Remove all** from the **Edit > Breakpoints** menu.

Determining What XSLT / XQuery Expression 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*.

The screenshot illustrates the 'XHTML Output to Source Mapping' feature in Oxygen XML Developer. It shows three main components:

- personal.xml (Source XML):** Contains three person entries. The second entry, `<person id="two.worker">`, is highlighted in blue.
- personal.xsl (XSLT Stylesheet):** Contains XSLT code. The `<xsl:value-of select="email/text">` and `<xsl:value-of select="link/@manager">` nodes are highlighted in blue.
- XHTML Output:** A table with three columns: Name, Email, and Link. The second row, corresponding to 'two.worker', is highlighted in blue.
- Output Mapping Stack:** A table showing the mapping of XSL/XQuery nodes to the output. The stack is as follows:

#	XSL/XQuery Node	Attributes	Resource
8	xsl:value-of	(select="link/@manager")	personal.xsl
7	font	(color="black") (name="verdana")...	personal.xsl
6	xsl:element	(name="td")	personal.xsl
5	xsl:element	(name="tr")	personal.xsl
4	xsl:template	(match="//person")	personal.xsl
3	xsl:apply-templates		personal.xsl
2	xsl:element	(name="table")	personal.xsl
1	html		personal.xsl
0	xsl:template	(match="/")	personal.xsl

Arrows indicate the mapping from the XHTML output back to the source XML and XSLT code. A red arrow points from the 'two.worker' entry in the XHTML output to the corresponding entry in 'personal.xml'. A green arrow points from the 'two.worker' entry in the XHTML output to the `<xsl:value-of select="link/@manager">` node in 'personal.xsl'.

Figure 383: XHTML Output to Source Mapping

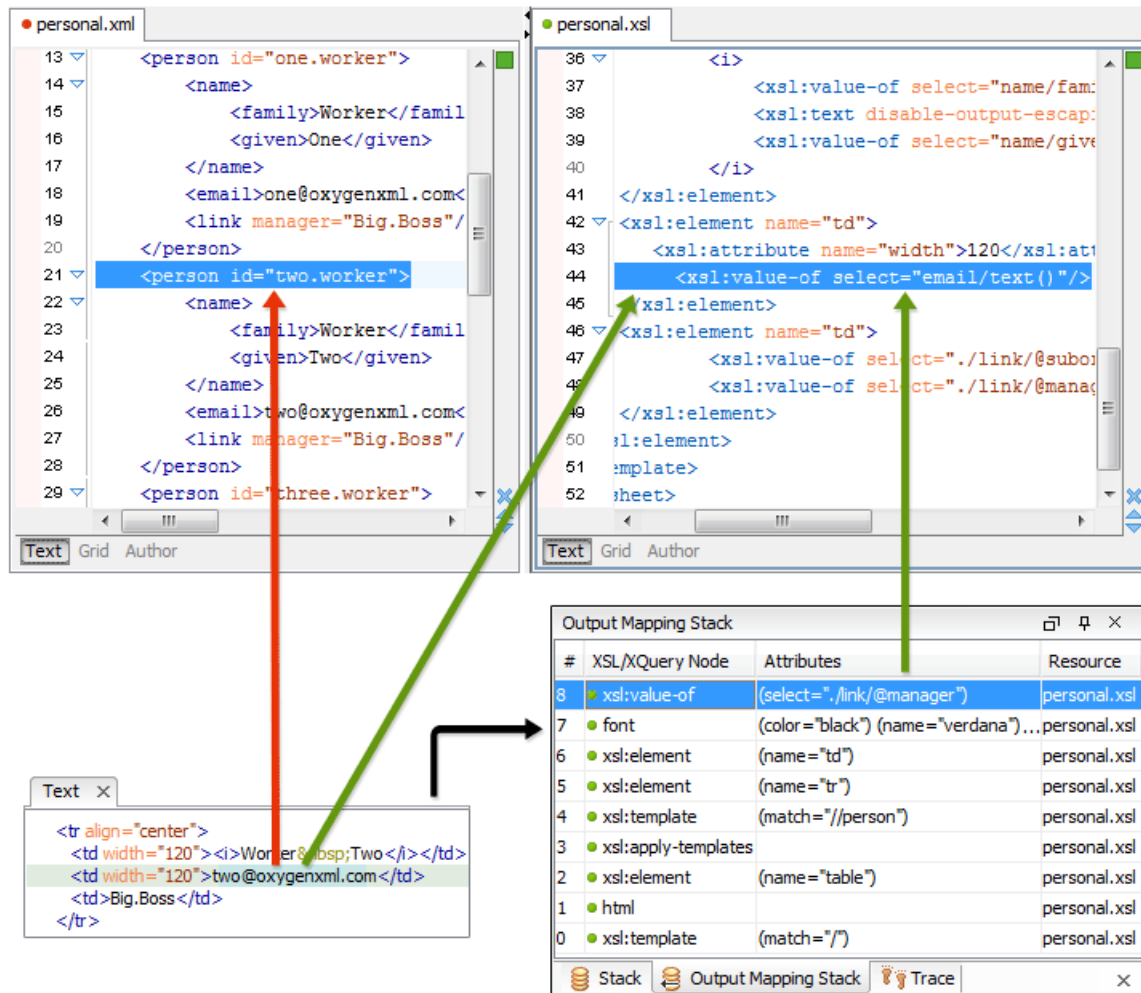


Figure 384: Text Output to Source Mapping

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 778

[Trace History View](#) on page 779

[Templates View](#) on page 780

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.xml.Oxygen` as the main Java class of the configuration.

The main Java class `ro.sync.xml.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 9.6.0.7 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 9.6.0.7 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 9.6.0.7 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.

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.

Viewing Profiling Information

This section explains the views that display the profiling data collected by the profiles during the transformation.

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 from the **Window > Show View** menu.

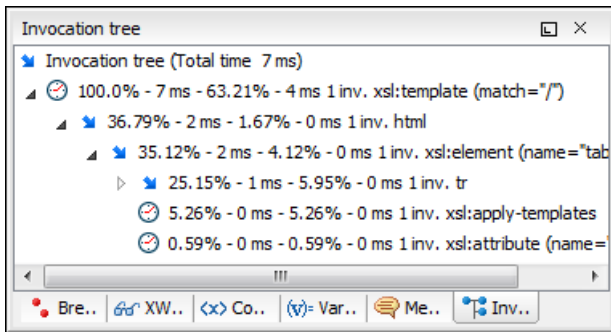


Figure 385: Invocation Tree View

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 from the **Window > Show View** menu.

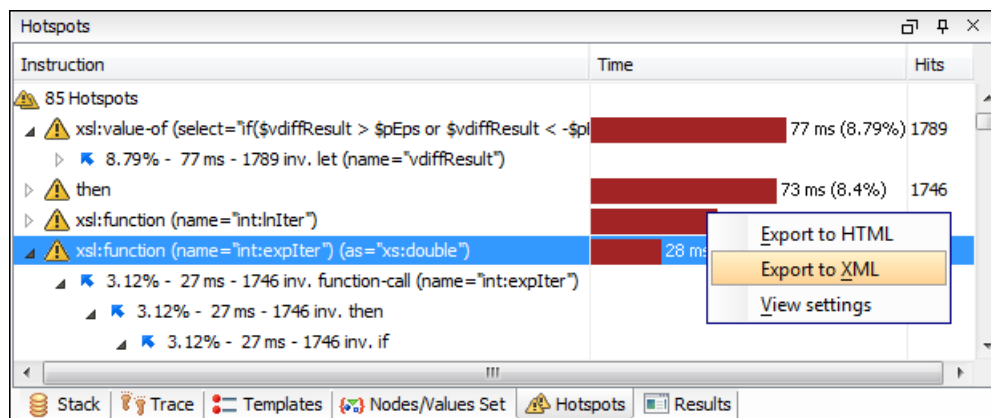


Figure 386: Hotspots View

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.

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 783).

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.

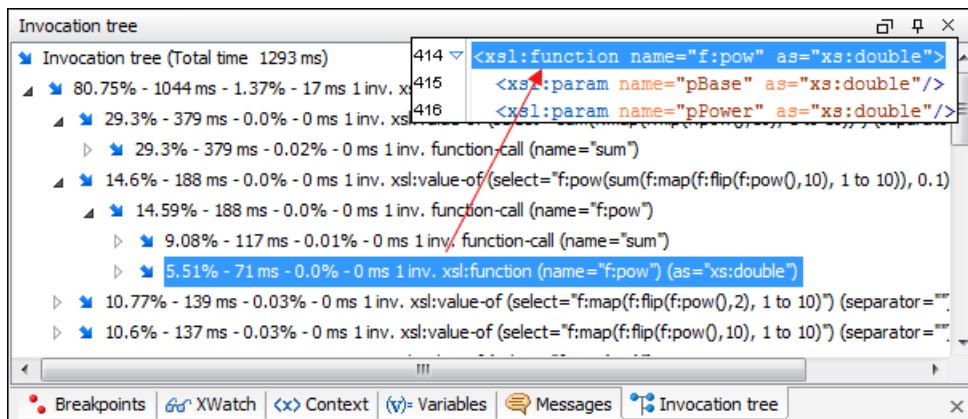


Figure 387: Source backmapping

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 Developer 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 Developer 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 Developer 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 http://oxygenxml.com/demo/XSLT_Profiling.html.

Chapter 14

Document Types (Frameworks)

Topics:

- [Predefined Document Types \(Frameworks\)](#)

Presents the built-in support for various document types.

A *document type* or *framework* is associated to an XML file 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 **Text** mode.
- Predefined scenarios used for transformations for the class of XML documents defined by the document type.
- XML catalogs.
- Directories with file templates.

Oxygen XML Developer 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 Developer, go to <http://oxygenxml.com/demo/FrameworkConfiguration.html>.

Predefined Document Types (Frameworks)

Predefined Document Types

The following predefined document types (frameworks) are fully supported in Oxygen XML Developer and each of these document types include built-in transformation scenarios, validation, content completion and file templates:

- [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 P4](#) - 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.
- [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.

Other Document Types

Oxygen XML Developer also provides limited support (including file templates) for a variety of other document types, including:

- [EPUB \(NCX, OCF, OPF 2.0 & 3.0\)](#) - A standard for e-book files.
- [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 *QuickFixes* 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.
- [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.
- [JavaScript](#) - Programming language of HTML and the Web.
- [XML Spec](#) - 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 - A document that describes a mapping between external entity references and locally-cached equivalents.

DocBook 4 Document Type

DocBook is a very popular set of tags for describing books, articles, and other prose documents, particularly technical documentation.

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`.

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/`.

The default XML catalog, `catalog.xml`, is stored in `[OXYGEN_INSTALL_DIR]/frameworks/docbook/`.

DocBook 4 Transformation Scenarios

Default transformation scenarios allow you to convert DocBook 4 to DocBook 5 documents and transform DocBook documents to WebHelp, PDF, HTML, HTML Chunk, XHTML, XHTML Chunk, EPUB and EPUB 3.

Related information

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

[Editing a Transformation Scenario](#) on page 596

DocBook4 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 the WebHelp Classic with Feedback System](#) on page 649.

To watch our video demonstration about the feedback-enabled WebHelp system, go to http://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`.

clean.output

Deletes all files from the output folder before the transformation is performed (or not, if the value is `no`). The default value is `no`.

l10n.gentext.default.language

This parameter is used to identify the correct stemmer that differs from language to language. 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 search (the default value 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 are included in 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. If you want to integrate social media features (such as widgets for Facebook™, Twitter™, Google+™), the XML must be well-formed, each widget must be in separate `div` or `span` element, and each widget must be included in an XML comment (also, the start and end tags for the XML comment must be included). The following code excerpt 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)) {
      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-label="share">
  </div>
```

webhelp.footer.include

Specifies whether or not to include footer in each WebHelp page. Possible values are `yes` and `no`. If set to `yes` and the `webhelp.footer.file` parameter has a value, the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default footer is inserted in each WebHelp page.

<code>webhelp.logo.image.target.url</code>	Specifies a target URL that is set on the logo image. When you click the logo image, the browser goes to the address.
<code>webhelp.logo.image</code>	Specifies a path to an image displayed as a logo in the left side of the output head.
<code>webhelp.product.id</code> (available only for Feedback-enabled systems)	This parameter specifies a short name for the documentation target, or product (for example, <code>mobile-phone-user-guide</code> , <code>hvac-installation-guide</code>).  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: <code>} = ; * % + &</code>.
<code>webhelp.product.version</code> (available only for Feedback-enabled systems)	Specifies the documentation version number (for example, 1.0, 2.5, etc.). New users can view only the latest 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: <code>} = ; * % + &</code>.
<code>webhelp.search.japanese.dictionary</code>	The file path of the dictionary that will be used by the <i>Kuromoji</i> morphological engine. This engine is used for indexing Japanese content in the WebHelp pages. This indexer does not work in the Oxygen XML Developer or the Oxygen XML WebHelp plugin. To use it, you need to download the http://mvnrepository.com/artifact/org.apache.lucene/lucene-analyzers-kuromoji/4.1.0 and place it in the <code>DITA_OT_DIR/plugins/com.oxygenxml.webhelp/lib</code> directory.
<code>webhelp.search.ranking</code>	If this parameter is set to <code>false</code> then the 5-star rating mechanism is no longer displayed on the Search tab (default setting is <code>true</code>).
<code>webhelp.skin.css</code>	Path to a CSS file that sets the style theme in the output WebHelp pages. It can be the <code>OXYGEN_INSTALL_DIR/frameworks/docbook/xsl/com.oxygenxml.webhelp/skin.css</code> file, or it can be a custom skin CSS generated with the Oxygen Skin Builder .

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 System Output](#) on page 611

DocBook to PDF Output Customization

Main steps for customization of PDF output generated from DocBook XML documents.

When the default layout and output look of the DocBook to PDF transformation need to be customized, the following main steps should be followed. In this example a company logo image is added to the front matter of a book. Other types of customizations should follow some similar 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 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 Developer and set the customization layer in *the XSL URL property of the scenario*.

Related information

<http://www.sagehill.net/docbookxsl/PrintCustomEx.html>

The book **DocBook XSL: The Complete Guide** by *Bob Stayton* contains more details about customizing the PDF output.

http://www.oxygenxml.com/demo/DocBook_Customization.html

Video demonstration for creating a DocBook customization layer in Oxygen XML Developer.

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. In your DocBook to EPUB transformation, 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.

DocBook to DITA Transformation

Oxygen XML Developer 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.

DocBook 4 Templates

Default templates are available in the *New File* wizard. You can use them to create a skeletal form of a DocBook 4 book or article. These templates are stored in the `[OXYGEN_INSTALL_DIR]/frameworks/docbook/templates/DocBook 4` folder.

Here are some of the DocBook 4 templates available when creating *new documents from templates*:

- **Article**
- **Article with MathML**
- **Article with SVG**
- **Article with XInclude**
- **Book**
- **Book with XInclude**
- **Chapter**
- **Section**
- **Set of Books**

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 framework that pulls in the target data for each document. The database document is static and all the document data is pulled in dynamically. 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 agttargets 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">
            &agttargets;
          </document>
        </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.

An example of a `target.db` file:

```
<div element="book" href="#MailAdminGuide" number="1" targetptr="user_accounts">

  <ttitle>Administering User Accounts</ttitle>
  <xref text="How to administer user accounts" xref="user_accounts">
  <div element="part" href="#d5e4" number="1">
    <ttitle>First Part</ttitle>
    <xref text="Part I, 'First Part'" xref="user_accounts">
    <div element="chapter" href="#d5e6" number="1">
      <ttitle>Chapter Title</ttitle>
      <xref text="Chapter 1, Chapter Title" xref="user_accounts">
      <div element="sect1" href="#src_chapter" number="1"
targetptr="src_chapter">
        <ttitle>Section1 Title</ttitle>
```

```

<xref>xreflabel_here</xref>
</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.

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.

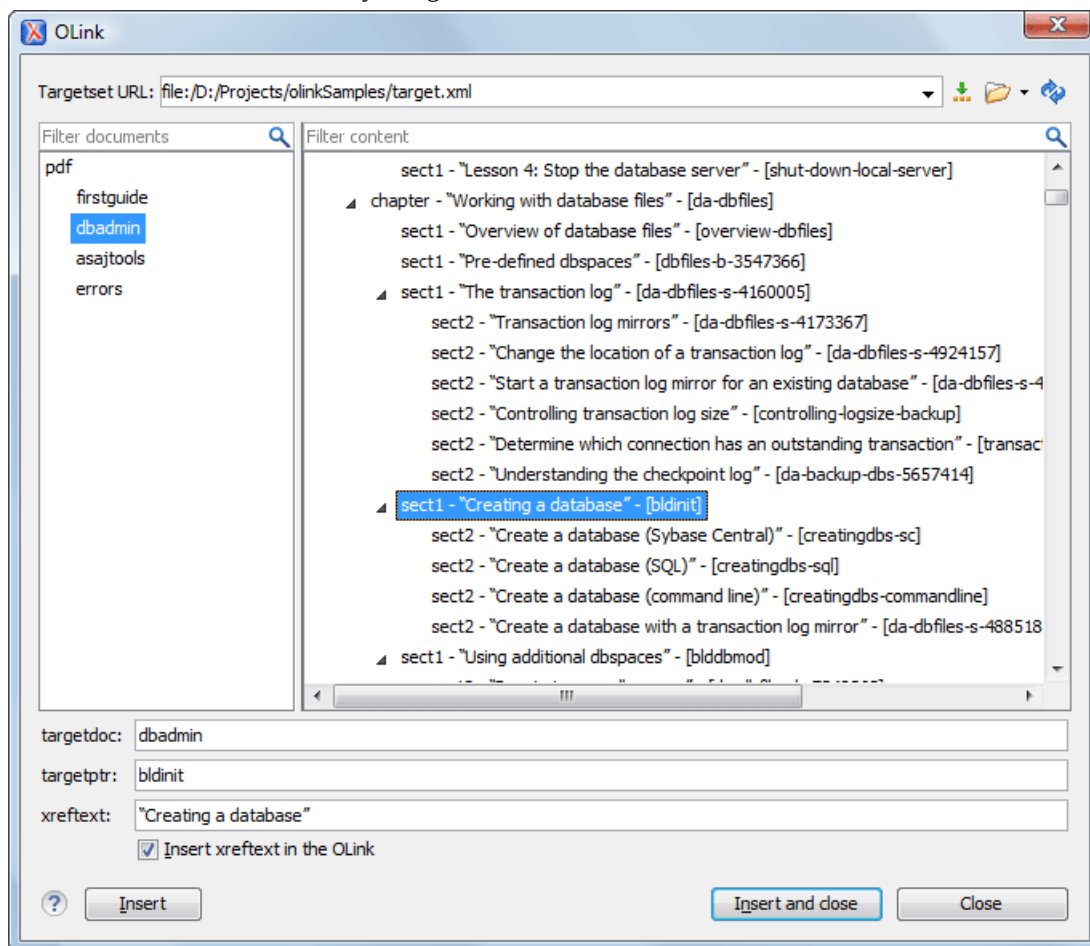


Figure 388: Insert OLink Dialog Box

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

A file is considered to be a DocBook 5 document when the namespace is *http://docbook.org/ns/docbook*.

The default schema that is used if one is not detected in the DocBook 5 file is *docbookx1.rng* and it is stored in *[OXYGEN_INSTALL_DIR]/frameworks/docbook/5.0/rng/*. Other types of schemas are also located in various folders inside the *[OXYGEN_INSTALL_DIR]/frameworks/docbook/5.0/* directory (for example, XML Schema files are located in the *xsd* folder).

The default XML catalog, *catalog.xml*, is stored in *[OXYGEN_INSTALL_DIR]/frameworks/docbook/5.0/*.



Note: A default XML catalog and schema files for experimental DocBook 5.1 documents are also available in the *[OXYGEN_INSTALL_DIR]/frameworks/docbook/5.1/* directory.

DocBook 5 Transformation Scenarios

Default transformation scenarios allow you to transform DocBook 5 documents to WebHelp, PDF, HTML, HTML Chunk, XHTML, XHTML Chunk, EPUB, and EPUB 3.

Related information

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

[Editing a Transformation Scenario](#) on page 596

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 the WebHelp Classic with Feedback System](#) on page 649.

To watch our video demonstration about the feedback-enabled WebHelp system, go to http://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`.

clean.output

Deletes all files from the output folder before the transformation is performed (or not, if the default value is `no`).

l10n.gentext.default.language

This parameter is used to identify the correct stemmer that differs from language to language. 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 search (if `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 are included in 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 page. If you want to integrate social media features (such as widgets for Facebook™, Twitter™, Google+™), the footer content must be well-formed, each widget must be in separate `div` or `span` element, and the content must be included in an XML comment (also, the start and end tags for the XML comment must be included). The following code excerpt 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)) {
      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-label="share">
  </div>
```

webhelp.footer.include

Specifies whether or not to include footer in each WebHelp page. Possible values are `yes` and `no`. If set to `yes` and the `webhelp.footer.file` parameter has a value, the content of that file is used as footer. If the `webhelp.footer.file` has no value then the default 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, the browser goes to the 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 system.



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 use



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 for indexing Japanese content in the WebHelp pages. This indexer does not come with Oxygen XML Developer or the Oxygen XML WebHelp plugin. To use it, you need to download it from <http://mvnrepository.com/artifact/org.apache.lucene/lucene-analyzers-kuromoji/4.1.0> and place it in the `DITA_OT_DIR/plugins/com.oxygenxml.webhelp/lib` directory.

webhelp.search.ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer 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 the `OXYGEN_INSTALL_DIR/frameworks/docbook/xsl/com.oxygenxml.webhelp/skin.css` file in the `lib` directory, or it can be a custom skin CSS generated with the *Oxygen Skin Builder*.

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 System Output](#) on page 611

DocBook to PDF Output Customization

Main steps for customization of PDF output generated from DocBook XML documents.

When the default layout and output look of the DocBook to PDF transformation need to be customized, the following main steps should be followed. In this example a company logo image is added to the front matter of a book. Other types of customizations should follow some similar 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 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 Developer and set the customization layer in *the XSL URL property of the scenario*.

Related information

<http://www.sagehill.net/docbookxsl/PrintCustomEx.html>

The book **DocBook XSL: The Complete Guide** by *Bob Stayton* contains more details about customizing the PDF output.

http://www.oxygenxml.com/demo/DocBook_Customization.html

Video demonstration for creating a DocBook customization layer in Oxygen XML Developer.

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. In your DocBook to EPUB transformation, 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.

DocBook to DITA Transformation

Oxygen XML Developer 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.

DocBook 5 Templates

Default templates are available in the *New File wizard* and can be used for easily creating a skeletal form of a DocBook 5 book or article. These templates are stored in the `[OXYGEN_INSTALL_DIR]/frameworks/docbook/templates/DocBook 5` folder.

Here are some of the DocBook 5 templates available when creating *new documents from templates*:

- **Article**
- **Article with MathML**
- **Article with SVG**
- **Article with XInclude**
- **Book**
- **Book with XInclude**
- **Chapter**
- **Section**
- **Set of Books**

Some experimental DocBook 5.1 templates are also available and are stored in the `[OXYGEN_INSTALL_DIR]/frameworks/docbook/templates/DocBook 5.1 (Experimental)` folder. They include the following:

- **Assembly**
- **Topic**

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 `o:links` from that collection. The target database document is an XML file that is created once. It provides a framework that pulls in the target data for each document. The database document is static and all the document data is pulled in dynamically.

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">
            &agttargets;
          </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.

An example of a `target.db` file:

```
<div element="book" href="#MailAdminGuide" number="1" targetptr="user_accounts">

  <ttitle>Administering User Accounts</ttitle>
  <xref text="How to administer user accounts" />
  <div element="part" href="#d5e4" number="I">
    <ttitle>First Part</ttitle>
    <xref text="Part I, 'First Part'" />
    <div element="chapter" href="#d5e6" number="1">
      <ttitle>Chapter Title</ttitle>
      <xref text="Chapter 1, Chapter Title" />
      <div element="sect1" href="#src_chapter" number="1"
targetptr="src_chapter">
        <ttitle>Section1 Title</ttitle>
        <xref text="xreflabel_here" />
      </div>
    </div>
  </div>
</div>
```

5. Insert `o:link` elements in the DocBook documents.

When editing a DocBook XML document in **Author** mode, the **Insert OLink** action is available in the **OLink** 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.

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.

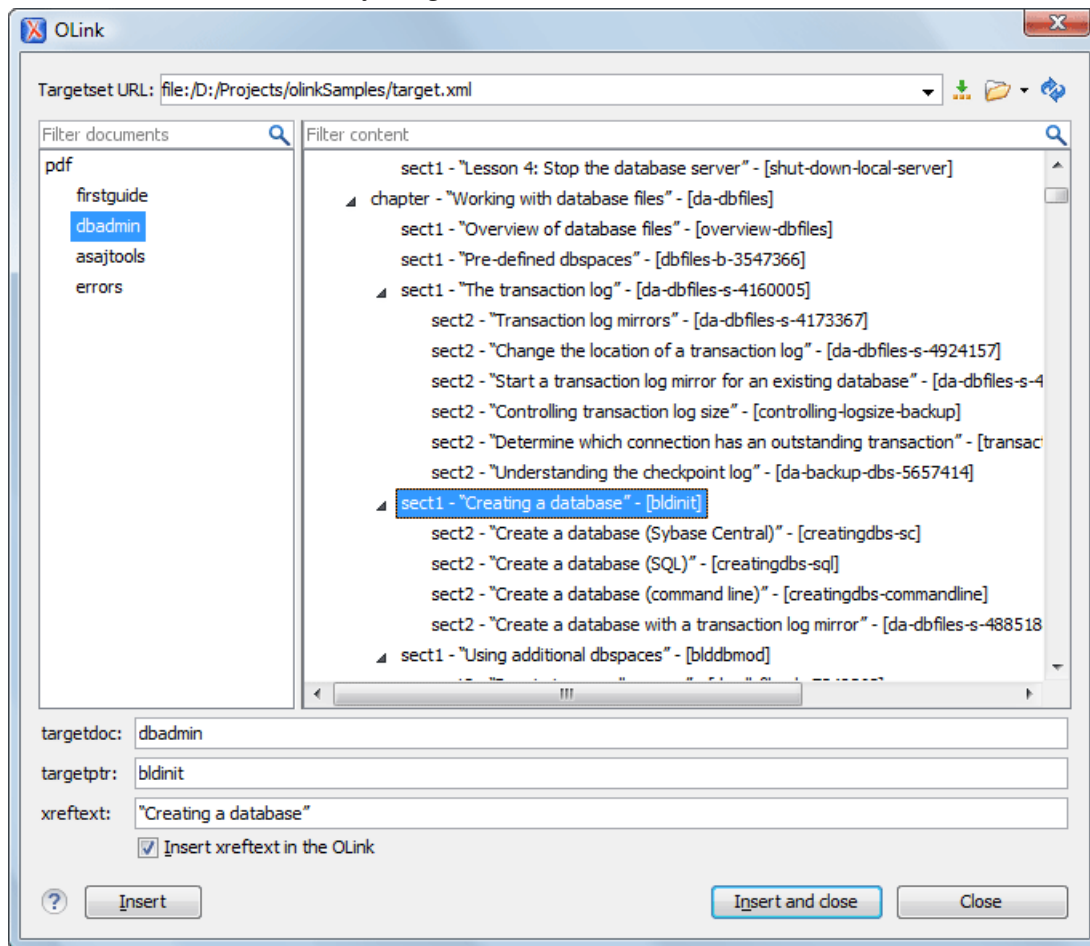


Figure 389: Insert OLink Dialog Box

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.

DITA Topics Document Type

The Darwin Information Typing Architecture (DITA) is an XML-based architecture oriented to 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. Oxygen XML Developer provides schema-driven (DTD, RNG, XSD) templates for DITA documents.

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 enabled from the [Document Type Association preferences page](#).

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/**.

The default 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**

DITA Topic Transformation Scenarios

The following default transformation scenarios are available for DITA Topics:

- **DITA XHTML** - Transforms a DITA topic to XHTML using DITA Open Toolkit.
- **DITA PDF** - Transforms a DITA topic to PDF using the DITA Open Toolkit and the Apache FOP engine.

Related information

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

[Editing a Transformation Scenario](#) on page 596

DITA Templates

The default templates available for DITA topics are stored in various sub-folders inside the **[OXYGEN_INSTALL_DIR]/frameworks/dita/templates/** folder. They can be used for easily creating a DITA concept, reference, task, topic, or other DITA documents.

Here are some of the DITA templates available when creating [new documents from templates](#):

- **Composite** - New DITA Composite
- **Concept** - New DITA Concept
- **General Task** - New DITA Task
- **Glossentry** - New DITA Glossentry
- **Glossgroup** - New DITA Glossgroup
- **Machinery Task** - New DITA Machinery Task
- **Reference** - New DITA Reference
- **Task** - New DITA Task
- **Topic** - New DITA Topic
- **Learning Assessment** - New DITA Learning Assessment (learning specialization in DITA 1.2)
- **Learning Content** - New DITA Learning Content (learning specialization in DITA 1.2)
- **Learning Summary** - New DITA Learning Summary (learning specialization in DITA 1.2)
- **Learning Overview** - New DITA Learning Overview (learning specialization in DITA 1.2)
- **Learning Plan** - New DITA Learning Plan (learning specialization in DITA 1.2)
- **Troubleshooting** - DITA 1.3 Troubleshooting specialization

DITA for Publishers topic specialization templates (available only when using the [built-in DITA-OT 1.8 version](#)):

- **D4P Article** - New DITA for Publishers article
- **D4P Chapter** - New DITA for Publishers chapter
- **D4P Concept** - New DITA for Publishers concept

- **D4P Conversion Configuration** - New DITA for Publishers conversion configuration
- **D4P Cover** - New DITA for Publishers cover
- **D4P Part** - New DITA for Publishers part
- **D4P Sidebar** - New DITA for Publishers sidebar
- **D4P Subsection** - New DITA for Publishers subsection
- **D4P Topic** - New DITA for Publishers topic

DITA Map Document Type

DITA maps are documents that collect and organize references to DITA topics to indicate the relationships between the topics. They can also serve as outlines or tables of contents for DITA deliverables and as build manifests for DITA projects.

Maps allow scalable reuse of content across multiple contexts. They can be used by information architects, authors, and publishers to plan, develop, and deliver content.

A file is considered to be a DITA map document when either of the following is 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 enabled.

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/`.

The default catalogs for the *DITA Map* document type are as follows:

- `[OXYGEN_INSTALL_DIR]/frameworks/dita/catalog.xml`
- `DITA_OT_DIR/catalog-dita.xml`

DITA Map Transformation Scenarios

The following default transformations scenarios are available:

- Predefined transformation scenarios allow you to transform a DITA map to a variety of outputs, such as PDF, ODF, XHTML, WebHelp, EPUB, and CHM files.
- **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 transformation scenario if you want to generate a DITA map statistics report containing 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 596

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

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 611

Support for Right-to-Left (RTL) Oriented Languages for DITA WebHelp

To activate support for RTL 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

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` - the date when the page was last modified. The date format is `YYYY-MM-DD`.
- `changefreq` - 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` - 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.



Note: `lastmod`, `changefreq`, and `priority` are optional elements.

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 Developer 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.

Indexing Japanese Content for DITA WebHelp Pages

To optimize the indexing of Japanese content in WebHelp pages generated from DITA map transformations, the Kuromoji analyzer can be used. This analyzer is not included in the Oxygen XML Developer installation kit and must be downloaded and added.

To use the Kuromoji analyzer to index Japanese content in your WebHelp system, follow these steps:

1. Download the analyzer jar file from
<http://mvnrepository.com/artifact/org.apache.lucene/lucene-analyzers-kuromoji/4.0.0>.
2. Place the Kuromoji analyzer jar file in the following directory:
`DITA_OT_DIR/plugins/com.oxygenxml.webhelp/lib`.
3. For the analyzer to work properly, search terms that are entered into your WebHelp pages must be separated by spaces.

Optionally a Japanese user dictionary can be set with the `webhelp.search.japanese.dictionary` parameter.

Related information

[DITA Map to WebHelp Output](#) on page 545

Compiled HTML Help (CHM) Output Format

To perform a **Compiled HTML Help (CHM)** transformation Oxygen XML Developer needs Microsoft HTML Help Workshop to be installed on your computer. Oxygen XML Developer 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 Developer uses the `htmlhelp.locale` parameter to correctly display specific characters of different languages in the output of the **Compiled HTML Help (CHM)** transformation. The **Compiled HTML Help (CHM)** default scenario that comes bundled with Oxygen XML Developer has the `htmlhelp.locale` parameter set to `en-US`.

The default value of the `htmlhelp.locale` is `en-US`. To customize this parameter, go to  **Configure**

Transformation Scenarios and click the  **Edit** button. In the parameter tab search for the `htmlhelp.locale` parameter and change its value to the desired language tag.

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 Developer requires KindleGen to generate Kindle output from DITA maps. To install KindleGen for use by Oxygen XML Developer, 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 Developer and open a DITA map in the **DITA Maps Manager** view.
4. In the **DITA Maps Manager** view, open the  **Configure Transformation Scenario(s)** dialog box.
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.

Migrating OOXML Documents to DITA

Oxygen XML Developer integrates the entire DITA for Publishers plugins suite that allows you to migrate content from Open Office XML documents to DITA.

You can use any of the following methods to migrate such documents to DITA:

- Open an OOXML document in Oxygen XML Developer. The document is opened in the **Archive Browser** view.
- From the **Archive Browser**, open `document.xml`.



Note: `document.xml` holds the content of the document.

- Click  **Configure Transformation Scenario(s)** on the toolbar and apply the **DOCX DITA** scenario. If you encounter any issues with the transformation, click the link below for further details about the Word to DITA Transformation Framework.

Related information

<http://dita4publishers.sourceforge.net/>

DITA Map Templates

The default templates available for DITA maps are stored in `[OXYGEN_INSTALL_DIR]/frameworks/dita/templates/map` folder.

Here are some of the DITA map templates available when creating *new documents from templates*:

- **DITA Map - Bookmap** - New DITA Bookmap.
- **DITA Map - Map** - New DITA map.
- **DITA Map - Learning Map** - New DITA learning and training content specialization map.
- **DITA Map - Learning Bookmap** - New DITA learning and training content specialization bookmap.
- **DITA Map - Eclipse Map** - IBM specialization of DITA map used for producing Eclipse Help plugins.

DITA for Publishers Map specialization templates:

- **D4P Map** - New DITA for Publishers map.
- **D4P Pub-component-map** - New DITA for Publishers pub-component-map.
- **D4P Pubmap** - New DITA for Publishers pubmap.

XHTML Document Type

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.

A file is considered to be an XHTML document when the root element name is `html`.

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/nvd/`.
- 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)/`.

The default 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/nvd/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`

XHTML Transformation Scenarios

The following default transformation scenarios are available for XHTML:

- **XHTML to DITA concept** - Converts an XHTML document to a DITA concept document.
- **XHTML to DITA reference** - Converts an XHTML document to a DITA reference document.
- **XHTML to DITA task** - Converts an XHTML document to a DITA task document.
- **XHTML to DITA topic** - Converts an XHTML document to a DITA topic document.

Related information

[Editing a Transformation Scenario](#) on page 596

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

XHTML Templates

Default templates are available for XHTML. They are stored in `[OXYGEN_INSTALL_DIR]/frameworks/xhtml/templates` folder and they can be used for easily creating basic XHTML documents.

Here are some of the XHTML templates available when creating [new documents from templates](#).

- **XHTML - 1.0 Strict** - New Strict XHTML 1.0
- **XHTML - 1.0 Transitional** - New Transitional XHTML 1.0
- **XHTML - 1.1 DTD Based** - New DTD based XHTML 1.1
- **XHTML - 1.1 DTD Based + MathML 2.0 + SVG 1.1** - New XHTML 1.1 with MathML and SVG insertions
- **XHTML - 1.1 Schema based** - New XHTML 1.1 XML Schema based

TEI ODD Document Type

The **Text Encoding Initiative - One Document Does it all (TEI ODD)** 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.

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`.

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/`.

The default 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`

TEI ODD Transformation Scenarios

The following default transformations are available:

- **TEI ODD XHTML** - Transforms a TEI ODD document into an XHTML document
- **TEI ODD PDF** - Transforms a TEI ODD document into a PDF document using the Apache FOP engine
- **TEI ODD EPUB** - Transforms a TEI ODD document into an EPUB document
- **TEI ODD DOCX** - Transforms a TEI ODD document into a DOCX document
- **TEI ODD ODT** - Transforms a TEI ODD document into an ODT document
- **TEI ODD RelaxNG XML** - Transforms a TEI ODD document into a RelaxNG XML document
- **TEI ODD to DTD** - Transforms a TEI ODD document into a DTD document
- **TEI ODD to XML Schema** - Transforms a TEI ODD document into an XML Schema document
- **TEI ODD to RelaxNG Compact** - Transforms a TEI ODD document into an RelaxNG Compact document

Related information

[Editing a Transformation Scenario](#) on page 596

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

TEI ODD Templates

There is only one default template that is stored in the `[OXYGEN_INSTALL_DIR]/frameworks/tei/templates/TEI ODD` folder and can be used for easily creating a basic TEI ODD document. This template is available when creating [new documents from templates](#).

- **TEI ODD** - New TEI ODD document

TEI P4 Document Type

The **Text Encoding Initiative (TEI) Guidelines** 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.

A file is considered to be a TEI P4 document when one of the following conditions are true:

- The local name of the root is `TEI . 2`.
- The public id of the document is `-//TEI P4`.

The default schema that is used if one is not detected in the TEI P4 document is `tei2.dtd` and it is stored in `[OXYGEN_INSTALL_DIR]/frameworks/tei/xml/teip4/schema/dtd/`.

The default catalogs for the TEI P4 document type are as follows:

- `[OXYGEN_INSTALL_DIR]/frameworks/tei/xml/teip4/schema/dtd/catalog.xml`
- `[OXYGEN_INSTALL_DIR]/frameworks/tei/xml/teip4/custom/schema/dtd/catalog.xml`
- `[OXYGEN_INSTALL_DIR]/frameworks/tei/xml/teip4/stylesheet/catalog.xml`

TEI P4 Transformation Scenarios

The following default transformations are available:

- **TEI HTML** - Transforms a TEI document into an HTML document.
- **TEI P4 - TEI P5 Conversion** - Convert a TEI P4 document into a TEI P5 document.
- **TEI PDF** - Transforms a TEI document into a PDF document using the Apache FOP engine.

Related information

[Editing a Transformation Scenario](#) on page 596

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

TEI P4 Templates

The default templates are stored in `[OXYGEN_INSTALL_DIR]/frameworks/tei/templates/TEI P4` folder and they can be used for easily creating basic TEI P4 documents. These templates are available when creating [new documents from templates](#).

- **TEI P4 - Lite** - New TEI P4 Lite
- **TEI P4 - New Document** - New TEI P4 standard document

TEI P5 Document Type

The **Text Encoding Initiative (TEI) Guidelines** 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.

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`.

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/`.

The default 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`

To watch our video demonstration about TEI editing, go to http://oxygenxml.com/demo/WYSIWYG_TEI_Editing.html.

TEI P5 Transformation Scenarios

The following default transformations are available:

- **TEI P5 XHTML** - transforms a TEI P5 document into an XHTML document.
- **TEI P5 PDF** - transforms a TEI P5 document into a PDF document using the Apache FOP engine.
- **TEI EPUB** - transforms a TEI P5 document into an EPUB output. The EPUB output will contain any images referenced in the TEI XML document.
- **TEI DOCX** - transforms a TEI P5 document into a DOCX (OOXML) document. The DOCX document will contain any images referenced in the TEI XML document.
- **TEI ODT** - transforms a TEI P5 document into an ODT (ODF) document. The ODT document will contain any images referenced in the TEI XML document.

Related information

[Editing a Transformation Scenario](#) on page 596

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

TEI P5 Templates

The default templates are stored in `[OXYGEN_INSTALL_DIR]/frameworks/tei/templates/TEI P5` folder and they can be used for easily creating basic TEI P5 documents. These templates are available when creating [new documents from templates](#):

- **TEI P5 - All** - New TEI P5 All.
- **TEI P5 - Bare** - New TEI P5 Bare.
- **TEI P5 - Lite** - New TEI P5 Lite.

- **TEI P5 - Math** - New TEI P5 Math.
- **TEI P5 - Speech** - New TEI P5 Speech.
- **TEI P5 - SVG** - New TEI P5 with SVG extensions.
- **TEI P5 - XInclude** - New TEI P5 XInclude aware.

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 Developer.

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 Developer `frameworks` folder.

JATS Document Type

The JATS (NISO Journal Article Tag Suite) document type is a technical standard that defines an XML format for scientific literature.

A file is considered to be a JATS document when the PUBLIC ID of the document contains the string - `//NLM//DTD`.

Default schemas that are used if one is not detected in the JATS document are stored in `[OXYGEN_INSTALL_DIR]/frameworks/jats/02-DTD/`.

The default XML catalog, `JATS-catalog-02.xml`, is stored in `[OXYGEN_INSTALL_DIR]/frameworks/02-DTD/`.

JATS Transformation Scenarios

The following default transformation scenario is available for JATS documents:

- **JATS Preview (simple HTML)** - Converts a JATS document to a simple HTML document.

Related information

[Editing a Transformation Scenario](#) on page 596

[Configure Transformation Scenario\(s\) Dialog Box](#) on page 598

JATS Templates

Default templates are available for JATS documents. They are stored in `[OXYGEN_INSTALL_DIR]/frameworks/jats/templates` folder and they can be used for easily creating basic JATS documents.

The default JATS templates that are available when creating *new documents from templates* are as follows:

- **Archiving** - JATS archiving tag set version 1.1.
- **Authoring** - JATS authoring tag set version 1.1.
- **Book** - JATS book tag set version 1.1.
- **Publishing** - JATS publishing tag set version 1.1.

EPUB Document Type

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 Developer supports both OPF 2.0 and OPF 3.0.

Document Templates

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.

The following EPUB templates are available when creating *new documents from templates*:

- **NCX - Toc** - New table of contents.
- **OCF - Container** - New container based OCF.
- **OCF - Encryption** - New encryption based OCF.
- **OCF - Signatures** - New signature based OCF.
- **OPF 2.0 - Content (2.0)** - New OPF 2.0 content.
- **OPF 3.0 - Content (3.0)** - New OPF 3.0 content.

DocBook Targetset Document Type

DocBook *Targetset* documents are used to resolve cross references with the DocBook `o:link`.

A file is considered to be a *Targetset* when the root name is `targetset`.

The default schema, `targetdatabase.dtd`, for this type of document is stored in `[OXYGEN_INSTALL_DIR]/frameworks/docbook/xsl/common/`.

Document Templates

The default template for DocBook *Targetset* documents is located in the `[OXYGEN_INSTALL_DIR]/frameworks/docbook/templates/Targetset` folder.

The following DocBook *Targetset* template is available when creating *new document from templates*:

- **DocBook Targetset - Map** - New Targetset map.

Chapter 15

Extending Oxygen XML Developer Using the SDK

Topics:

- *[Extending Oxygen XML Developer with Plugins](#)*

This chapter includes information about extending Oxygen XML Developer with plugins.

Oxygen XML Developer has an SDK that can be used as a base to develop frameworks and plugins. The SDK is available on our website at http://www.oxygenxml.com/oxygen_sdk.html.

This chapter includes information about extending Oxygen XML Developer with plugins.

Extending Oxygen XML Developer 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 Developer. 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 Developer Eclipse Plugin you can look at the [Eclipse IDE Integration Sample Project](#) to see how an Eclipse plugin can interact with the Oxygen XML Developer APIs.

Introduction to Plugins

A plugin can have multiple plugin extensions. The following plugin extensions are available (in the order of importance).

Plugins that are used in the entire workspace:

- [Workspace Access Plugin Extension](#) on page 822
- [JavaScript-based Workspace Access Plugin Extension](#) on page 823
- [Components Validation Plugin Extension](#) on page 825
- [Custom Protocol Plugin Extension](#) on page 825
- [Resource Locking Custom Protocol Plugin Extension](#) on page 826
- [Open Redirect Plugin Extension](#) on page 827

Plugins that work only in the **Text** editing mode:

- [General Plugin Extension](#) on page 829
- [Selection Plugin Extension](#) on page 829
- [Document Plugin Extension](#) on page 829

General Configuration of an Oxygen XML Developer Plugin

The Oxygen XML Developer functionality can be extended with plugins that implement a clearly specified API.

On the Oxygen XML Developer 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 Developer `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 Developer `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 Developer 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 type *Selection*, *Document* or *General*, 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 Developer **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).

Installation

To install a plugin in Oxygen XML Developer, follow these steps:

1. Go to the Oxygen XML Developer installation directory and locate the `plugins` directory.
The `plugins` directory contains all the plugins available to Oxygen XML Developer.
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 Developer.

Alternatively, you can go to **Help > Manage Addons** and use the **Manage Add-ons** dialog box. This dialog allows you to install available plugins and manage the ones that are currently in use.

Types of plugin extensions

A plugin can have one or more defined plugin extensions that provide functionality to the application.

Workspace Access Plugin Extension

This plugin type allows you to contribute actions to the main menu and toolbars of Oxygen XML Developer, to create custom views and 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 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="SampleWorkspaceAccessToolbarID" initialSide="NORTH" initialRow="1"/>
</plugin>
```

The `toolbar` element adds a toolbar in the Oxygen XML Developer interface and allows you to contribute your own plugin-specific actions. The following attributes are supported:

- `id` - Unique identifier for the plugin 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="SampleWorkspaceAccessID" initialSide="WEST" initialRow="0"/>
</plugin>
```

The `view` element adds a view in the Oxygen XML Developer 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 Developer, 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 allowed 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. 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`).

To be notified when editors are opened, selected, and closed, you can use the

`StandalonePluginWorkspace.addEditorChangeListener` API method to add a listener.

JavaScript-based Workspace Access Plugin Extension

This is a JavaScript-based plugin extension that allows you to contribute actions to the main menu and toolbars of Oxygen XML Developer, 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](#) on page 822, but the implementation is JavaScript-based and uses the bundled [Rhyno](#) library to create and work with Java API from the JavaScript code.

The plugin descriptor file `plugin.xml` needs to point to 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 contextual menu action to DITA Maps Manager pop-up 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 `pluginWorkspaceAccess.pluginWorkspaceAccess` 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);
    edChangeListener = {
        /*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 performed in the DITA Maps manager view*/
            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));
        }
        edChangeListener = new JavaAdapter(Packages.ro.sync.exml.workspace.api.listeners.WSEditorChangeListener,
            edChangeListener);
        pluginWorkspaceAccess.addEditorChangeListener(
            edChangeListener,
            Packages.ro.sync.exml.workspace.api.PluginWorkspace.DITA_MAPS_EDITING_AREA);
    }

    function applicationClosing(pluginWorkspaceAccess) {
        Packages.java.lang.System.err.println("Application closing " + pluginWorkspaceAccess);
    }
}
```

Option Page Plugin Extension

This extension type allows developers to add custom preference pages to the application **Preferences** dialog box.

The extension must implement the `ro.sync.exml.plugin.option.OptionPagePluginExtension` interface. The provided callbacks allow the developer to create the 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 appear listed in the **Preferences** dialog box in the **Plugins** category. The `plugin.xml` configuration file can specify one or more such extensions using constructs like this:

```
<extension type="OptionPage" class="my.package.CustomOptionPagePluginExtension"/>
```

Components Validation Plugin Extension

This plugin type allows the developer to customize the editor menus, toolbars, and other components by allowing 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.xml.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 Developer:
 - `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.
 - `validateAcceleratorAction(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 Developer 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()` - Check 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 Developer 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 the developer 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](#) on page 822 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 Developer.
 - `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.

Resource Locking Custom Protocol Plugin Extension

This plugin type allows the developer to work with a custom designed protocol for retrieving and storing files. It can lock a resource on opening it in Oxygen XML Developer. This type of plugin extends the custom protocol plugin type with resource locking support.

Such a plugin provides the following API:

- The interface `URLStreamHandlerWithLockPluginExtension` - The plugin receives callbacks following the simple protocol for resource locking and unlocking imposed by Oxygen XML Developer.

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`).

XML Refactoring Operations Plugin Extension

This plugin type allows the developer to specify one or more directories from which 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 Developer 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.

Sample 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>

```

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 Developer 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.

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 Developer 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.xml.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 Developer `URLStreamHandler` is used.

To use this type of extension in your plugin, create an extension of `TargetedURLHandler` type in your `plugin.xml` 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 Developer `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 Developer) 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
        // for which the 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;
    }
}
```

Lock Handler Factory Plugin Extension

This type of 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` 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>
```

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 plugin type allows the developer to invoke custom code and to interact with the workspace of Oxygen XML Developer.

This plugin is the most general plugin type. It 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 Developer.

Selection Plugin Extension

A selection plugin can be applied to both an XML document and a non-XML document. It works as follows: the user makes a selection in the editor, displays the contextual menu, and selects the item corresponding to the plugin from the **Plugins** submenu.

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 Developer.
 - `getDocumentURL()` - Returns the URL of the current edited document.

Related information

[Editor Variables](#) on page 154

Document Plugin Extension

This plugin type can only be applied to an XML document. It can modify the current document that is received as callback parameter.

The plugin is started by selecting the corresponding menu item from the contextual menu of the XML editor (**Text** mode), **Plugins** submenu. 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 Developer.

Oxygen XML Developer 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 Developer 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 Developer and the CMS are as follows:

- Contribute to the Oxygen XML Developer 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 Developer 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 Developer. 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 The plugin workspace access (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 the 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.
}
```

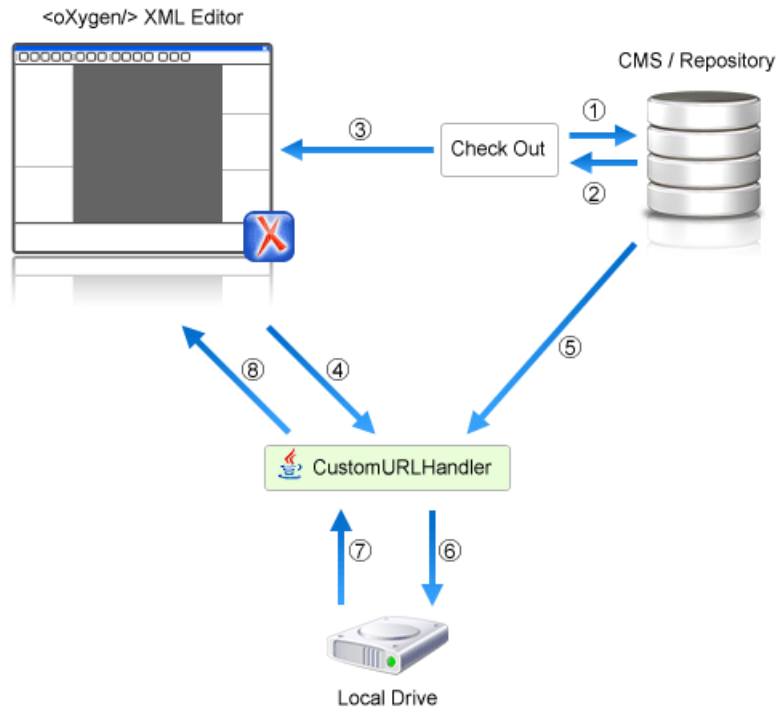


Figure 390: Check Out Process Diagram

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 Developer 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 Developer, 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. 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 "ro.sync.sample.plugin.workspace.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 "ro.sync.sample.plugin.workspace.customprotocol.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 Developer install folder and start Oxygen XML Developer.

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 Developer. To instruct the plugin to prefer its libraries over the ones used by Oxygen XML Developer, you can add the following attribute on the `<plugin>` root element:

`classLoaderType="preferReferencedResources"` from the `plugin.xml` descriptor.

A **Late Delegation Class Loader** (the main class loader in Oxygen XML Developer) 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 Developer 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 Developer 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 Developer `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 to jar: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

For creating a custom protocol plugin, apply the following 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 Developer 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. 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 Developer install folder.

Pack and Deploy 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 (JAR)*. 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 *JAR* 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 Developer 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 <http://www.oxygenxml.com/InstData/Addons/optional/updateSite.xml> as described in the *Installing Add-ons procedure*.
2. Restart Oxygen XML Developer. 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/JAR file and the descriptor file to an HTTP server. The URL to this location serves as the *Update Site URL*.

How to Share the Classloader Between a Framework and a Plugin

In some cases you may need to extend the functionality of Oxygen XML Developer 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 classloader 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 classed 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.

Example - A Selection Plugin

The following plugin is called `UppercasePlugin` and is an example of *selection plugin*. It is used in Oxygen XML Developer for capitalizing the characters in the current selection. This example consists of two Java classes and the plugin descriptor:

- `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>

```

Creating and Running Automated Tests

If you have developed complex custom plugins or 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 Developer 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 frameworks directory where the custom framework resides.
 - File `pluginsFolder` - The file path to the plugins directory. It can point to a custom plugins directory where the custom plugins 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=MCwCFGNoEGJSeiC3XCyIya lvjzHhGhhqAhRNRDpEu8RIWb8icCJO7HqfVP4++A\\=\n" +
            "-----END-LICENSE-KEY-----",
            PluginWorkspaceTCBase.XML_AUTHOR_PRODUCT);
    }

    /**
     * <p><b>Description:</b> TC for opening a file and using the 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 the plugin without having to re-bundle the Java classes of the plugin in a JAR library, follow these steps:

1. Download and unpack an [all platforms standalone version](#) of Oxygen XML Developer.
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` 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.xml.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.xml.EditorFrameDescriptor -Xmx1024m
-XX:MaxPermSize=384m -Dcom.oxygenxml.editor.plugins.dir=D:\projects\MyPluginProject
```



Note: If you need to configure the plugin for Oxygen XML Developer, set the `com.oxygenxml.app.descriptor` to `ro.sync.xml.AuthorFrameDescriptor` or `ro.sync.xml.DeveloperFrameDescriptor`, respectively.

8. Add a break point in the source of one of your Java classes.
9. Debug the created configuration. When the code reaches your breakpoint, the debug 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.

Chapter

16

Tools

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*](#)
- [*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*](#)
- [*SVN Client*](#)
- [*External Tools*](#)

This chapter presents the various tools that are included in Oxygen XML Developer.

Oxygen XML Developer includes a variety of helpful tools to help you accomplish XML-related tasks. This section presents many of those tools.

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 Developer 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.



Note: The predefined refactoring operations are also available from the **Refactoring** submenu in the contextual menu of **Text** mode. This is useful because by selecting the operations from the contextual menu, Oxygen XML Developer 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 are used also appear in the **Refactoring** submenu of the contextual menu in the **Project** view.

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.

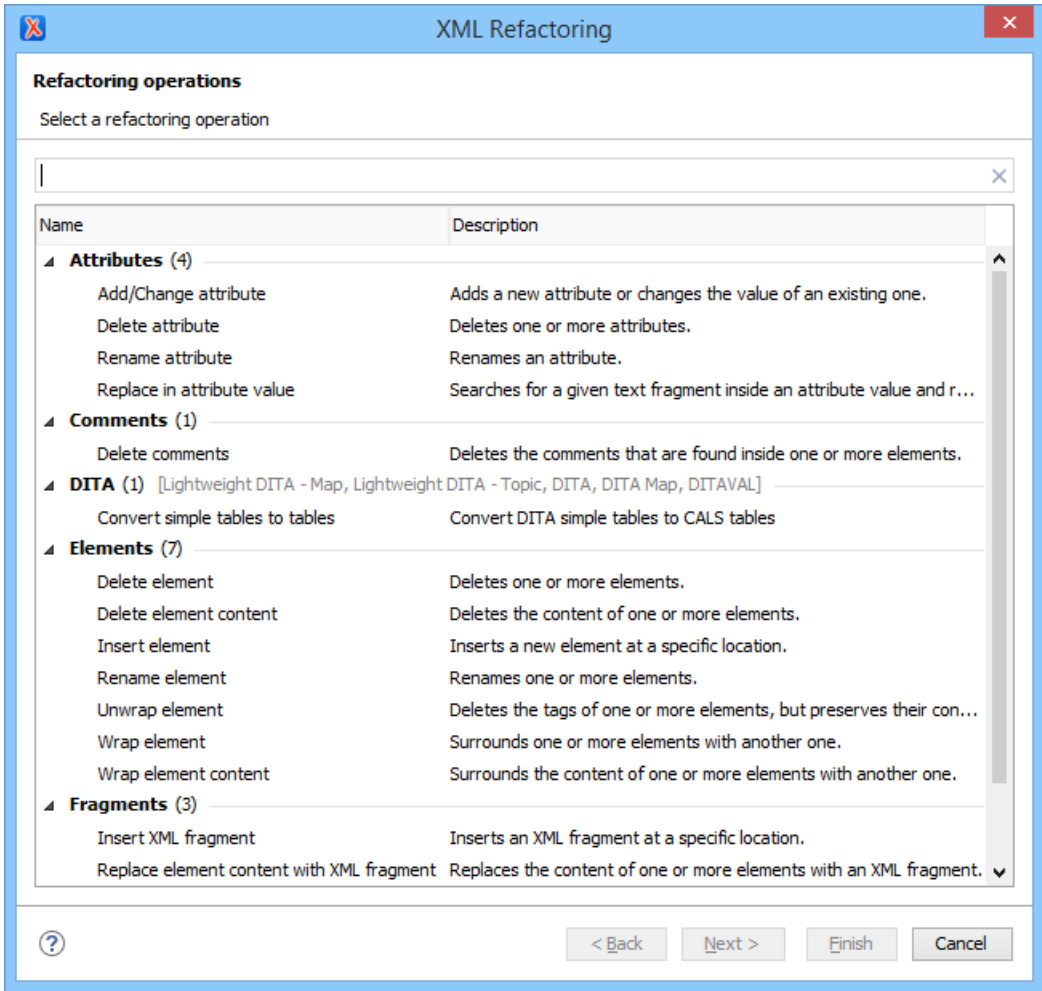


Figure 391: XML Refactoring Wizard

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.

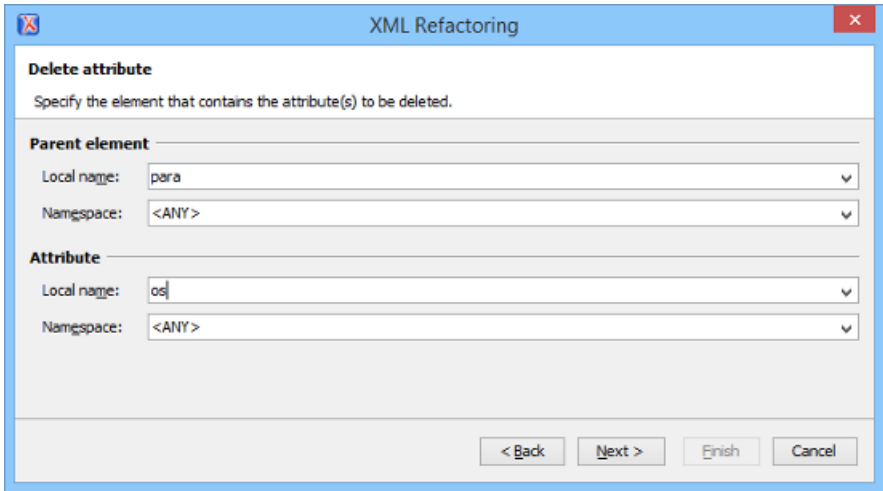


Figure 392: XML Refactoring 2nd Wizard Page (Delete Attribute Operation)

Scope and Filters The last wizard page allows you to select the set of files that represent the input of the operation. You can select from predefined resource sets (such as the current file, your whole project, the current DITA map hierarchy, etc.) or you can define your own set of resources by creating a working set.

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 enabled, only resources with a known XML file type will be affected by the operation.
- **Look inside archives** - When enabled, the resources inside archives will also be affected.

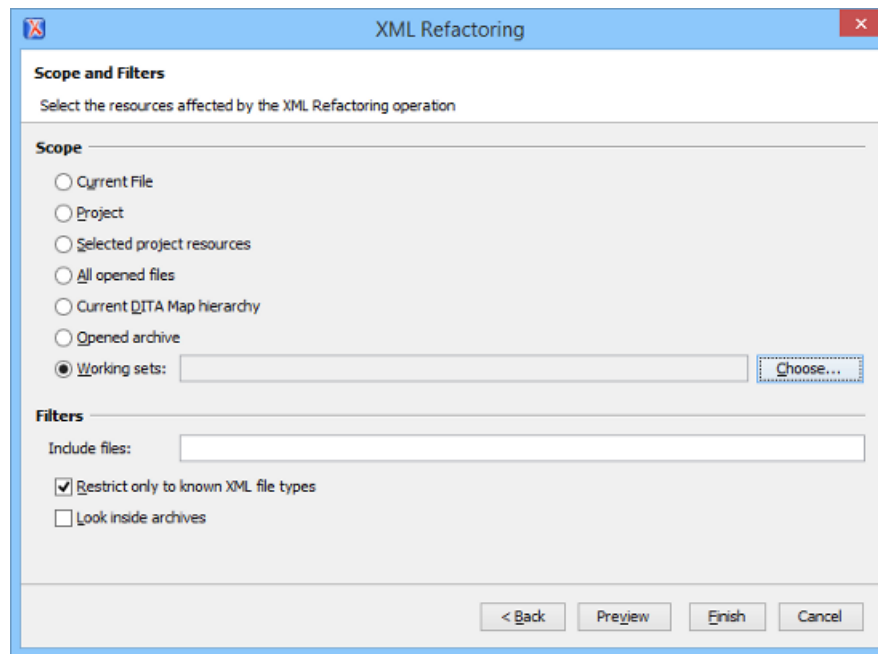


Figure 393: XML Refactoring - Scope and Filters Wizard Page

If an operation takes longer than expected you can use the  **Stop** button in the progress bar to cancel the operation.



Note: It is recommended that you use the **Preview** button to review all the changes that will be made by the refactoring operation before applying the changes.



Warning: After clicking the **Finish** button, the operation will be processed and Oxygen XML Developer provides no automatic means for reverting the operations. Any **Undo** action will only revert changes on the current document.

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 **Text** mode. The operations are also grouped by category in this submenu. When selecting the operations from the contextual menu, Oxygen XML Developer 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 attribute 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 *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.

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 Developer 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 Developer 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).

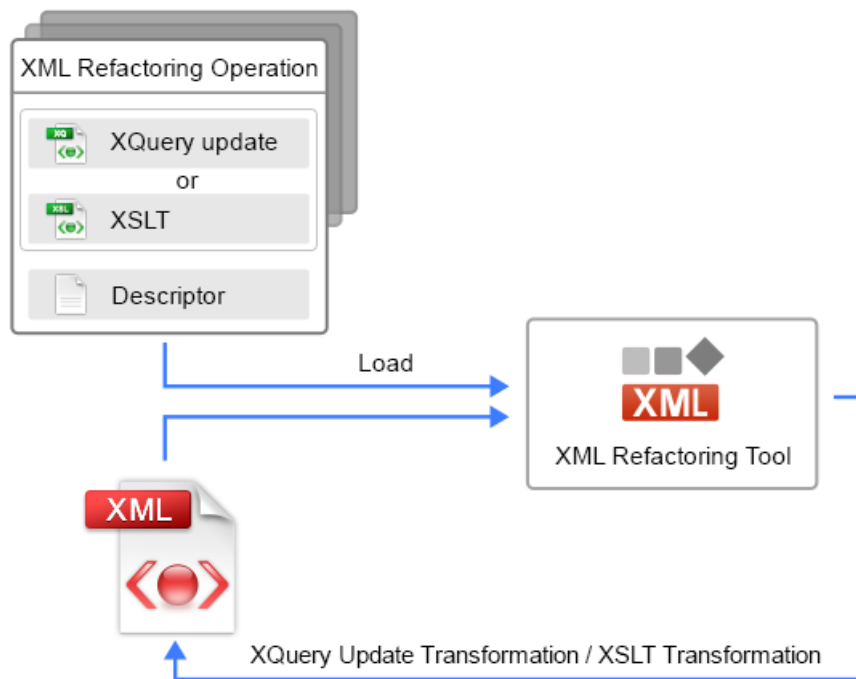


Figure 394: Diagram of an XML Refactoring Operation

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 Developer 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.
 - 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://oxygenxml.com/app/xml_refactory/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 performed on *comments* or *processing instructions* that are inside the root element.

Creating a Custom Refactoring Operation

To create a custom refactoring operation, follow these steps:

1. [Create an XQuery Update script or XSLT file.](#)
2. [Create an XML Refactoring Operation Descriptor file.](#)
3. Store both files in [one of the locations that Oxygen XML Developer](#) 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 333

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 examples](#) 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 Developer. Thus, if you want to create or edit an XQuery Update script you have to enable this facility 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 enabled 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 a custom operation descriptor file](#).

Related information

[Example of an XML Refactoring Operation](#) on page 330

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 Developer 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 Developer 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 Developer 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 Developer 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>The local name of the attribute's parent element.</description>
  </localName>
  <namespace label="Namespace" name="element_namespace" allowsAny="true">
    <description>The local name of the attribute's 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>The 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 XML Refactoring Operation](#) on page 330

Example of an XML Refactoring 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.

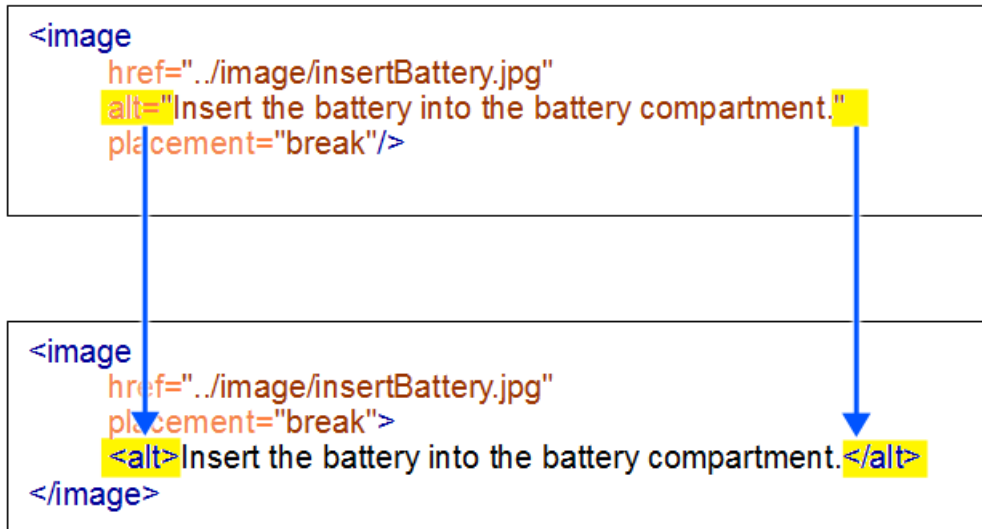


Figure 395: Example: Custom XML Refactoring Operation

A new custom XML refactoring operation requires:

- An [XQuery Update script](#) or [XSLT stylesheet](#).
- An [XML Refactoring operation descriptor file](#) that contains the path to the XQuery Update script or XSLT stylesheet.

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/attribute[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 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: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 for Creating a Custom Operation to Convert an Attribute to an Element

After you have developed the XQuery script or 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.xml"/> -->
  <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>The local name of the attribute's parent element.</description>
        </localName>
        <namespace label="Namespace" name="element_namespace" allowsAny="true">
          <description>The local name of the attribute's parent element</description>
        </namespace>
      </elementParameter>
    </section>
    <section label="Attribute">
      <attributeParameter dependsOn="elemID">
        <localName label="Name" name="attribute_localName">
          <description>The name of the attribute to be converted.</description>
        </localName>
        <namespace label="Namespace" name="attribute_namespace" allowsAny="true">
          <description>The 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.xml"/>
```

Results

After you have created these files, copy them into a folder *scanned by Oxygen XML Developer 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:

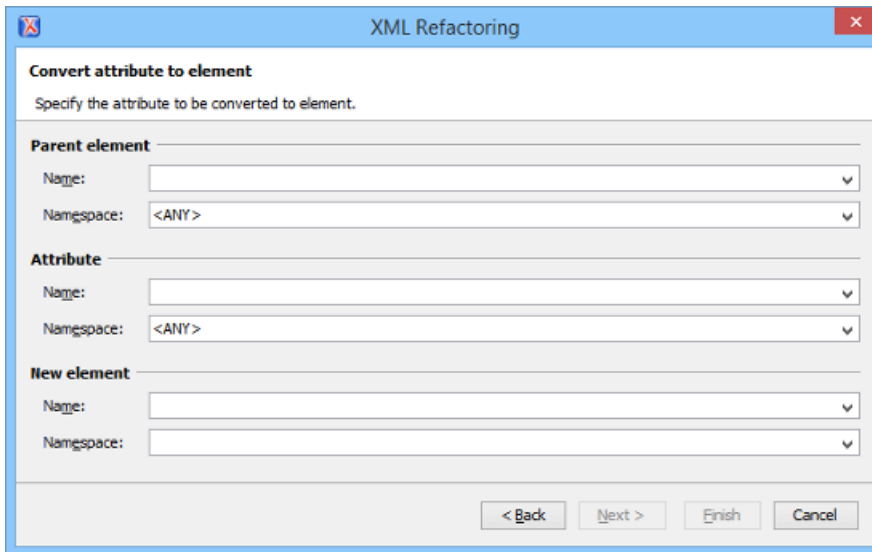


Figure 396: Example: XML Refactoring Wizard for a Custom Operation

Storing and Sharing Refactoring Operations

Oxygen XML Developer 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.
- Any folder. In this case, you need to [open the Preferences dialog box \(Options > Preferences\)](#), go to **XML > XML Refactoring**, and specify the same folder in the [Load additional refactoring operations from text box](#).



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 Developer installation directory (`[OXYGEN_INSTALL_DIR]/refactoring/`).

Sharing Custom Refactoring Operations

The purpose of Oxygen XML Developer 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 frameworks or projects.

After storing custom operations, you can share them with other users by sharing the resources.

Localizing XML Refactoring Operations

Oxygen XML Developer 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 possible values for each parameter.

Translated refactoring information uses the following form:

```
${i18n(translation_key)}
```

Oxygen XML Developer 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 Developer installation directory (`[OXYGEN_INSTALL_DIR]/refactoring/i18n`).

Example of a Refactoring Operation Descriptor File with *i18n* Support

```
<?xml version="1.0" encoding="UTF-8"?>

<refactoringOperationDescriptor xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://oxygenxml.com/app/xml_refactory
http://oxygenxml.com/app/xml_refactory/operation_descriptor.xsd"
  xmlns="http://oxygenxml.com/app/xml_refactory" 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 Developer 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 http://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.

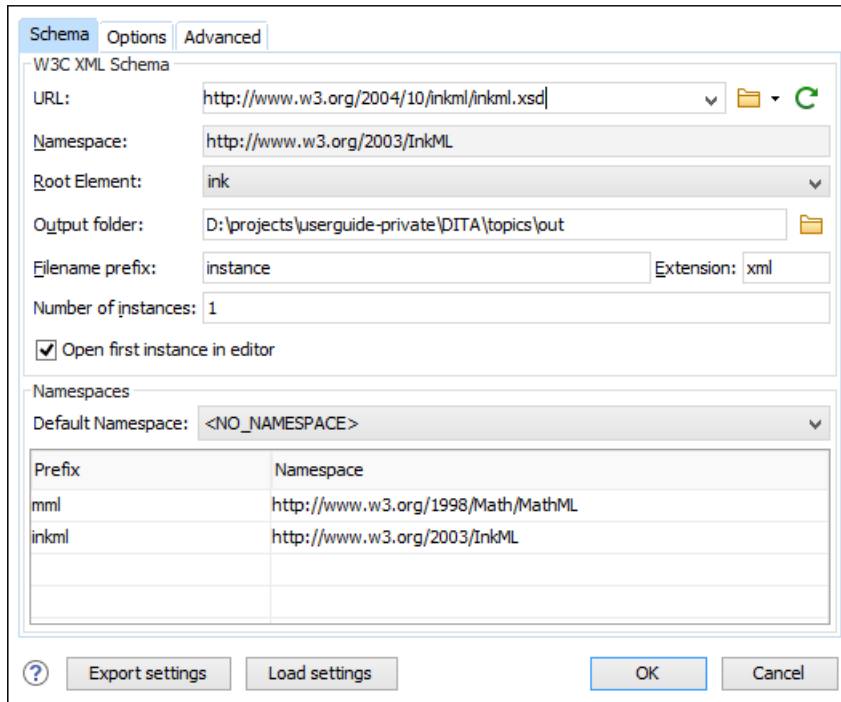


Figure 397: Generate Sample XML Files Dialog Box (Schema Tab)

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: <code>prefixN.extension</code> , where N represents an incremental number from 0 up to the specified <i>Number of instances</i> .
Number of instances	The number of XML files to be generated.
Open first instance in editor	When checked, 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.
Load settings	Use this button to load previously exported settings.
Export settings	Use this button to save the current settings for future use.

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.

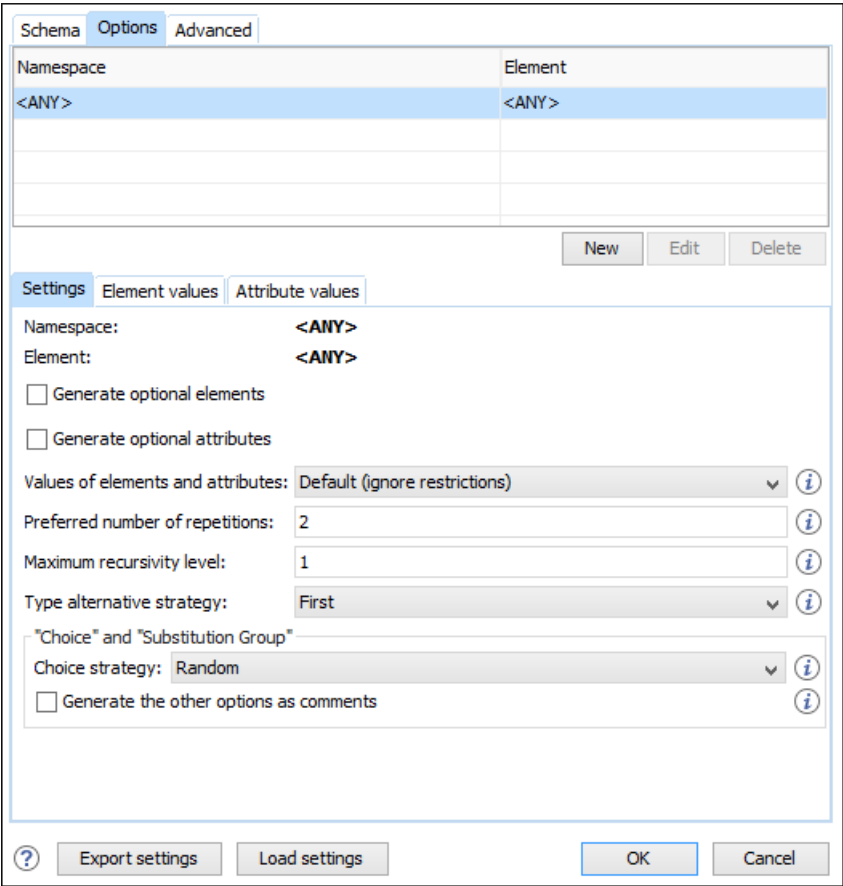


Figure 398: Generate Sample XML Files Dialog Box (Options Tab)

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 checked, all elements are generated, including the optional ones (having the minOccurs attribute set to 0 in the schema).
	Generate optional attributes	When checked, 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 **XML Instances Generator** preferences page). Note that type restrictions are ignored when this option is enabled. 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 **XML Instances Generator** 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 <code>minOccurs</code> and <code>maxOccurs</code> facets defined in the XML Schema. • If the value set here is between <code>minOccurs</code> and <code>maxOccurs</code>, then that value is used. • If the value set here is less than <code>minOccurs</code>, then the <code>minOccurs</code> value is used. • If the value set here is greater than <code>maxOccurs</code>, then <code>maxOccurs</code> 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 <code>xs:alternative</code> 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 <code>xs:choice</code> or <code>substitutionGroup</code> elements. The possible strategies are: • First - The first branch of <code>xs:choice</code> or the head element of <code>substitutionGroup</code> is always used. • Random - A random branch of <code>xs:choice</code> or a substitute element or the head element of a <code>substitutionGroup</code> is used.
Generate the other options as comments	If enabled, generates the other possible choices or substitutions (for <code>xs:choice</code> and <code>substitutionGroup</code>). 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.

Load settings Use this button to load previously exported settings.

Export settings Use this button to save the current settings for future use.

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.

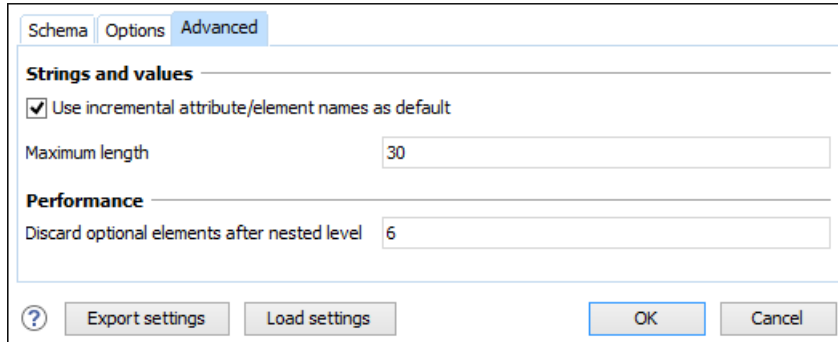


Figure 399: Generate Sample XML Files Dialog Box (Advanced Tab)

This tab includes the following options:

Use incremental attribute / element names as default	If checked, the value of an element or attribute starts with the name of that element or attribute. For example, for an <code>a</code> element the generated values are: <code>a1</code> , <code>a2</code> , <code>a3</code> , and so on. If not checked, the value is the name of the type of that element / attribute (for example: <code>string</code> , <code>decimal</code> , 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.

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 Developer 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:

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>
</settings>
```

```

<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 Developer generates an approximation of the source schema. Oxygen XML Developer 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.

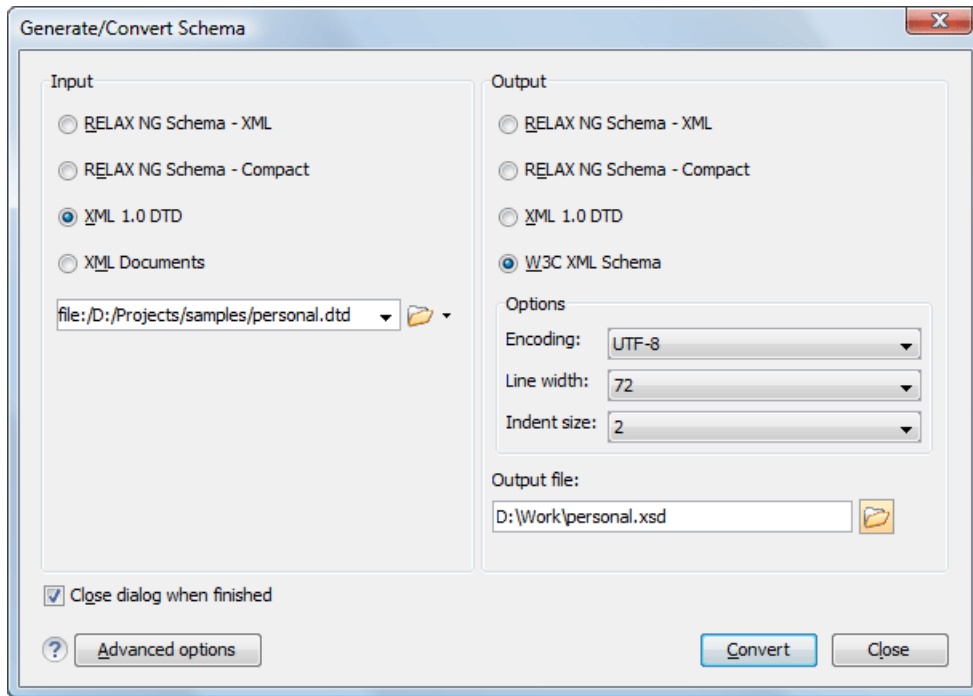


Figure 400: Generate/Convert Schema Dialog Box

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 This section is available if you select **W3C XML Schema** for the output.

Schema Output section

- **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 Developer 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 Developer 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.

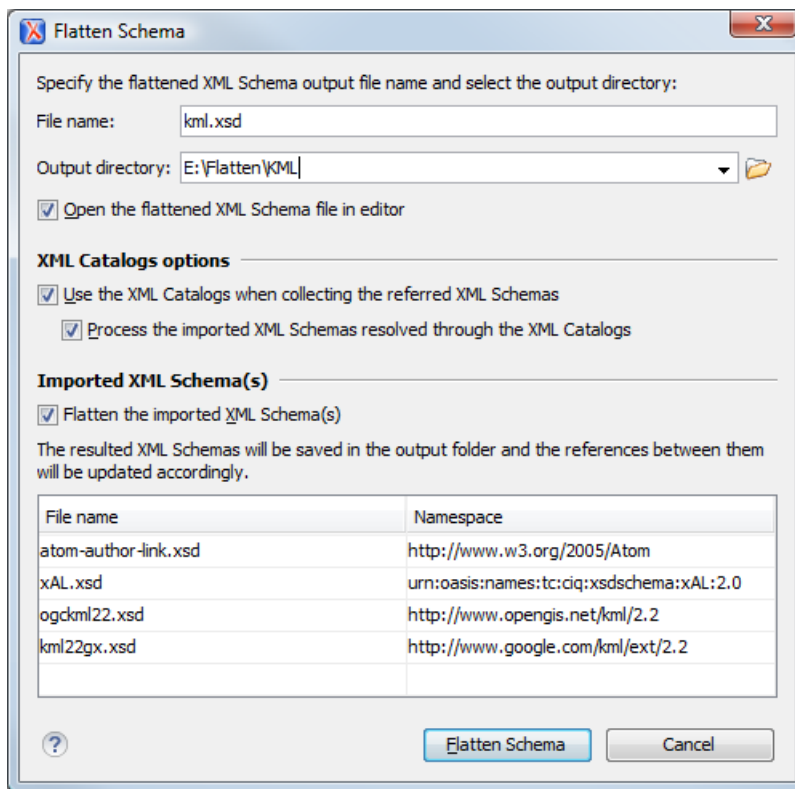


Figure 401: Flatten Schema Dialog Box

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 Developer 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.

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
```

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
```

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 Developer 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:

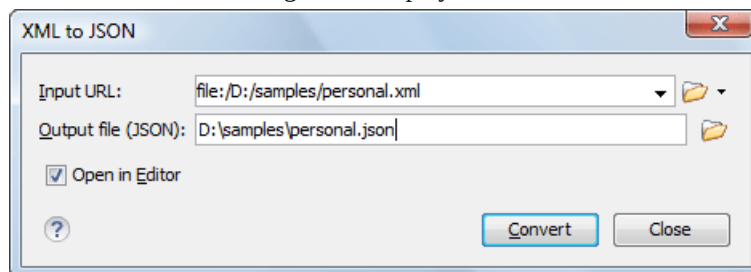


Figure 402: XML to JSON Dialog Box

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. Check the **Open in Editor** option to open the JSON result of the conversion in the Oxygen XML Developer JSON Editor.
5. Click the **Convert** button.

The original XML document is now converted to a JSON document.

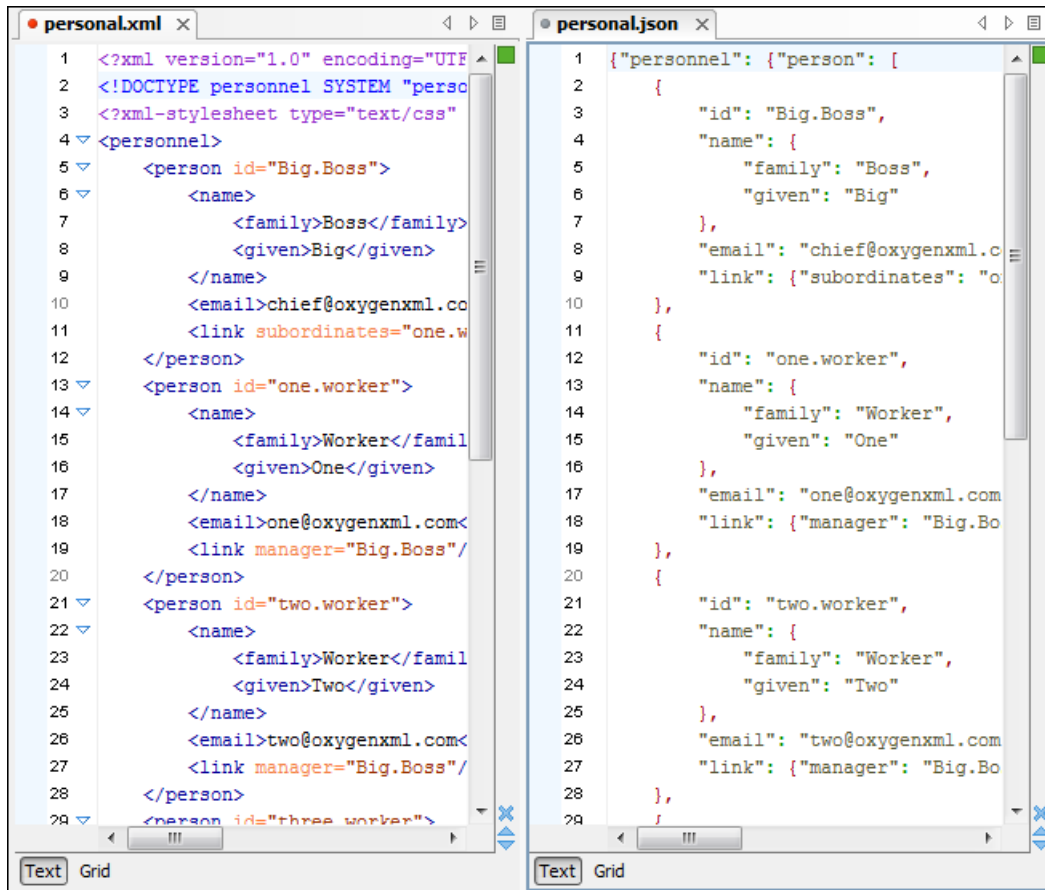


Figure 403: Example: XML to JSON Operation Result

Format and Indent (Pretty Print) Multiple Files

Oxygen XML Developer provides support for formatting and indenting (*pretty printing*) 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.

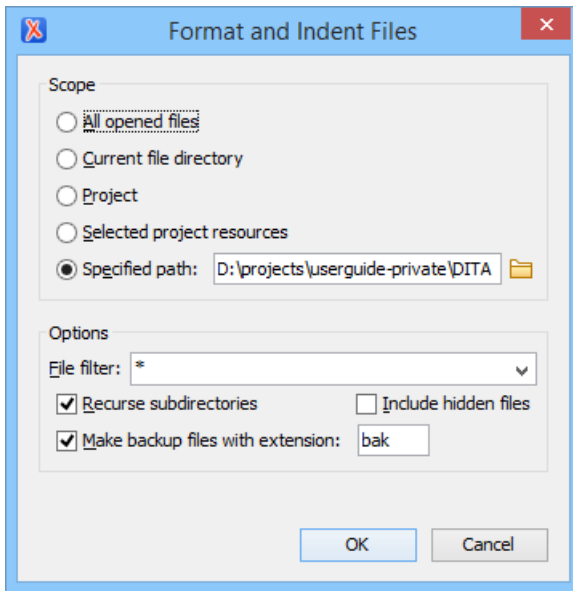


Figure 404: Format and Indent Files Dialog Box

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** - *Pretty prints* 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 enabled, 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 enabled, the *pretty print* is also performed in the hidden files.
- **Make backup files with extension** - When enabled, Oxygen XML Developer 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 Developer includes a tool for generating documentation for XSLT, XML Schema, XQuery, and WSDL documents.

Generating Documentation for an XML Schema

Oxygen XML Developer 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.

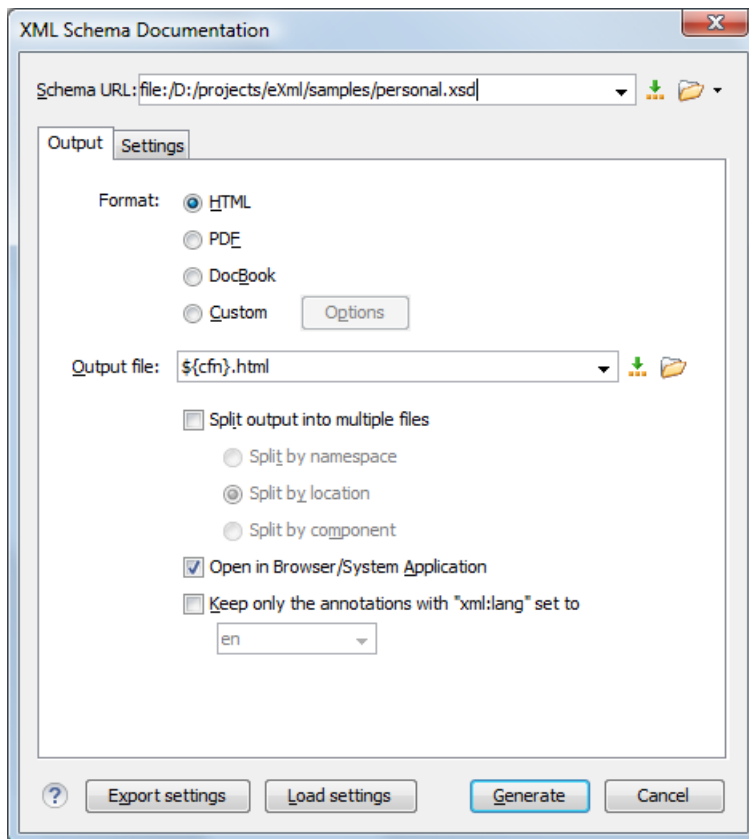


Figure 405: XML Schema Documentation Dialog Box

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](#).
 - **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.



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.

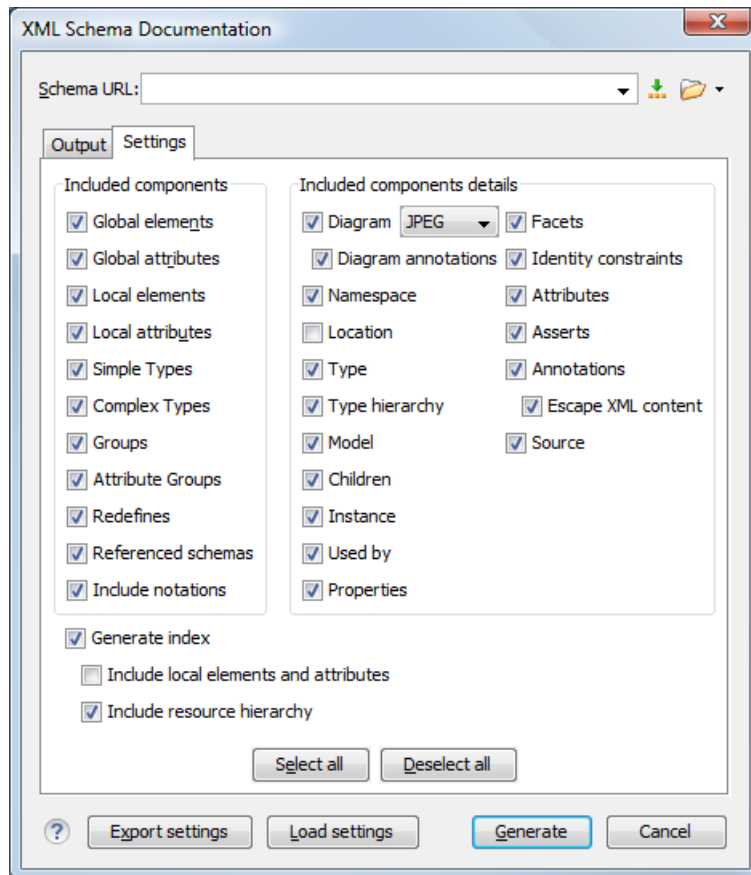


Figure 406: Settings Tab of the XML Schema Documentation Dialog Box

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 checked, 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 disabled 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](#).)

Load 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 426

Generating Documentation for an XSLT Stylesheet

You can use Oxygen XML Developer 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.

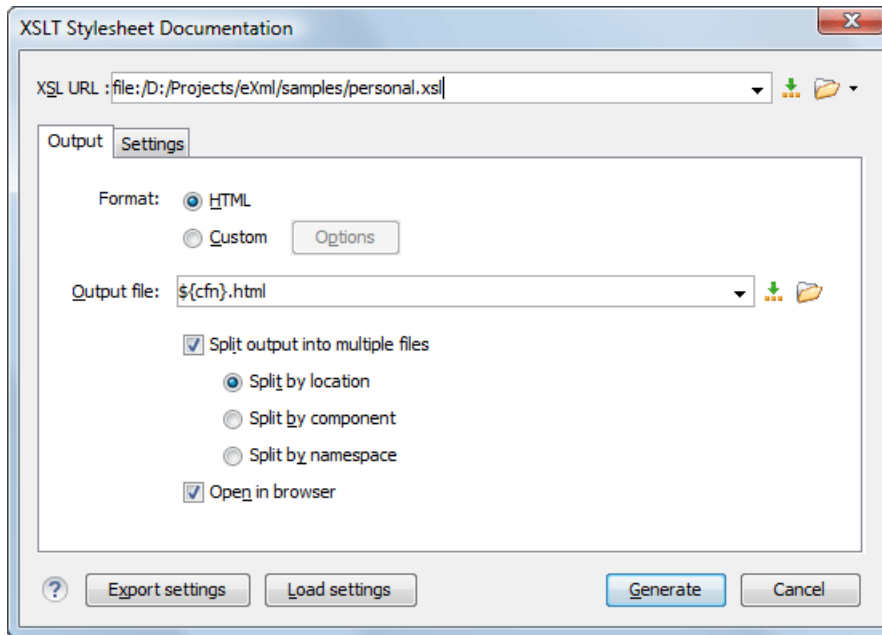


Figure 407: XSLT Stylesheet Documentation Dialog Box

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.

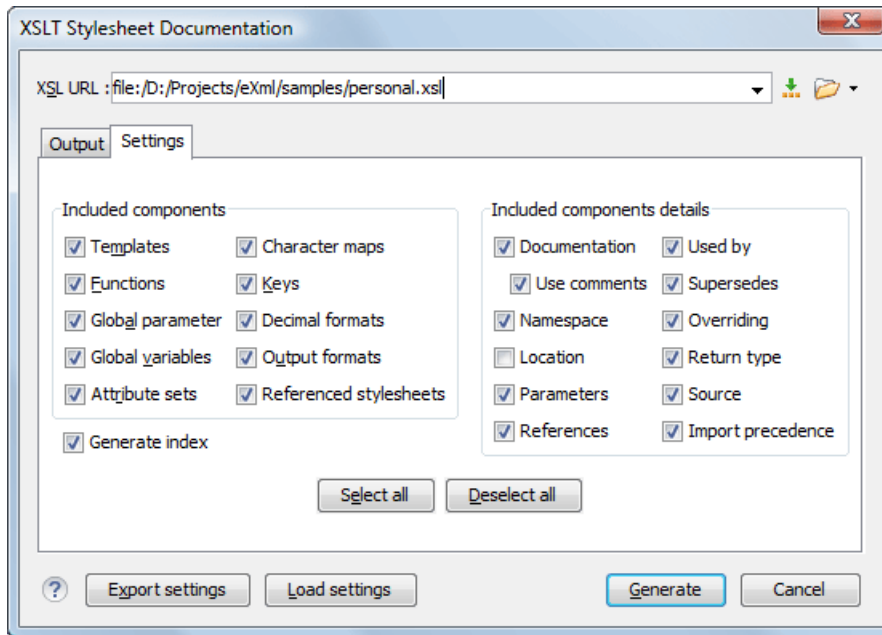


Figure 408: Settings Tab of the XSLT Stylesheet Documentation Dialog Box

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 Developer 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](#).)

Load settings - Reloads the settings from the exported file.

Generate - Use this button to generate the XSLT documentation.

Related information

[XSLT Stylesheet Documentation Support](#) on page 350

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.

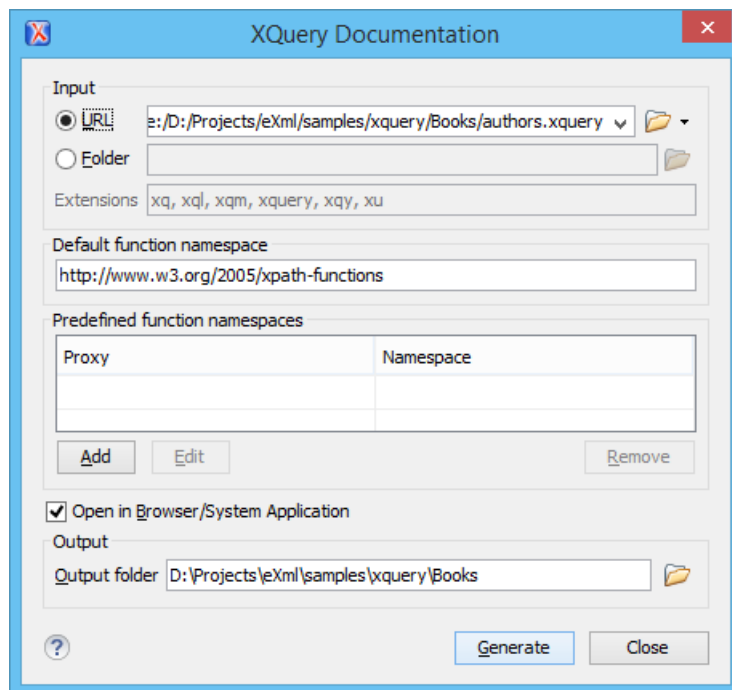


Figure 409: XQuery Documentation Dialog Box

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 Developer 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.

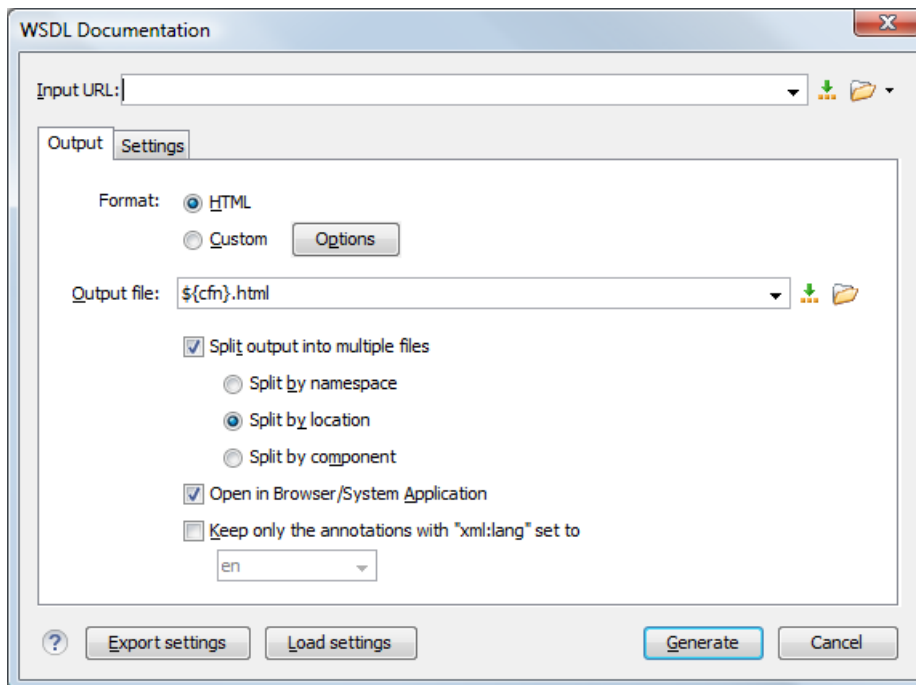


Figure 410: WSDL Documentation Dialog Box

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.

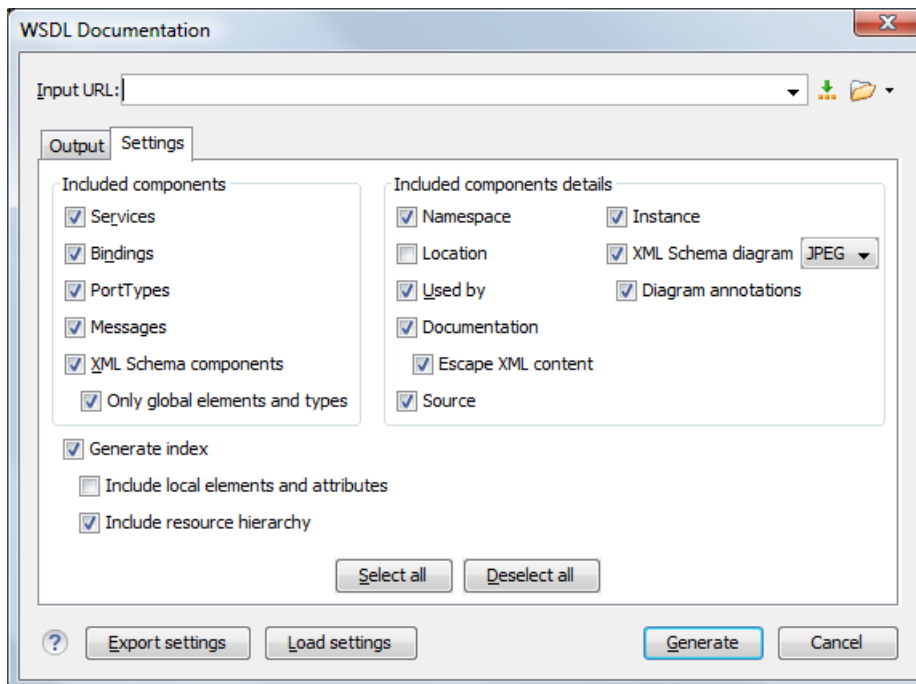


Figure 411: Settings Tab of the WSDL Documentation Dialog Box

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 checked, 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 disabled 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](#).)

Load 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.

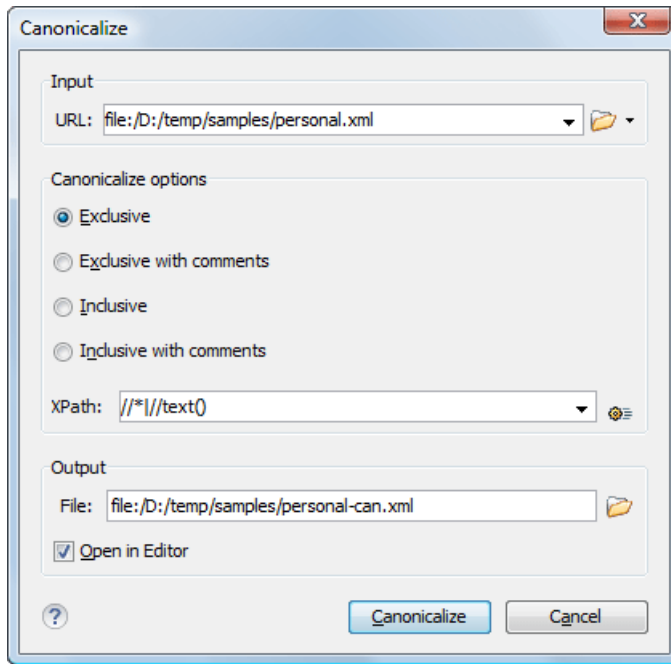


Figure 412: Canonicalization Settings Dialog Box

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 perspective 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 checked, the output file will be opened in the editor.

Related information

[Digital Signatures Overview](#) on page 513

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.

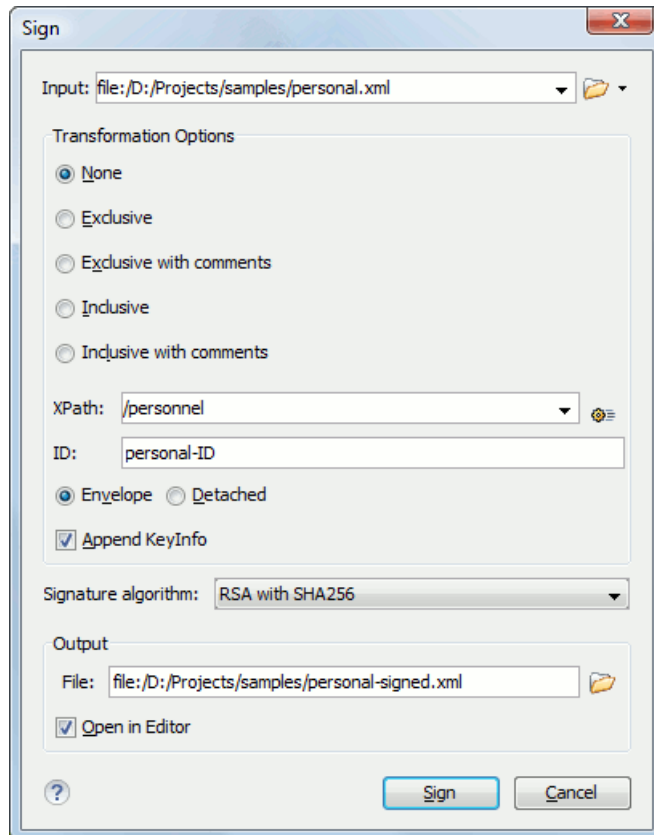


Figure 413: Signature Settings Dialog Box

The following options are available:

 **Note:** If Oxygen XML Developer 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.

 Could not obtain a valid certificate. [You must 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 perspective 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 checked, 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 checked, the output file will be opened in Oxygen XML Developer.

Related tasks

[Example of How to Digitally Sign XML Files or Content](#) on page 518

Related information

[Digital Signatures Overview](#) on page 513

[Verifying Signature](#) on page 518

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 tasks

[Example of How to Digitally Sign XML Files or Content](#) on page 518

Related information

[Digital Signatures Overview](#) on page 513

[Signing Files](#) on page 516

WSDL SOAP Analyzer

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).

Composing a SOAP Request

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.

Oxygen XML Developer 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.

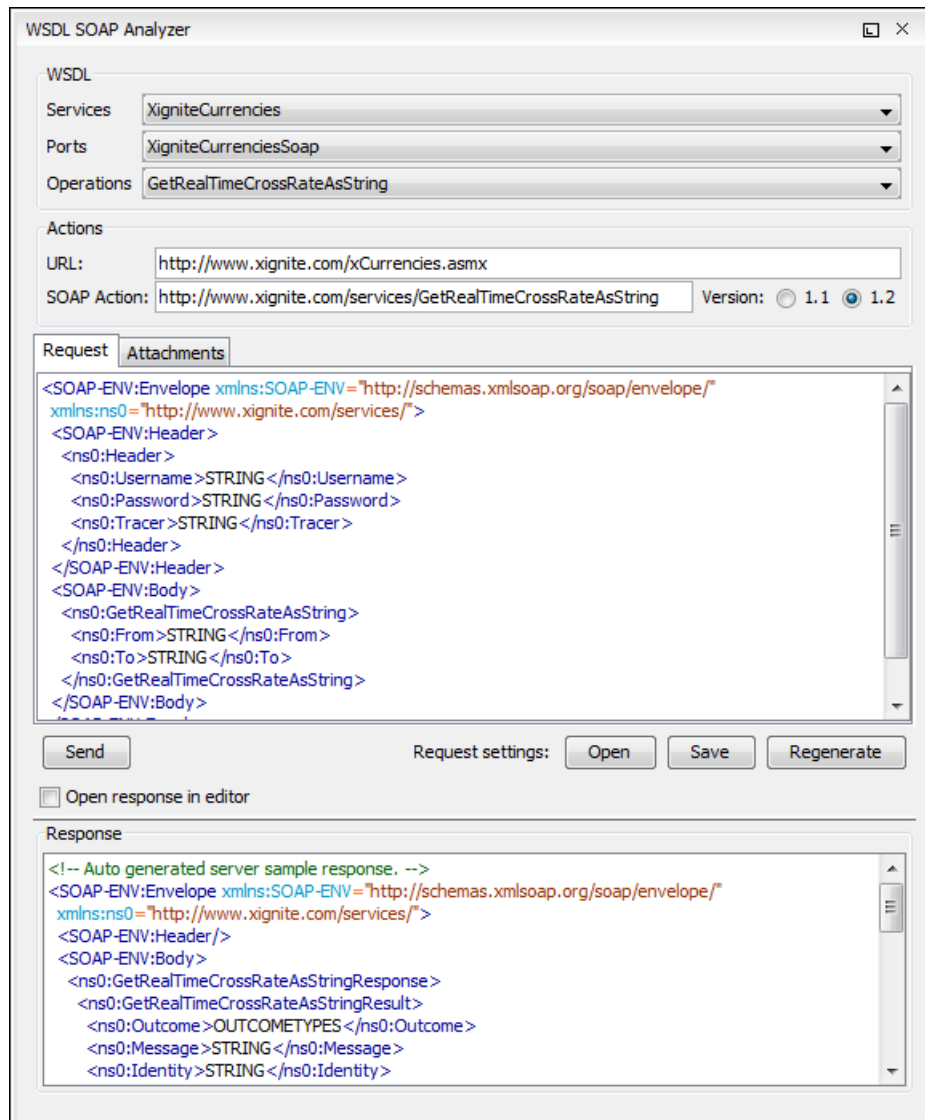


Figure 414: WSDL SOAP Analyzer Dialog Box

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 Developer 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 Developer 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 Developer 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 Developer 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.

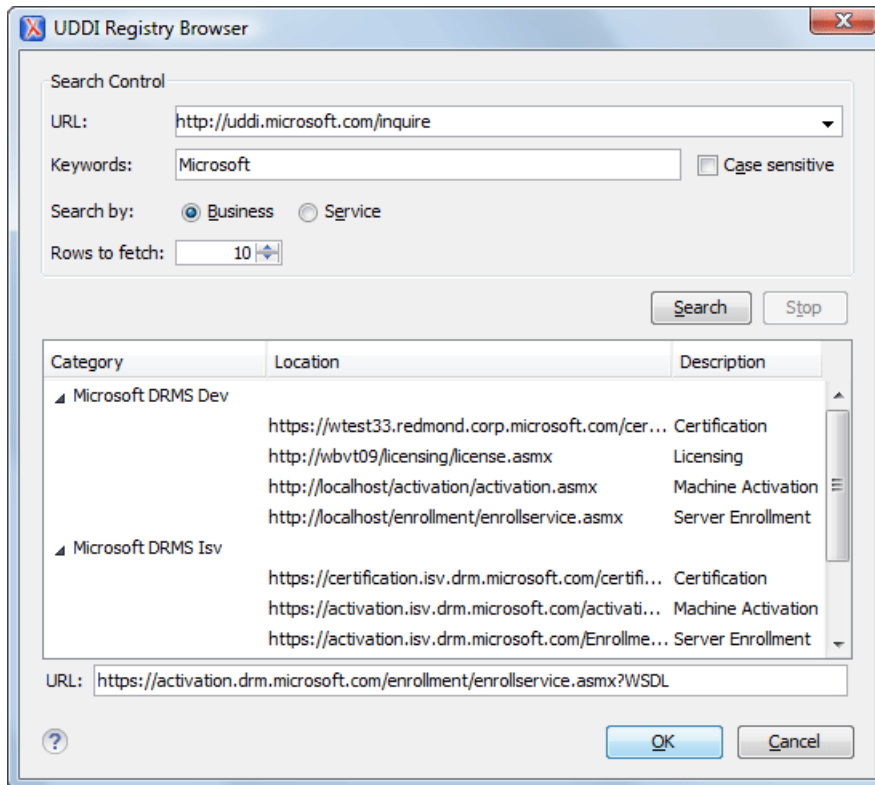


Figure 415: UDDI Registry Browser Dialog Box

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 checked, 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.

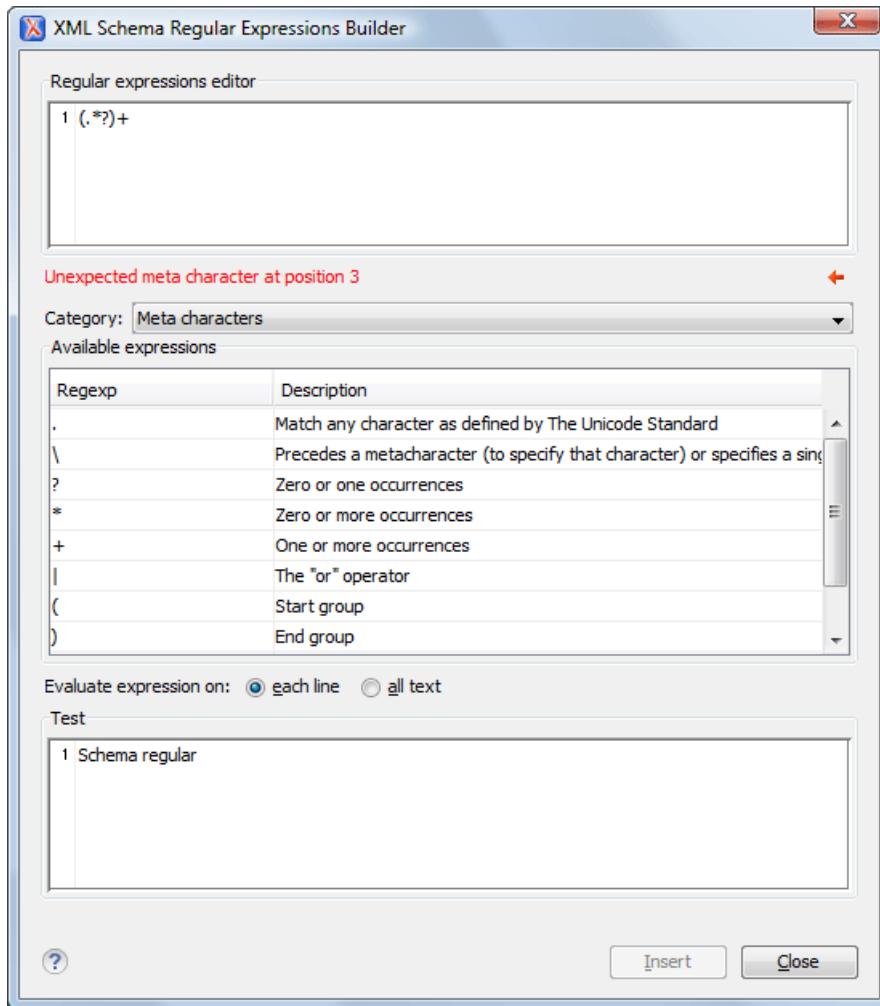


Figure 416: XML Schema Regular Expressions Builder Dialog Box

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 disabled 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 Developer options. See [Encoding for non XML files](#).

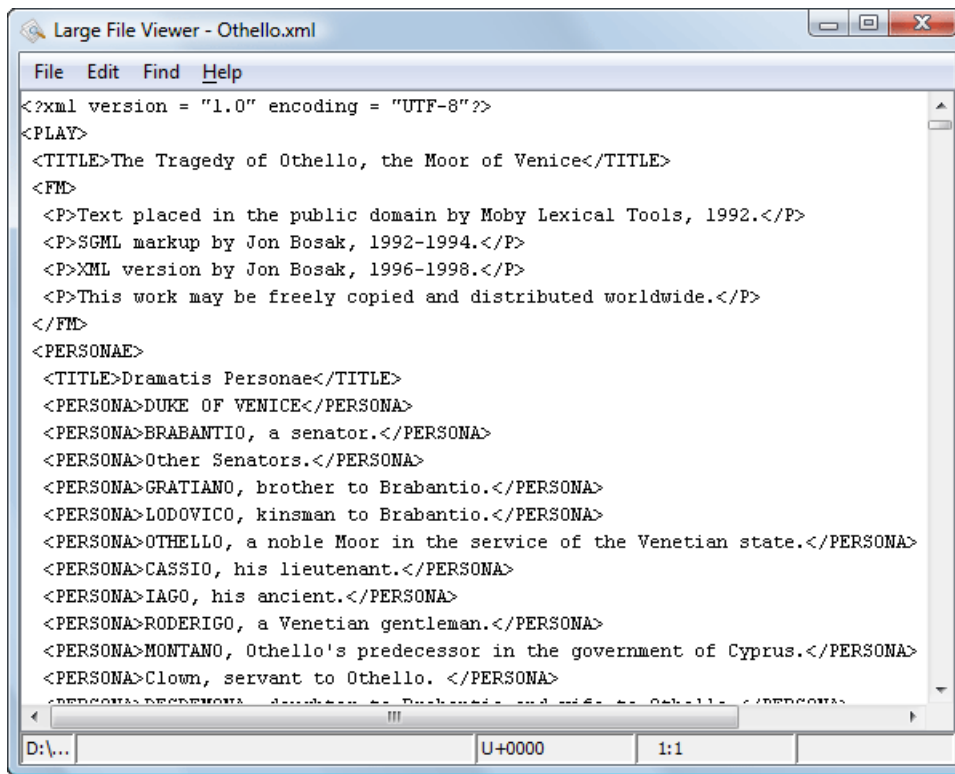


Figure 417: Large File Viewer

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 checked, operations are case-sensitive. Regular Expression - When checked, 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.

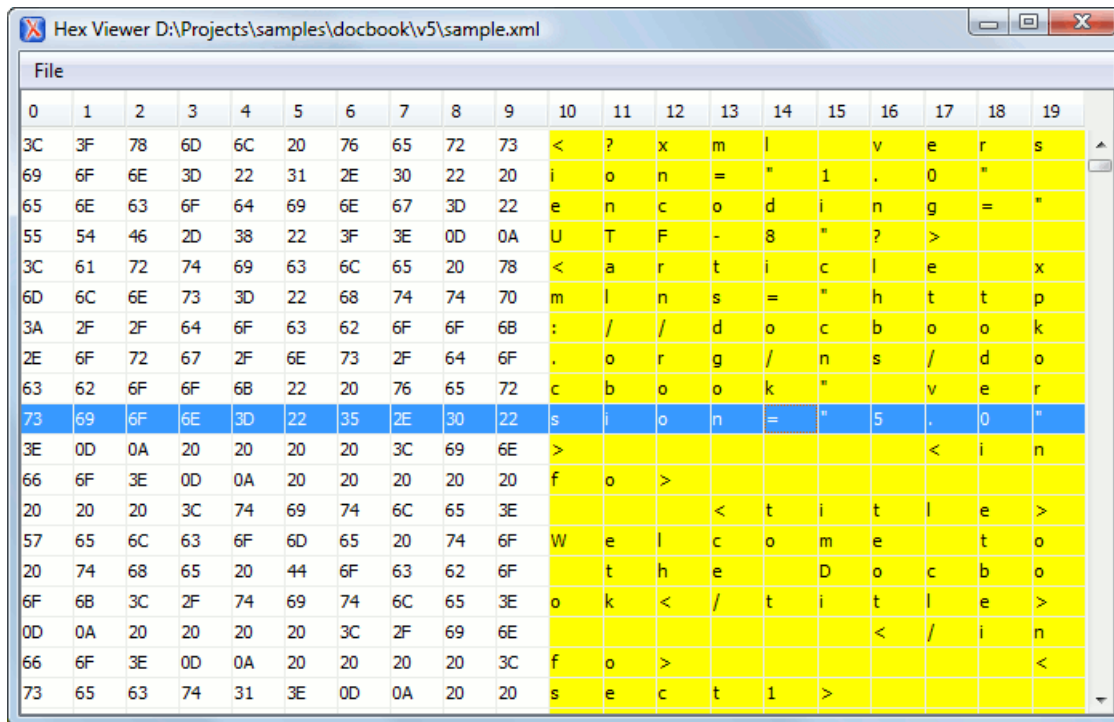


Figure 418: Hex Viewer

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 Developer includes a simple *SVG Viewer* that allows you to work with SVG images.

To open the viewer, select **SVG Viewer** from the **Tools** menu.

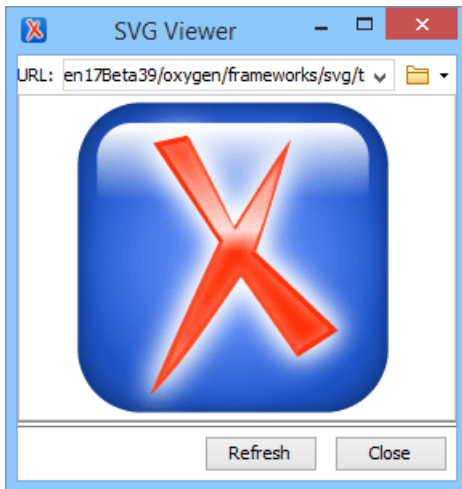


Figure 419: SVG Viewer

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

[SVG 1.2 Rendering Issues](#) on page 1009

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 Developer Tree Editor perspective.
- 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 Developer installation folder (`diffFiles.exe`).

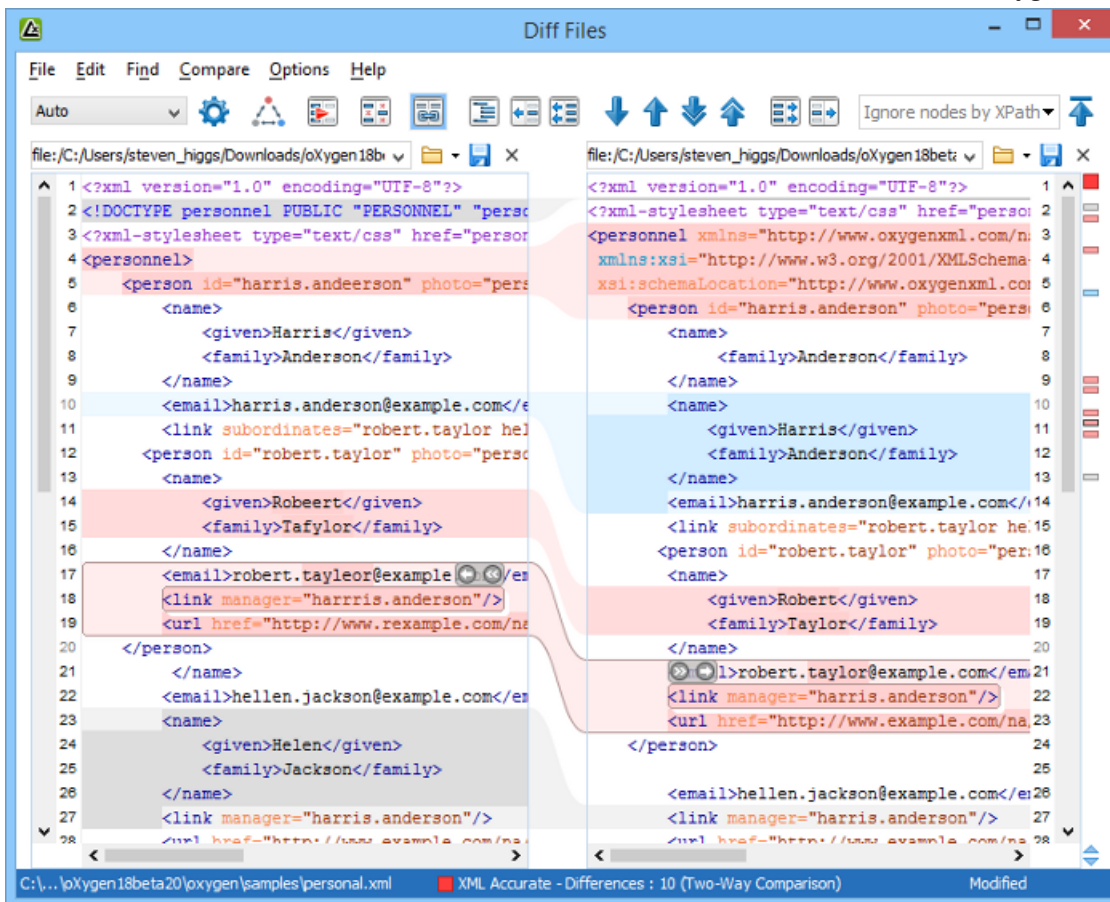


Figure 420: Compare Files Tool

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 perspective 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.

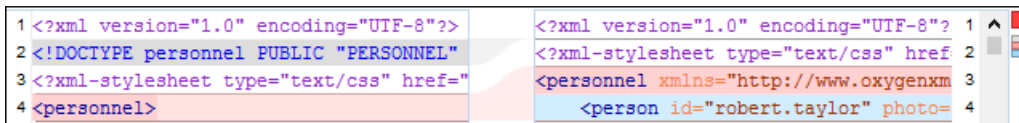


Figure 421: Two-Way Differences

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 Developer 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 Developer 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, a remote 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.

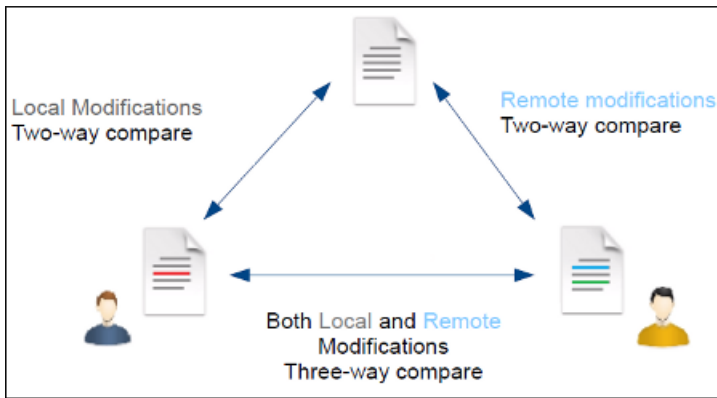


Figure 422: Three-Way Comparison

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 perspective 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 **Ancestor** 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.

7	<given>Robert</given>		<given>Helen</given>	8
8	<family>Taylor</family>		<family>Jackson</family>	9
9	</name>		</name>	10
10	<email>robert.taylor@example		<email>helen.jackson@example	11

Figure 423: 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 Developer 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

Oxygen XML Developer 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 are processed on text nodes using a word level comparison, meaning that it looks for identical words. This second level comparison makes it easier to spot precise differences and you can merge or reject the precise modifications.

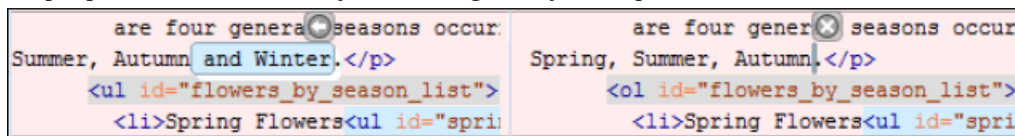


Figure 424: Second Level Diff Comparison



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.

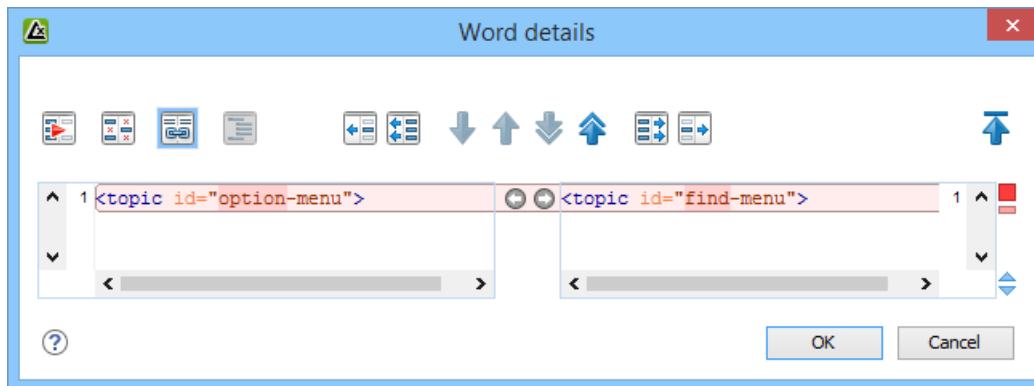


Figure 425: Word Level Comparison

To do a character level comparison, select **Show Character Level details** from the contextual menu or **Compare** menu.

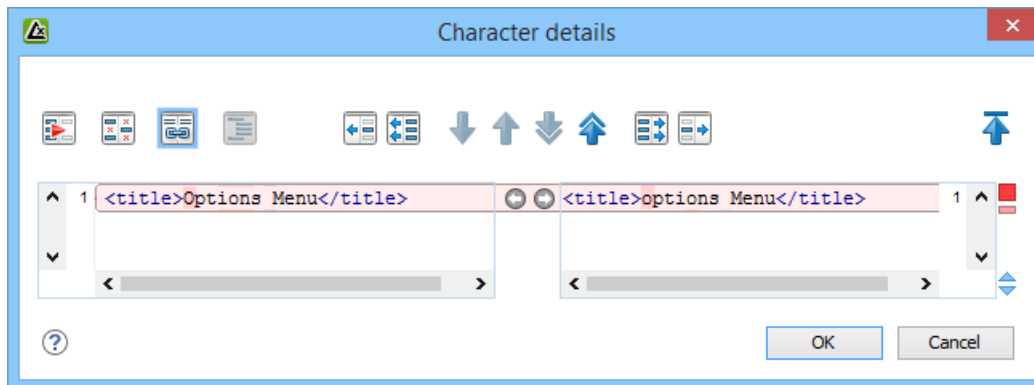


Figure 426: Character Level Comparison

Related information

[Files Comparison Preferences Page](#) on page 132

[Compare Directories](#) on page 530

Compare Files Tool Menus

This section describes the actions that are available in the menus of the **Files Comparison** interface.

File Menu

The **File** menu of the files comparison tool contain the following actions:

- Source** The file that is displayed in the left side of the application window.
- **Source** > **Open** - Browsers for a source file.
 - **Source** > **Open URL** - Opens the URL to be used as a source file. See [Open URL](#) for details.
 - **Source** > **Open File from Archive** - Browsers an archive content for a source file.
 - **Source** > **Save** - Saves the changes made in the source file.
 - **Source** > **Save As** - Displays the **Save As** dialog box that allows you to save the source file with a new name.

- **Source** >  **Close** - Closes the source file.

Target The file that is displayed in the right side of the application window.

- **Target** >  **Open** - Browses for a target file.
- **Target** >  **Open URL** - Opens the URL to be used as a target file. See [Open URL](#) for details.
- **Target** >  **Open File from Archive** - Browses an archive content for a target file.
- **Target** >  **Save** - Saves the changes made in the target file.
- **Target** > **Save As** - Displays the **Save As** dialog box that allows you to save the target file with a new name.
- **Target** >  **Close** - Closes the target file.

Ancestor The base file that will be compared with both files in a [three-way comparison](#).

- **Ancestor** >  **Open** - Browses for a target file.
- **Ancestor** >  **Open URL** - Opens the URL to be used as a target file. See [Open URL](#) for details.
- **Ancestor** >  **Open File from Archive** - Browses an archive content for a target file.

Exit Quits the application.

Edit Menu

The following actions are available in the **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

The **Find** menu includes the following actions:



Find/Replace

Perform *find/replace* operations in the currently focused **Editor**.

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

The following actions are available in the **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 of the application window.

 Next Block of Changes (Ctrl + Period (Command + Period on OS X))

Jumps to the next block of changes. This action is disabled 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 disabled 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 disabled 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 disabled 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 target file in the right side to the source file in the left side.

 Copy All Changes from Left to Right

Copies all changes from the source file in the left side to the target file in the right side.

 Copy Change from Right to Left

Copies the selected difference from the target file in the right side to the source file in the left side.

 Copy Change from Left to Right

Copies the selected difference from the source file in the left side to the target file in the right side.

 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 the same in both files to make it easier to compare the files.

Options Menu

The **Options** menu contains the following actions:

Preferences

Opens the preferences.

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.

- **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 of the application window.

 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

Enables or disables synchronized scrolling so that a selected difference can be seen on both sides of the application window. This option is enabled 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 the same in both files to make it easier to compare the files.

 Copy Change from Right to Left

Copies the selected difference from the target file in the right side to the source file in the left side.

 Copy All Changes from Right to Left

Copies all changes from the target file in the right side to the source file in the left side.

 Next Block of Changes (Ctrl + Period (Command + Period on OS X)**)**

Jumps to the next block of changes. This action is disabled 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 disabled 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 disabled 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 disabled 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 source file in the left side to the target file in the right side.

 Copy Change from Left to Right

Copies the selected difference from the source file in the left side to the target file in the right side.

Ignore Nodes by XPath

You can use this text field to enter an XPath 2.0 expression to ignore certain nodes from the comparison. This field is only available when comparing XML documents using the **XML Fast** or **XML Accurate** algorithms. If an XPath 2.0 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.

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 Developer installation folder (diffDirs.exe).

The directories comparison results are presented as a tree of files and directories. The directories and folders 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 Developer starts a *file comparison* between the two files, using the **Compare Files** tool.

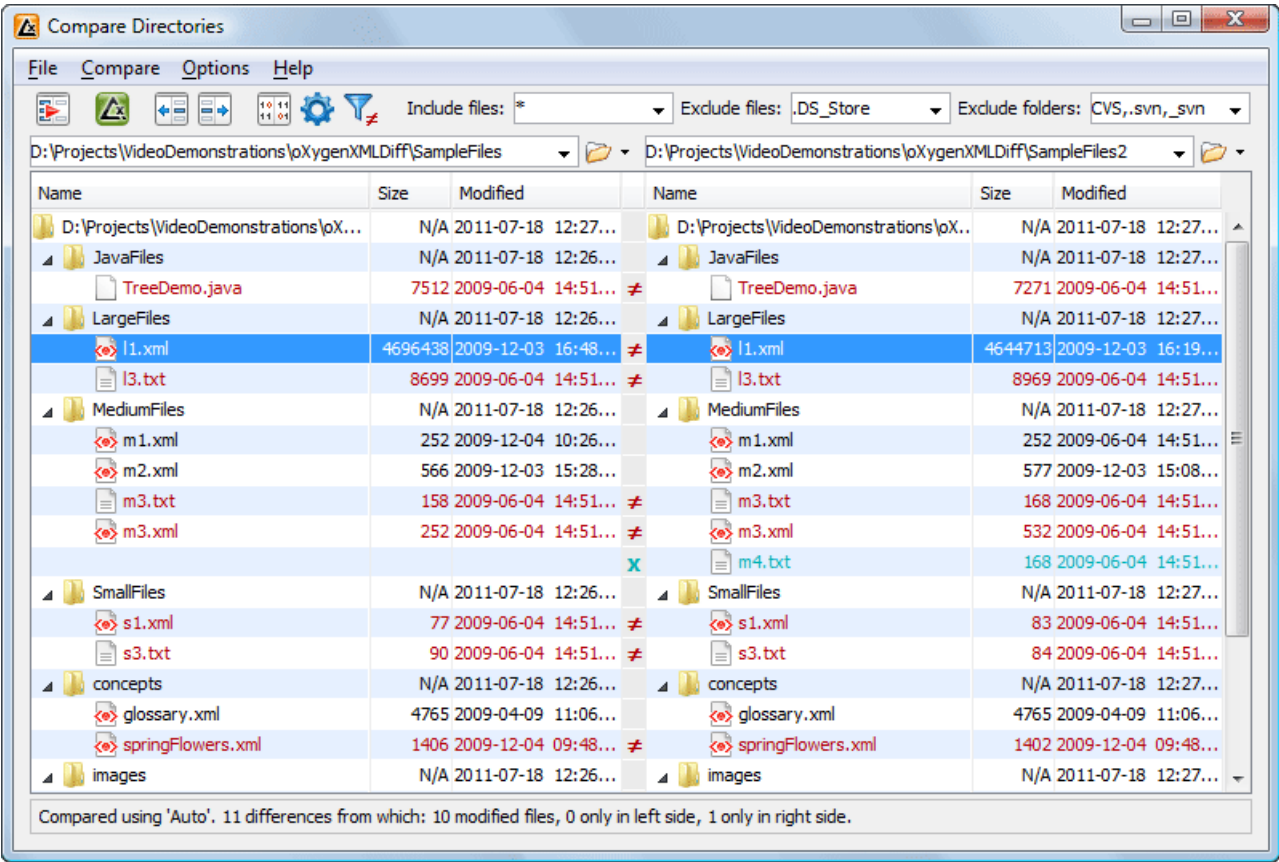


Figure 428: Compare Directories Tool

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 Developer treat supported archives as directories, enable 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.

Related information

[Compare Files](#) on page 519

Compare Directories Tool Menus

This section describes the actions that are available in the menus of the **Directories Comparison** interface.

Compare Menu

This menu contains the following 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).

Options Menu

The **Options** menu contains the following actions:

Preferences	Opens the preferences.

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

The **Help** menu contains the following actions:

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 enabled, when you select Help or press F1 while hovering over any part of the interface, Oxygen XML Developer attempts to open the help documentation in online mode. If this option is disabled 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 <code>com.oxygenxml.report.problems.url</code> system property. The report is sent in XML format through the <code>report</code> parameter of the POST HTTP method.
Support Center	Opens the Oxygen XML Developer Support Center web page in a browser.
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 Developer. 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 XML Developer support team.

Compare Toolbar of the Compare Directories Tool

The toolbar contains the following actions:



Figure 429: Compare toolbar

 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.

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*.

SVN Client

Describes how to use the 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** - 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.
- **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 <i>version control</i> only if the selected folder is under <i>version control</i> .
		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 <i>version control</i> only if the selected folder is under <i>version control</i> .
		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 <code>svn:externals</code> 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.
	 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.

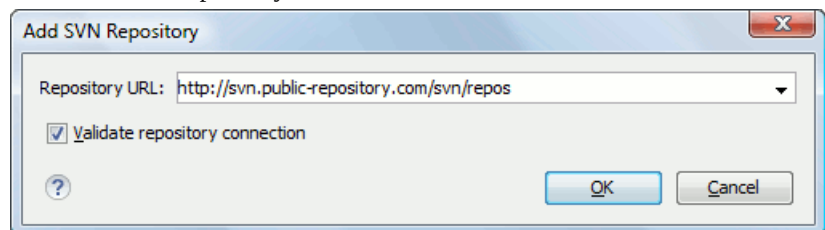


Figure 430: Add SVN Repository Dialog Box

If the **Validate repository connection** option is selected, the URL connection is validated before being added to the **Repositories** view.

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 enabled 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 <i>version control</i> , 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 <i>missing</i>), 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 <i>missing</i>), 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 of the application window.

 **Next Block of Changes (Ctrl + Period (Command + Period on OS X))**

Jumps to the next block of changes. This action is disabled 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 disabled 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 disabled 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 disabled 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 target file in the right side to the source file in the left side.
	 Copy Change from Right to Left	Copies the selected difference from the target file in the right side to the source file in the left side.
	 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 the same in both files to make it easier to compare the files.
	 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 <i>the annotations for a file and displays them in the Annotations view</i> , 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 <i>Revision Graph</i> . This operation is enabled for any resource selected into the Repositories view or Working Copy view.
Tools Menu	Share project	Allows you to <i>share a new project</i> 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 <i>Creating a Branch / Tag</i> .
	 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 <i>Merging</i> .
	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 <i>external</i> items. To read more about this action, see the <i>Switching the Repository Location</i> 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.
Window Menu	Reset Options	Resets all your options to the default ones.
	Reset Authentication	Resets the Subversion authentication information.
	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.
Help Menu	Reset Layout	Resets all the views to their default position.
	Help (F1)	Opens the Help dialog box.
	Use online help (Enabled by default)	If this option is enabled, when you select Help or press F1 while hovering over any part of the interface, Oxygen XML Developer attempts to open the help documentation in online mode. If this option is disabled 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.
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 Developer. 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 XML Developer support team.
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 than 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.

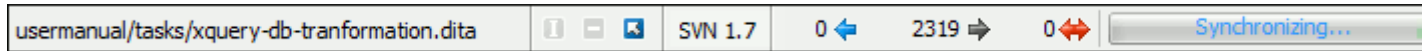


Figure 431: Status bar

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 Developer 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 Developer 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 checked, 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 Developer for the change to take effect.



Tip: The *FILE* protocol is recommended if the SVN repository and Oxygen XML Developer 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.

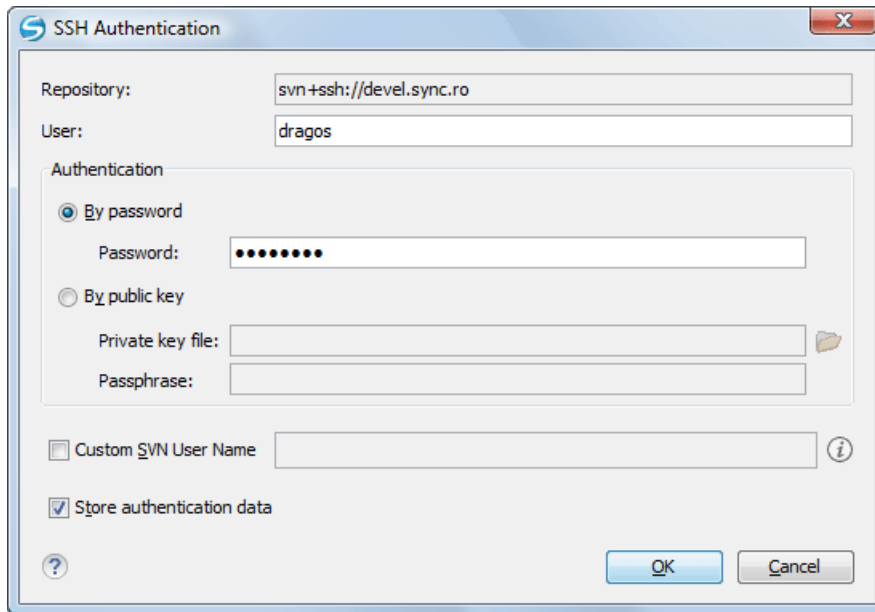


Figure 432: SSH Authentication Dialog Box

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.

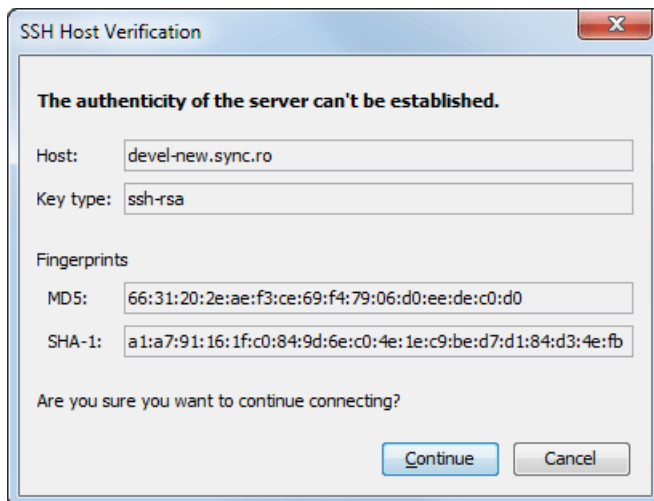


Figure 433: SSH server name and key fingerprint

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.

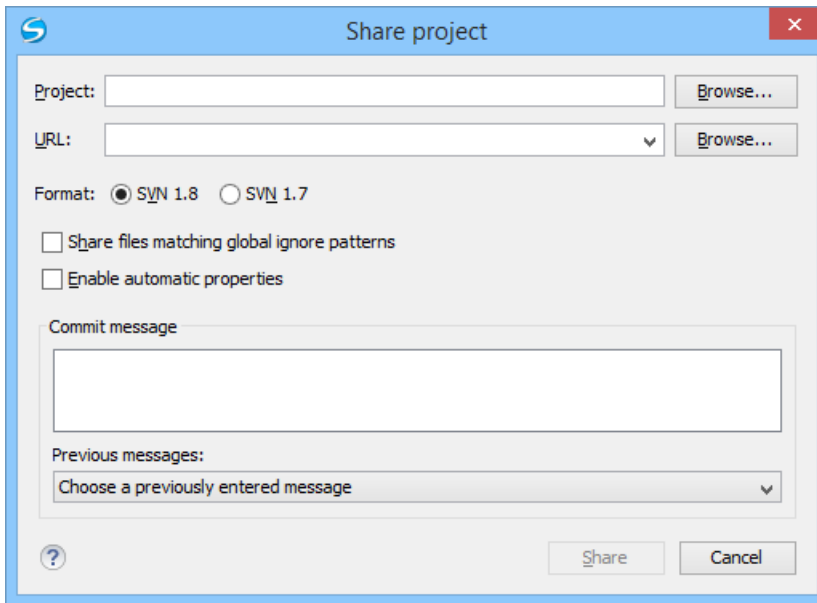


Figure 434: Share Project Dialog Box

The following options can be configured in the **Share project** dialog box:

Project	<i>The location of the project folder</i> 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	<i>The new location of the project</i> (inside the repository) that will be used to access it.  Note: <i>Peg revisions</i> 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 enabled, the file names that match the patterns defined in either of the following locations are also imported into the repository: • The <code>global-ignore</code> property in <i>the SVN configuration file</i>. • The <i>File name ignore patterns</i> 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 <code>enable-auto-props</code> runtime configuration directive, defined in <i>the SVN configuration file</i>.  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 <code>auto-props</code> section). Based on the value of the <code>enable-auto-props</code> 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.

Name	Date	Revision	Author	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	Executable ...
gradle-wrapper.properties	May 4, 2011	7623	alex	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	File
jsvn.bat	May 10, 2011	7630	alex	Windows B...
jsvnsetup.openvms	May 4, 2011	7618	alex	OPENVMS File
build.gradle	May 4, 2011	7618	alex	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-ogsi	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 435: 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.

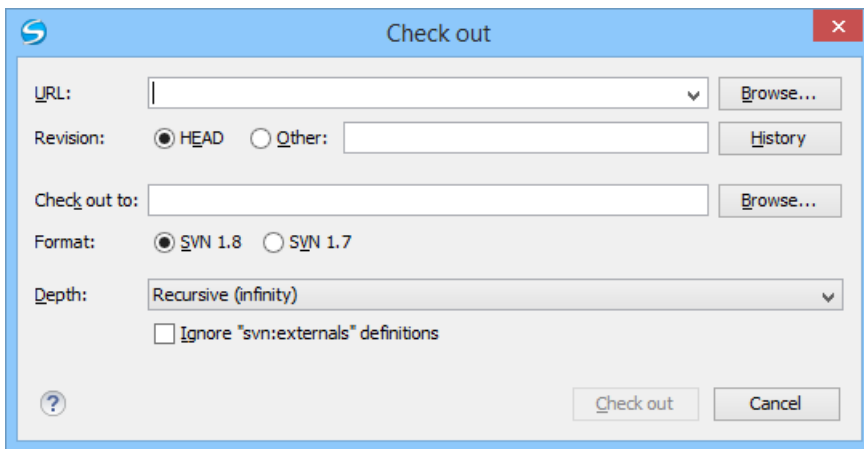


Figure 436: Check Out Dialog Box

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 enabled, 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.

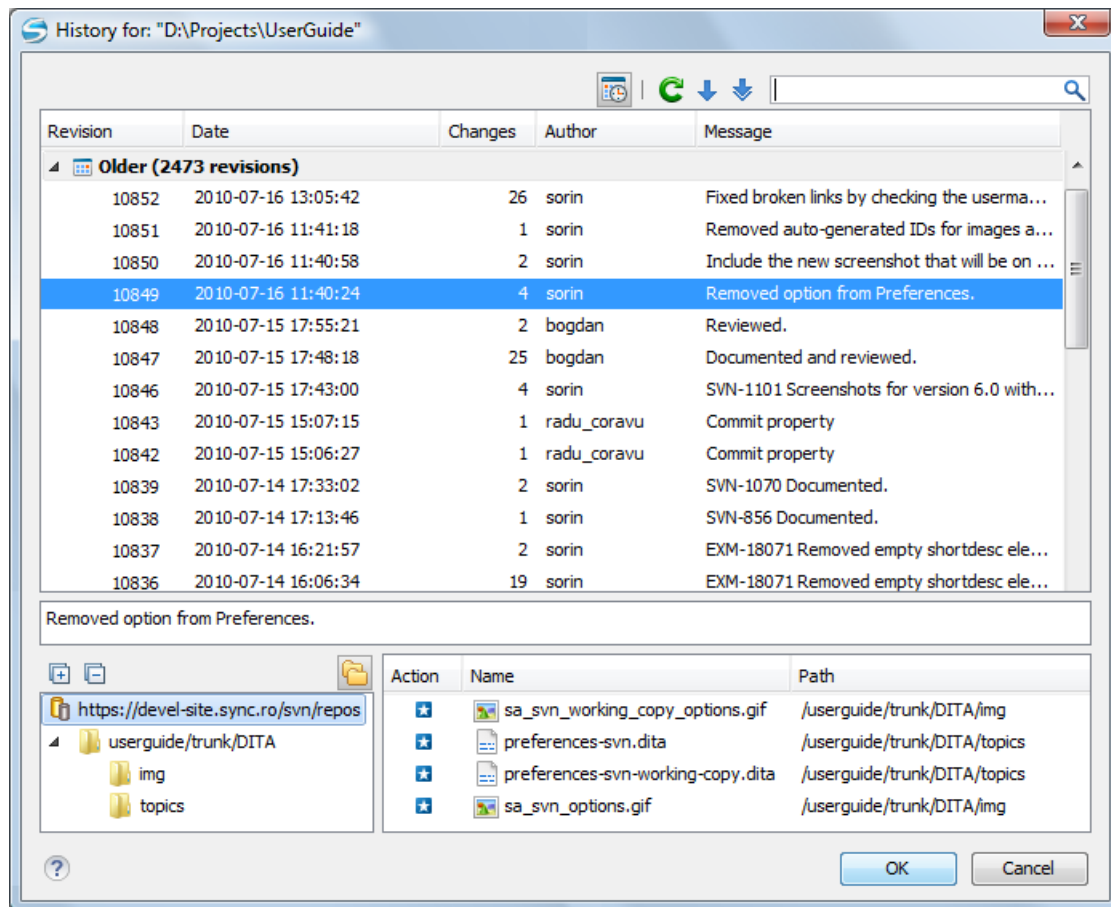


Figure 437: History Dialog Box

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.

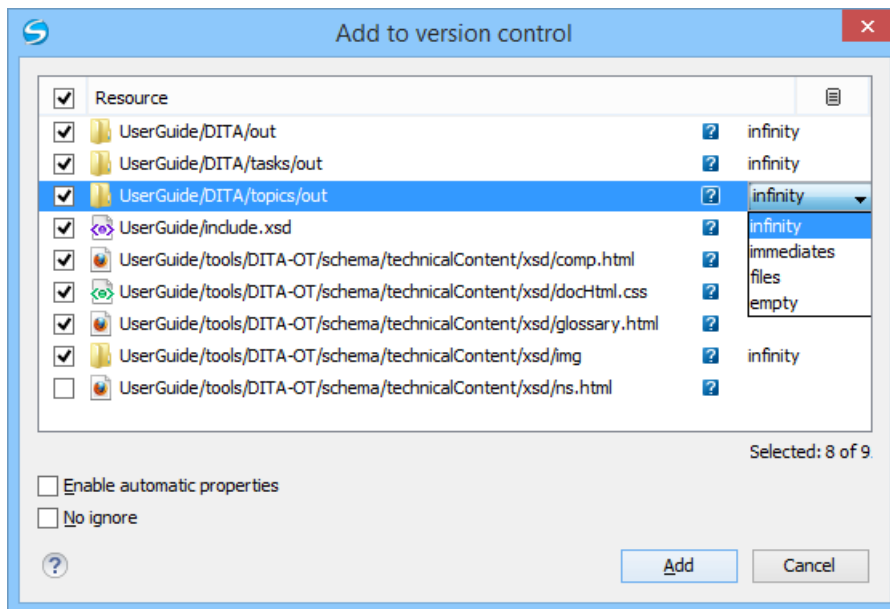


Figure 438: Add to Version Control Dialog Box

Note: Ignored (I) 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 enabled, the value is based on the listing mode of the items in the working copy view. When they are disabled, 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 enable 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 Developer 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 losing 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 enabled 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 Developer 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.

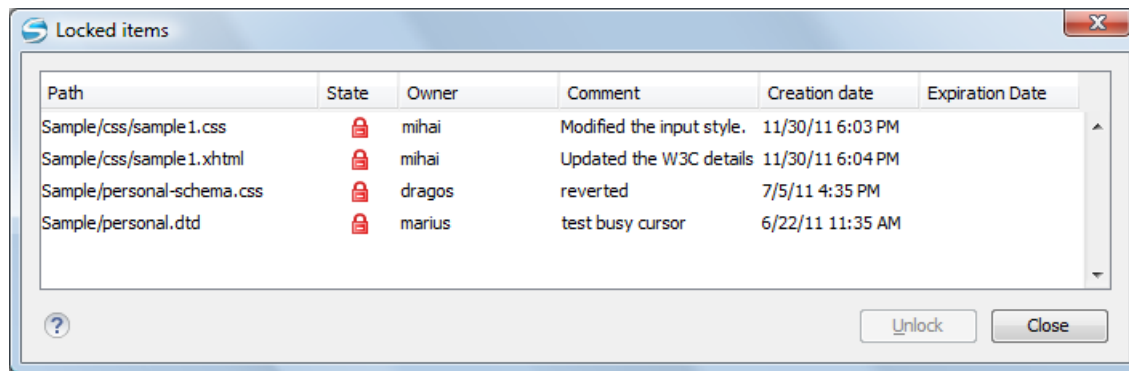


Figure 439: Locked Items Dialog Box

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 971

[Repository Locks](#) on page 964

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 enable 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](#).

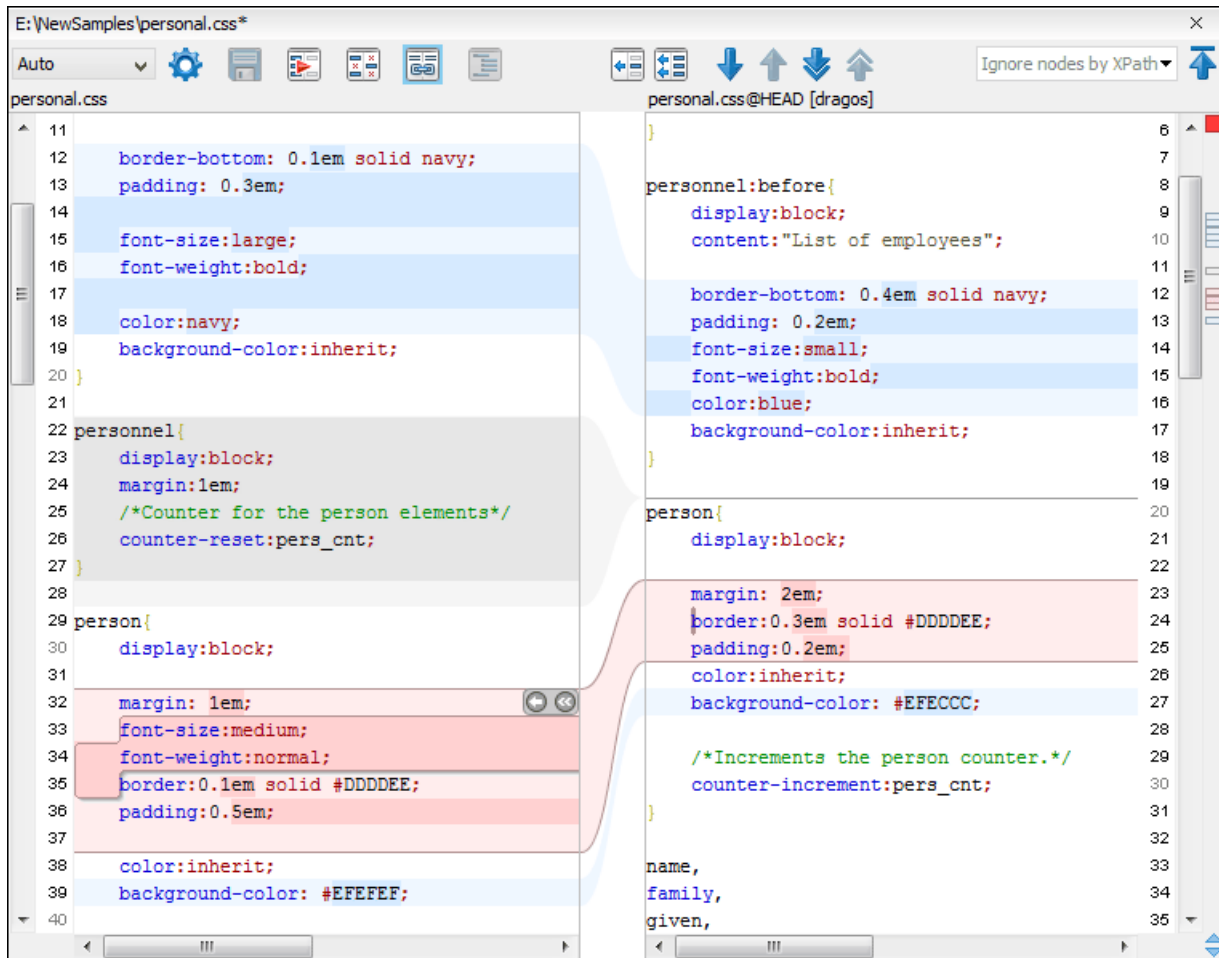


Figure 440: Compare View

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 disregarding the changes in the number of white spaces between words or lines there is available an option in the [SVN Preferences](#) which 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* and is enabled only 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 option Show warning dialog when edit conflicts is enabled* 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 of the application window.



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

Synchronizes scrolling so that a selected difference can be seen on both sides of the application window. This action enables/disables the previously described behavior.



Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on OS X))



Note: When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys the same in both files to make it easier to compare the files.

**Copy Change from Right to Left**

Copies the selected difference from the target file in the right side to the source file in the left side.

**Copy All Changes from Right to Left**

Copies all changes from the target file in the right side to the source file in the left side.

**Next Block of Changes (Ctrl + Period (Command + Period on OS X))**

Jumps to the next block of changes. This action is disabled 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 disabled 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 disabled 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 disabled 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.

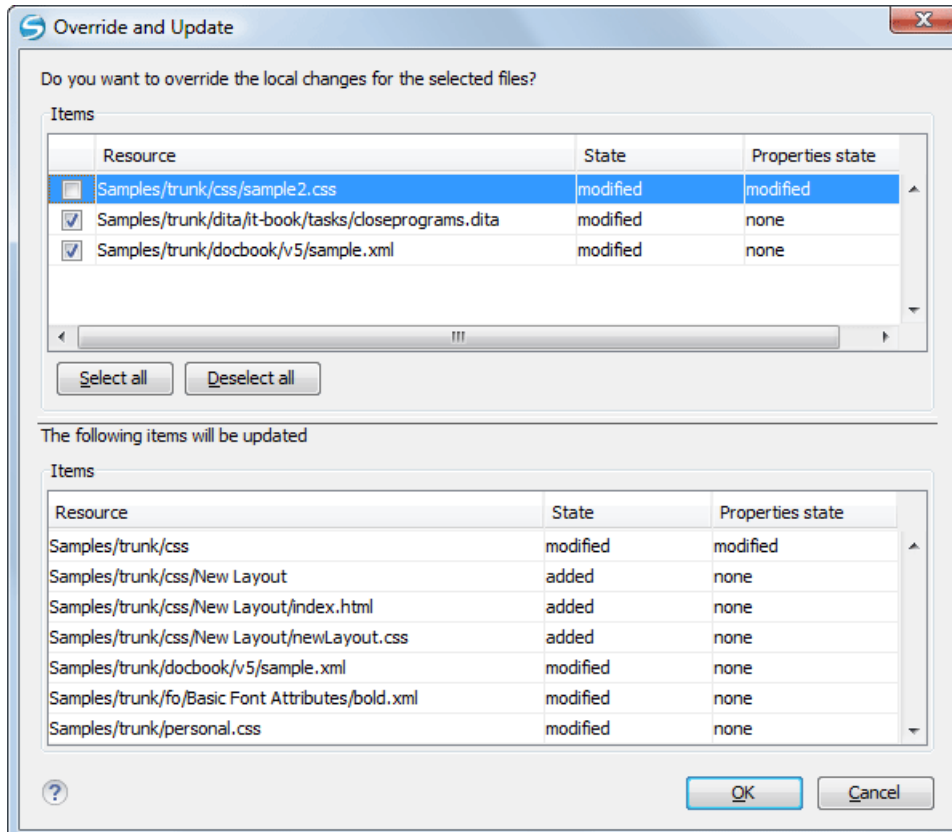


Figure 441: Override and Update Dialog Box

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:

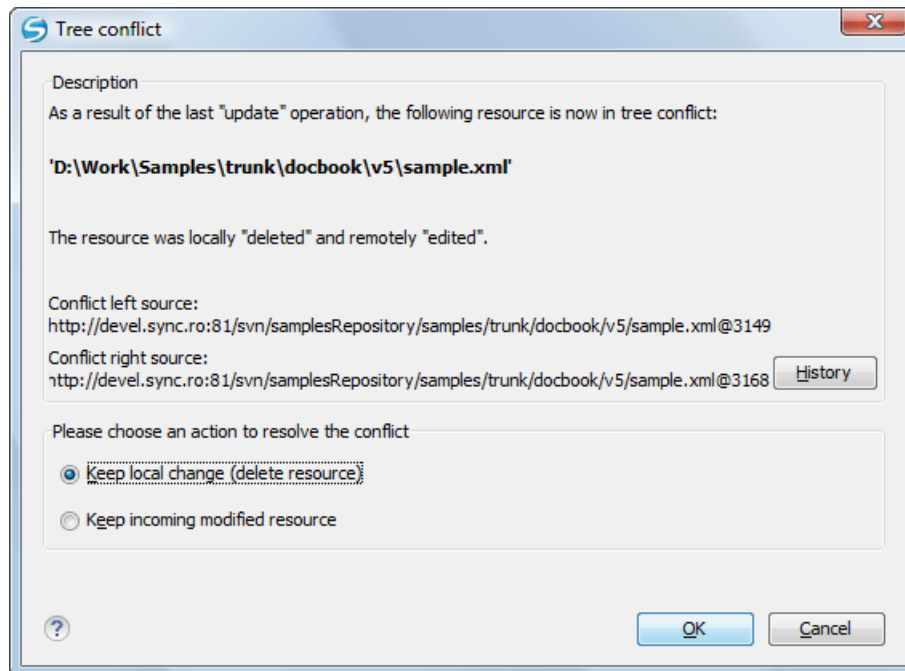


Figure 442: Resolve a tree conflict

- **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 Developer 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.

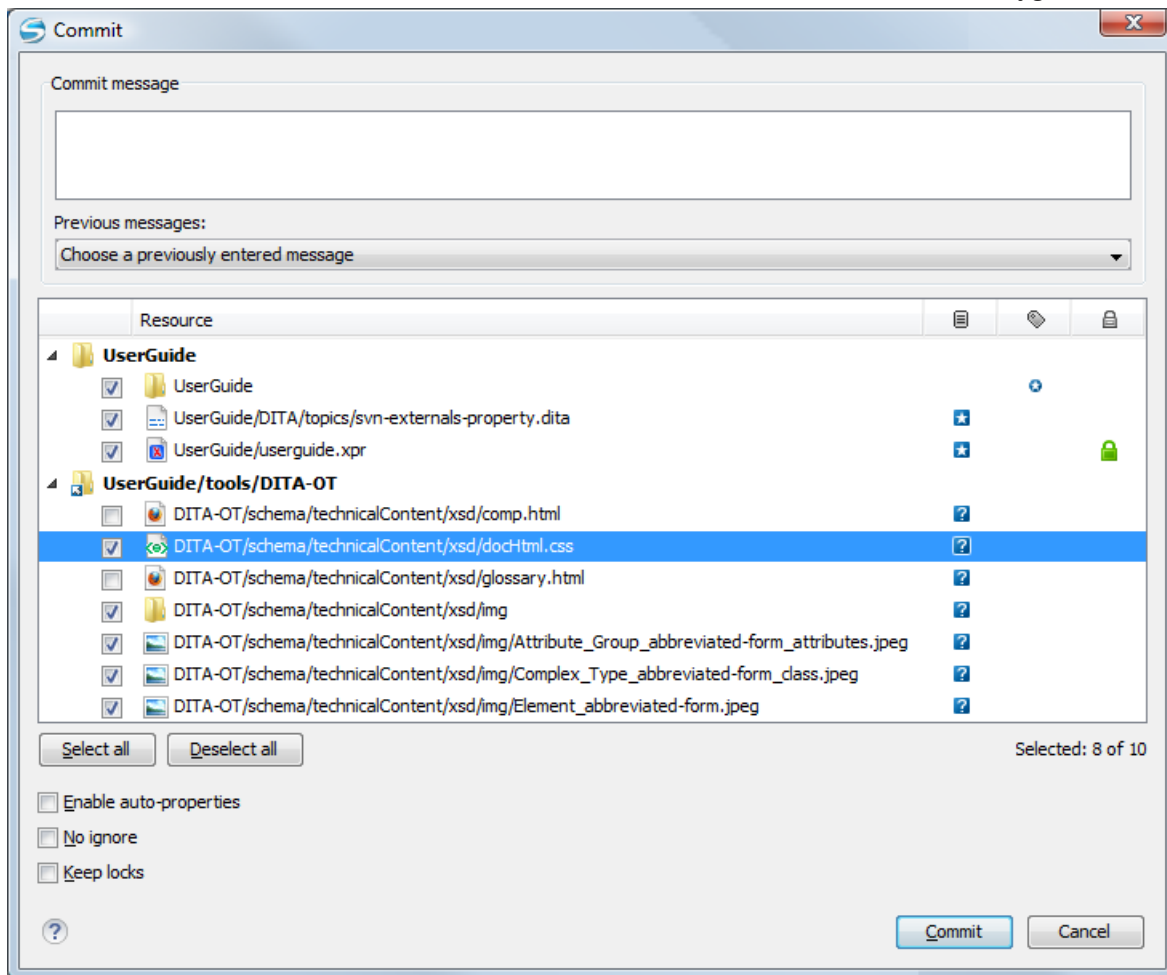


Figure 443: Commit dialog box

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, uncheck 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** is disabled, 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 from the **Window > Show View** menu.

Related information

[History View](#) on page 978

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 checking 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.

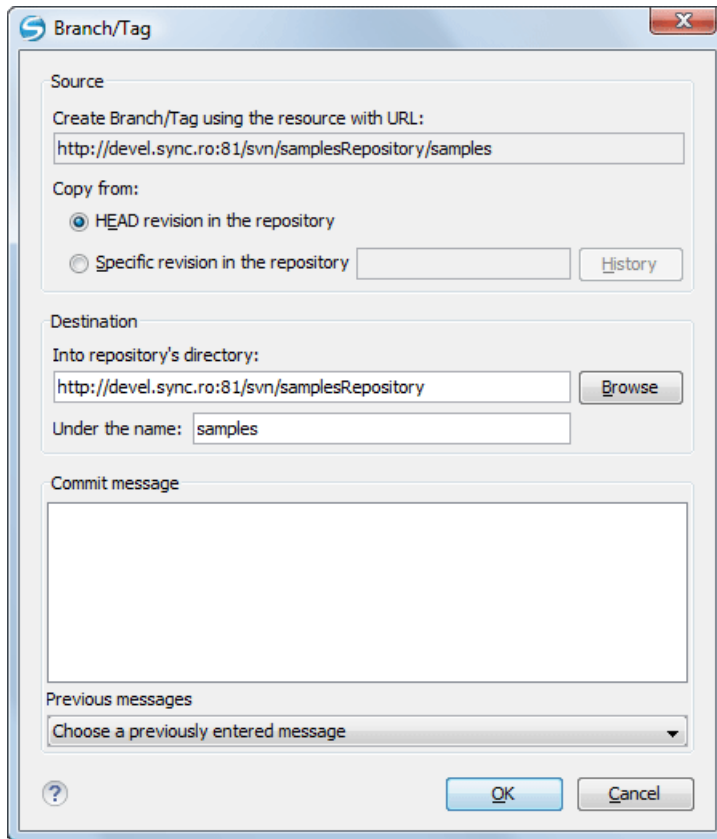


Figure 444: Branch/Tag Dialog Box

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.

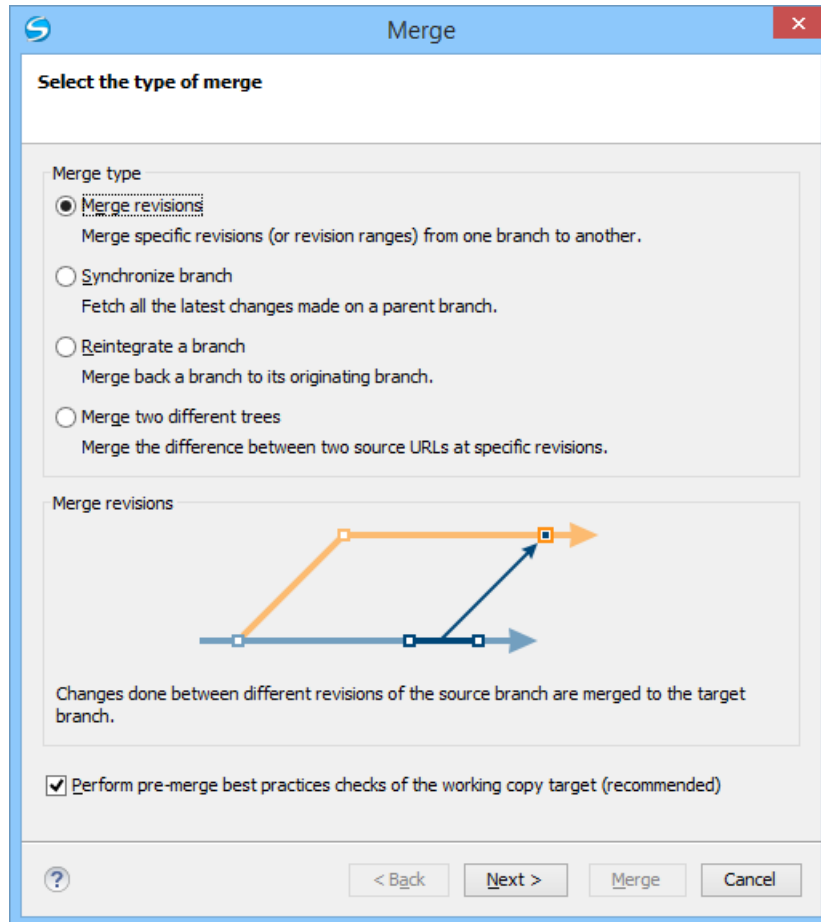


Figure 445: Merge Wizard

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 option:

- **Perform pre-merge best practices checks of the working copy target** - When enabled, 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 enabling 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.

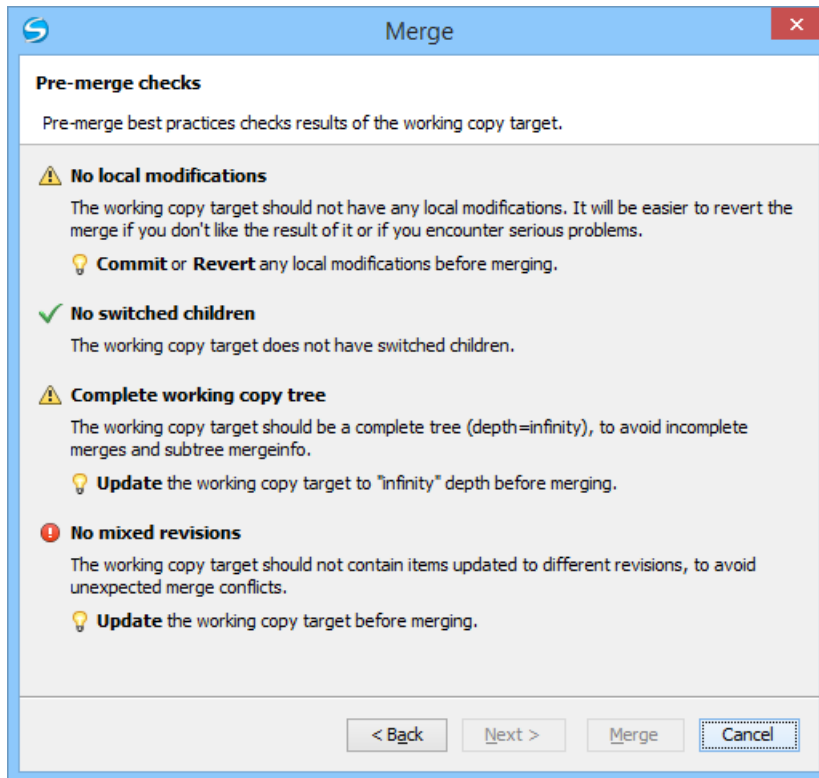


Figure 446: Pre-Merge Checks Wizard Page

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 enable 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 was enabled, *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 enabled 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).

- 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 enabled and you selected **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 enable 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 was enabled, [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 depth *infinity*).
 - 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 disabled 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 enable 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 was enabled, [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 enable 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 was enabled, [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 enabled 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 enabled and you selected **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:

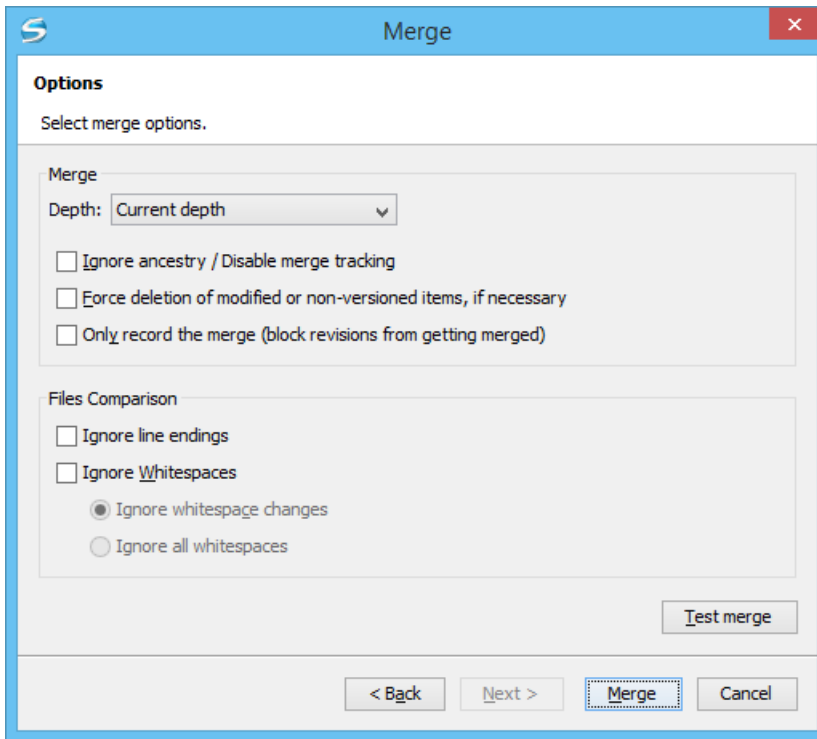


Figure 447: Merge Wizard - Advanced Options

- **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 disabled, 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, disabling the option replaces one of the files with the other one, and enabling 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 disabled, 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 disabled. 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 enable 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.

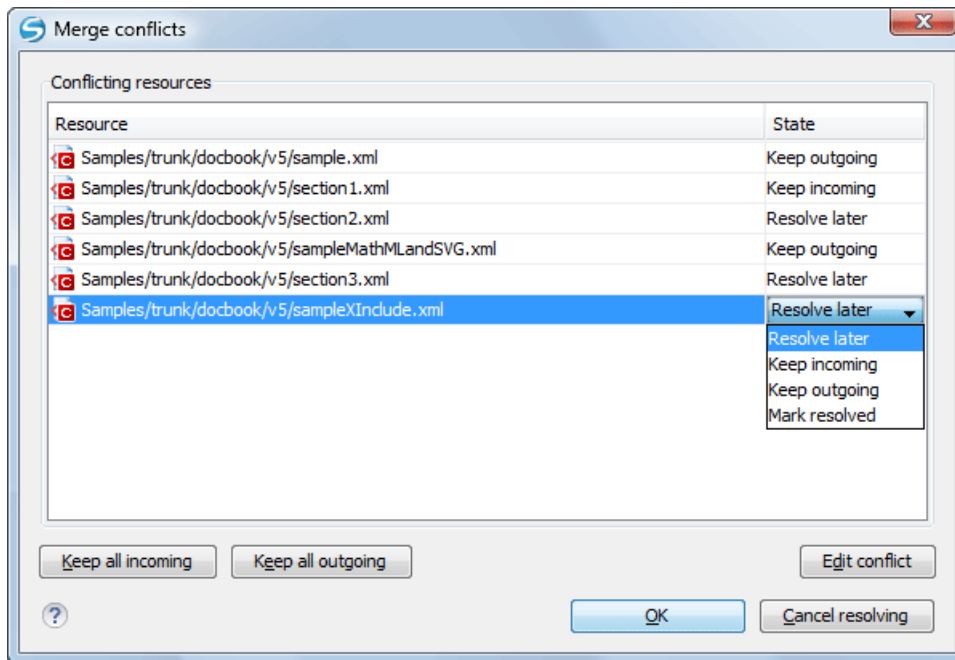


Figure 448: Merge Conflicts Dialog Box

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).

Additional Notes About the Merge Operation

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 enabled 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.

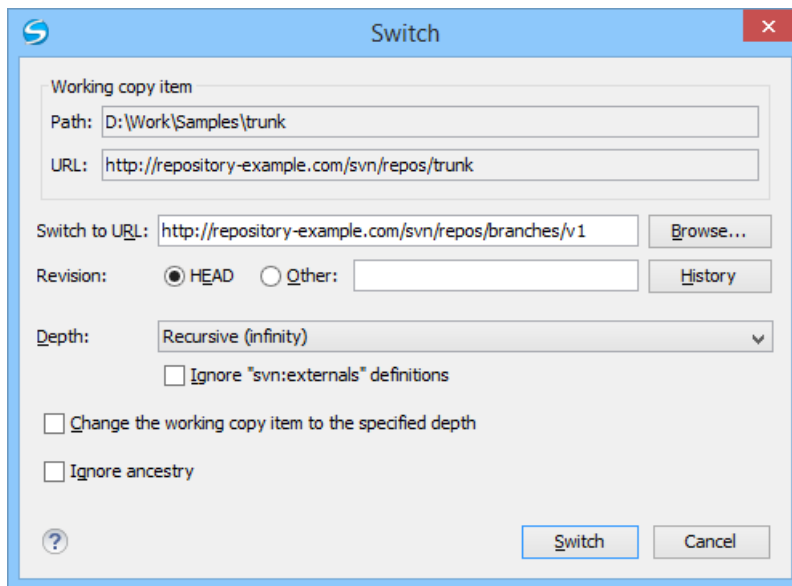


Figure 449: Switch Dialog Box

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 enabled, 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 disabled, the SVN system does not allow you to switch to a location that does not share a common ancestry with the current location. If enabled, 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.

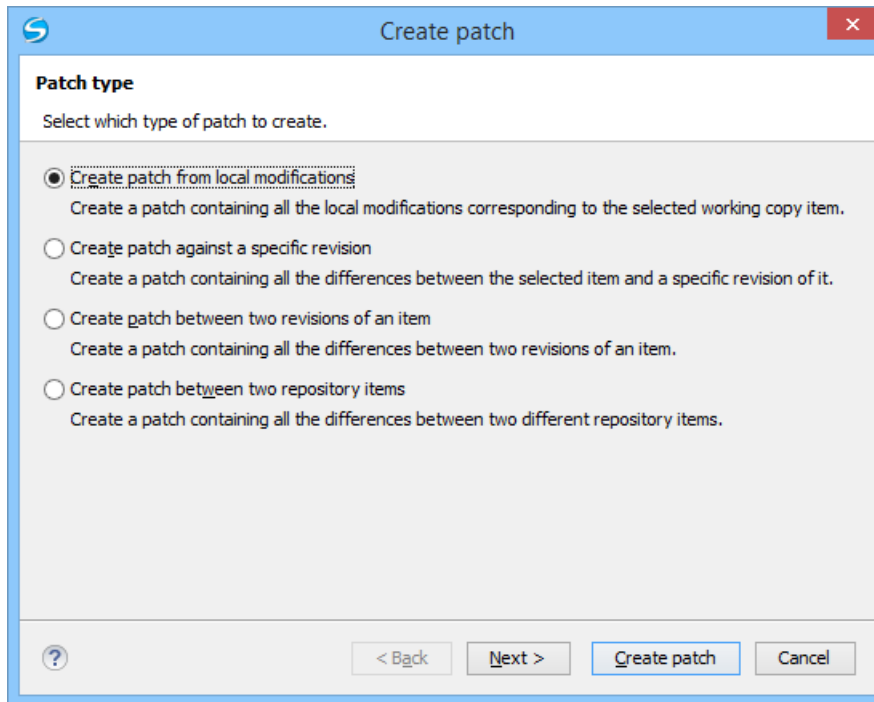


Figure 450: Create Patch Wizard - Local Items

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.

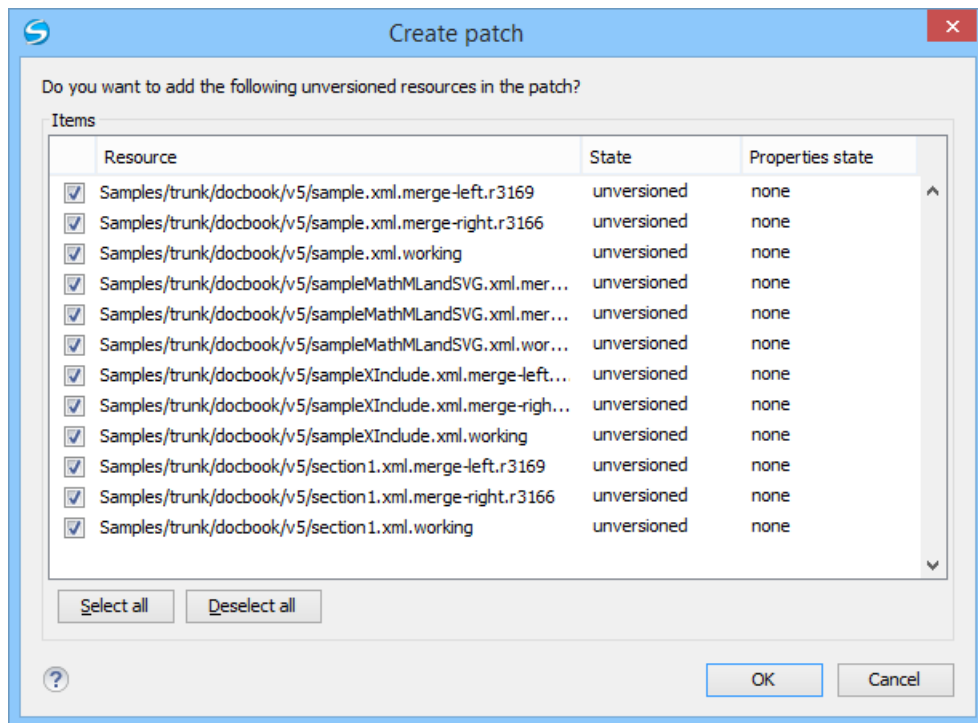


Figure 451: Create Patch Dialog Box - Add Unversioned Resources

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:

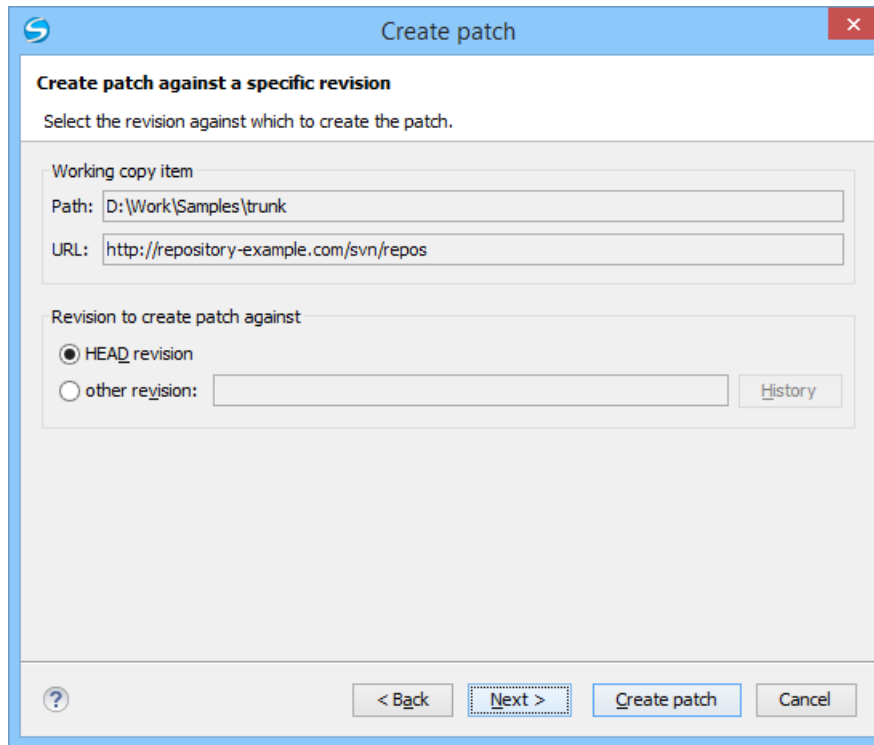


Figure 452: Create Patch Wizard - Step 2

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:

The screenshot shows a dialog box titled 'Create Patch Wizard - Step 2'. It is divided into two main sections: 'From' and 'To'.
 In the 'From' section, there are two radio buttons: 'HEAD revision' (unselected) and 'other revision' (selected). Next to 'other revision' is a text input field containing the number '27436'. To the right of this field is a button labeled 'History'.
 In the 'To' section, there are also two radio buttons: 'HEAD revision' (selected) and 'other revision' (unselected). Next to 'other revision' is an empty text input field. To the right of this field is another button labeled 'History'.

Figure 453: 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:

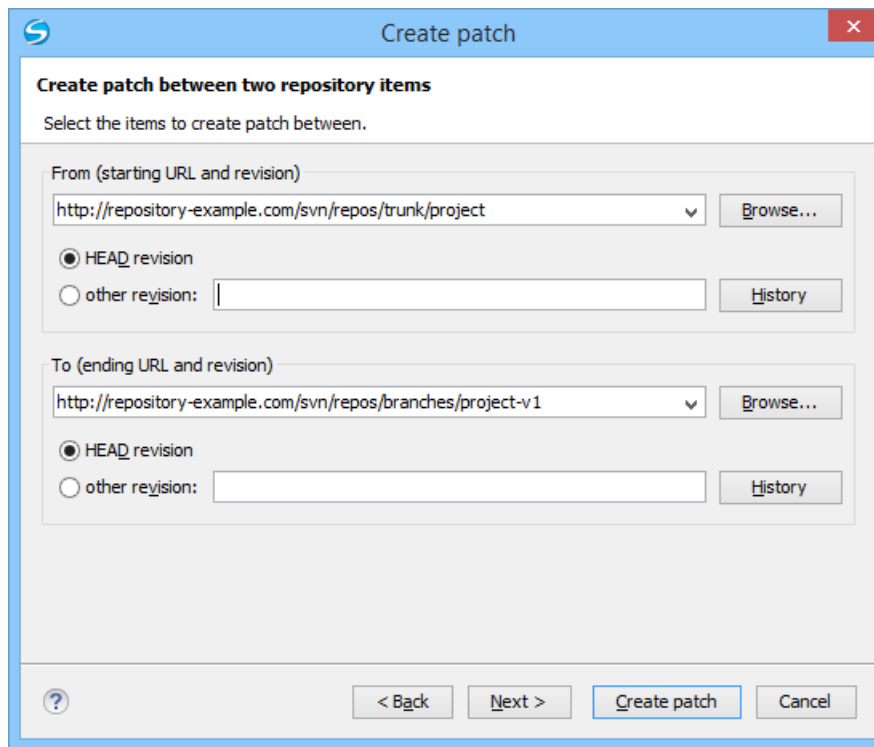


Figure 454: Create Patch Wizard - Step 2

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.

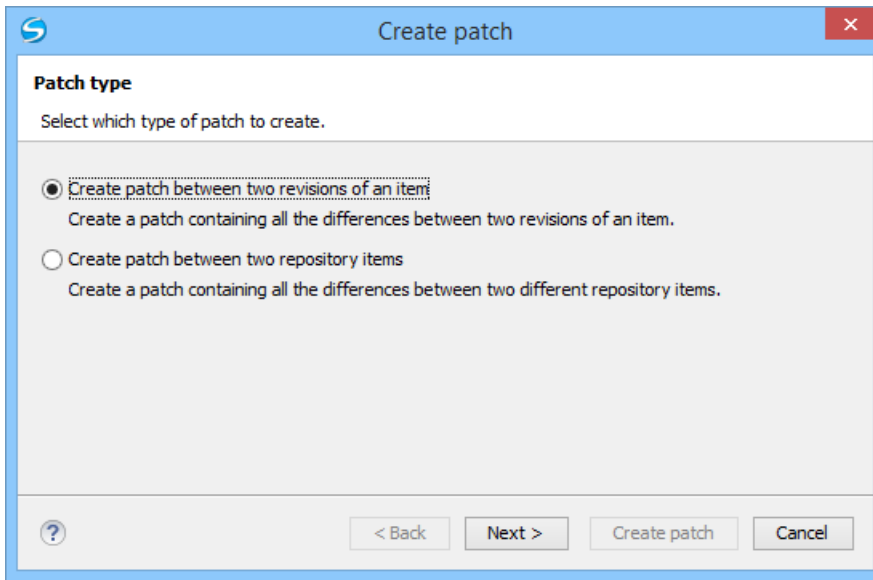


Figure 455: Create Patch Wizard - Remote Items

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 456: 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.

- 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.

- 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:

- Go to menu **Tools > Create patch**.

This opens the **Create patch** wizard.

- Select the **Create patch between two repository items** option in the dialog box.

- Press the **Next** button.

The second step of the wizard is opened:

Figure 457: Create Patch Wizard - Step 2

- 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 458: 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 enabled, the patch file does not include the content of the <i>added</i> items. This option corresponds to the <code>--no-diff-added</code> option of the <code>svn diff</code> command.
Ignore content of delete files	When enabled, the patch file does not include the content of the <i>deleted</i> items. This option corresponds to the <code>--no-diff-deleted</code> option of the <code>svn diff</code> command.
Treat copied files as newly added	When enabled, copied items are treated as new, rather than comparing the items with their sources. This option corresponds to the <code>--show-copies-as-adds</code> option of the <code>svn diff</code> command.
Include files having a binary MIME type	When enabled, the application is forced to compare items that are considered binary file types. This option corresponds to the <code>--force</code> option of the <code>svn diff</code> command.
Ignore properties	When enabled, differences in the properties of items are ignored. This option corresponds to the <code>--ignore-properties</code> option of the <code>svn diff</code> command.
Properties only	When enabled, only differences in the properties of the items are included in the patch file (file content is ignored). This option corresponds to the <code>--properties-only</code> option of the <code>svn diff</code> command.
	 Note: The Ignore properties and Properties only options are mutually exclusive.
Notice ancestry	If enabled, the SVN common ancestry is taken into consideration when calculating the differences. This option corresponds to the <code>--notice-ancestry</code> option of the <code>svn diff</code> command.

Files Comparison Section

Ignore line endings	If enabled, the differences in line endings are ignored when the patch is generated. This option corresponds to the <code>--ignore-eol-style</code> option of the <code>svn diff</code> command.
Ignore whitespaces	If enabled, it allows you to specify how the whitespace changes should be handled. When enabled, you can then choose between two options: • Ignore whitespace changes (<code>--ignore-space-change</code>) - 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 (<code>--ignore-all-space</code>) - 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 enabled, the patch is generated using the <i>Git</i> format. This option corresponds to the <code>--git</code> option of the <code>svn diff</code> 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.

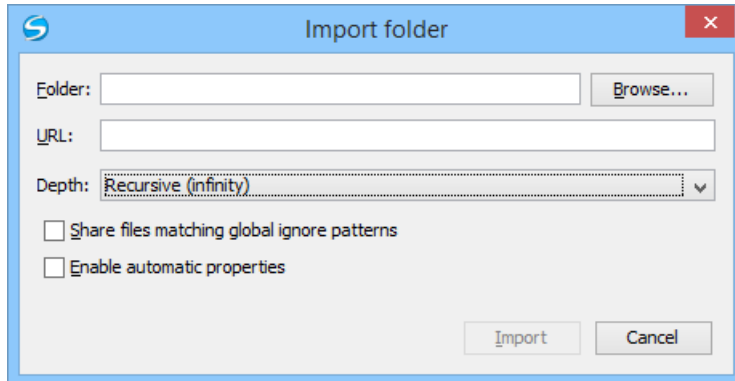


Figure 459: Import Folder Dialog Box

You can configure the following options:

Folder	Specify <i>the local folder</i> 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 <i>the repository location</i> that will be used to access the folder to be imported.  Note: <i>Peg revisions</i> 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) - Import 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 enabled, the file names that match the patterns defined in either of the following locations are also imported into the repository: • The <code>global-ignores</code> property in <i>the SVN configuration file</i>. • The <i>File name ignore patterns option</i> 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 <code>enable-auto-props</code> runtime configuration directive, defined in <i>the SVN configuration file</i>.  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 <code>auto-props</code> 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:

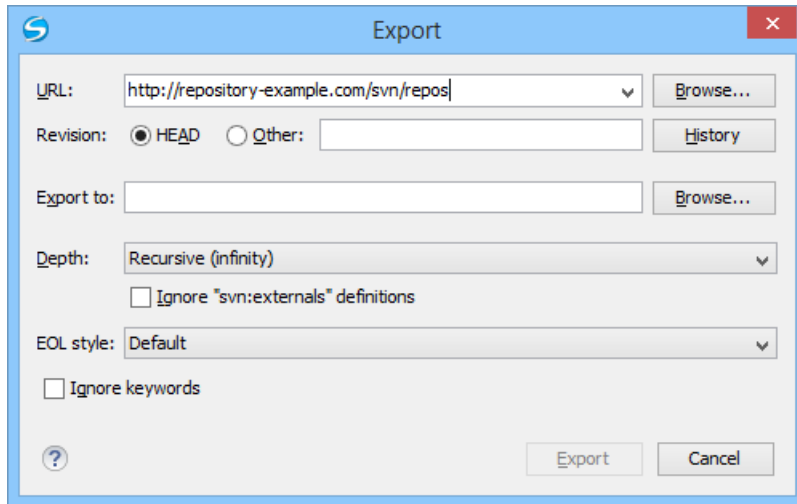


Figure 460: Export from Repository Dialog Box

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) - Export 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 enabled, 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 enabled, 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.

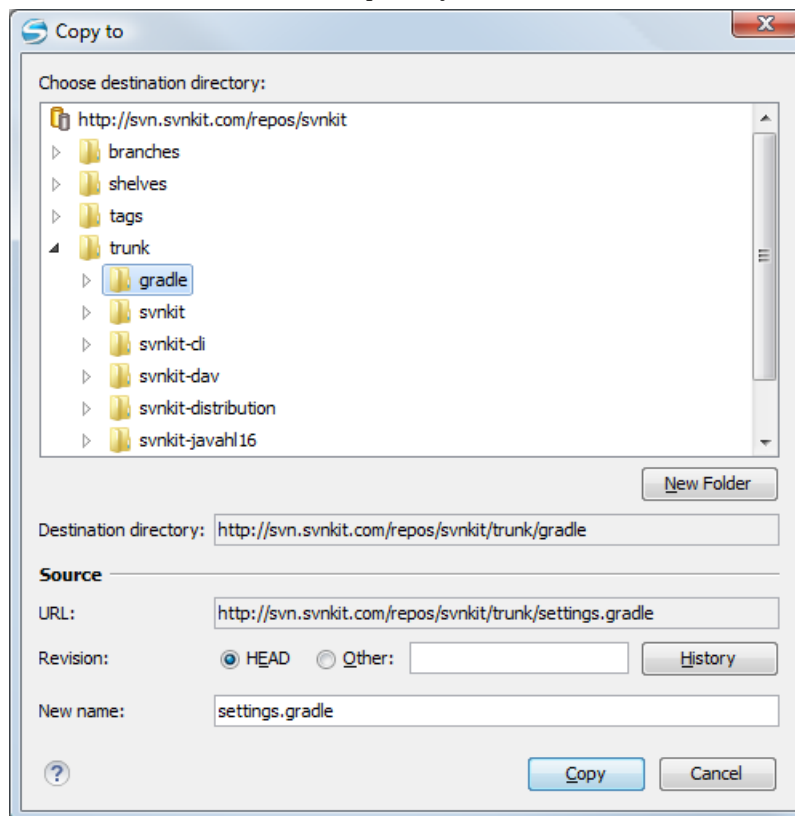


Figure 461: Copy/Move Items in Repository

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 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](#)

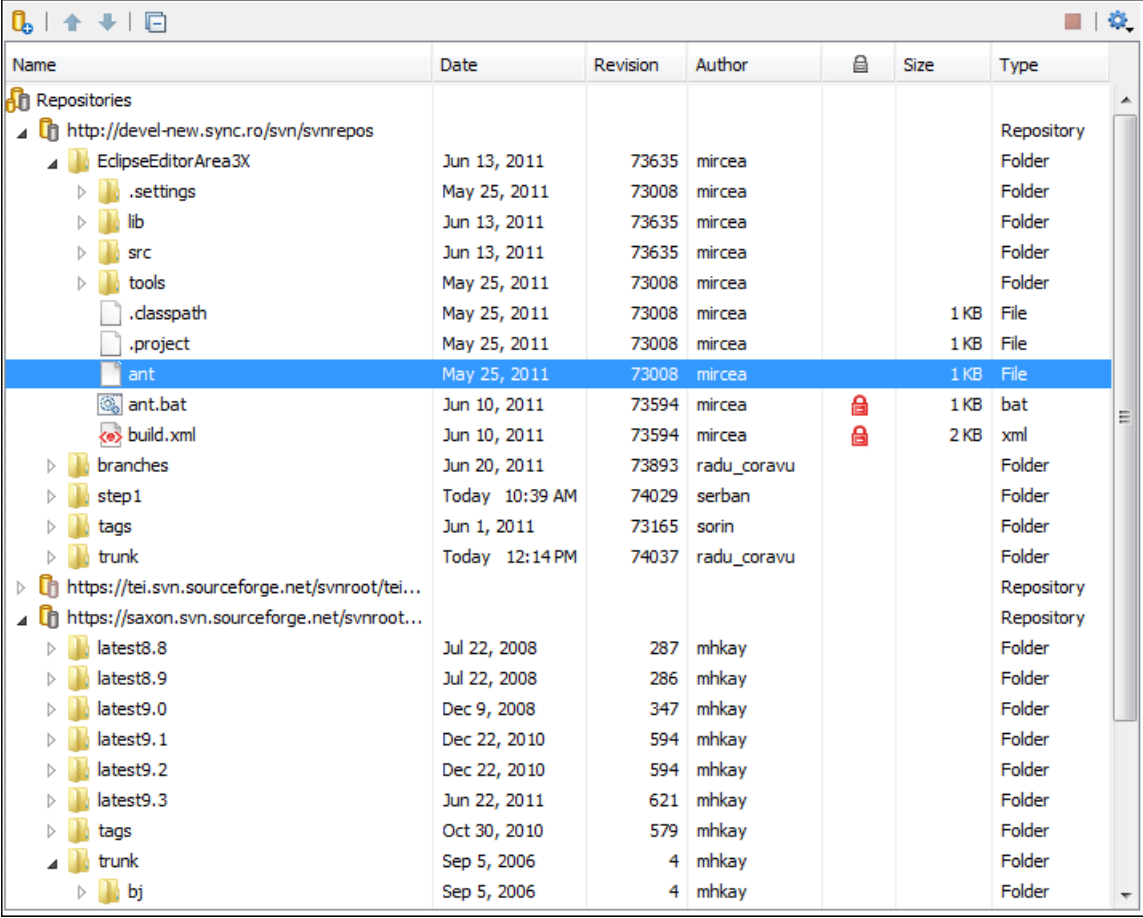
The other views that support the main working area are also presented in this section.

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 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	Lock	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 462: 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.

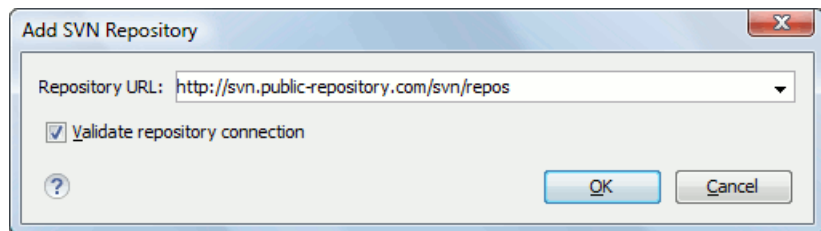


Figure 463: Add SVN Repository Dialog Box

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 (<u>Ctrl + Shift + L (Command + Shift + L on OS X)</u>) 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 (<u>Ctrl + Shift + I (Command + Shift + I on OS X)</u>) 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 (<u>Ctrl + H (Command + T on OS X)</u>)	Displays the history of the selected resource. At the start of the operation, you can set filtering options.
 Show Annotation (<u>Ctrl + Shift + A (Command + Shift + A on OS X)</u>)	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 (<u>Ctrl + G (Command + G on OS X)</u>)	This action allows you to see the graphical representation of a resource history. For more details about a resource revision graph see the section Revision Graph . This operation is enabled for any resource selected into the Repositories view or Working Copy view.
Copy URL Location (<u>Ctrl + Alt + U (Command + Alt + U on OS X)</u>)	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 (<i>HEAD</i>), to allow restoring previous versions of an item.
Move to (<u>Ctrl + M (Command + M on OS X)</u>)	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 (<u>Ctrl + Shift + F (Command + Shift + F on OS X)</u>)	Allows you to create a folder in the selected repository path (available only for folders).
Locking	(available only for files):
	 Lock (<u>Ctrl + K (Command + K on OS X)</u>) 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:

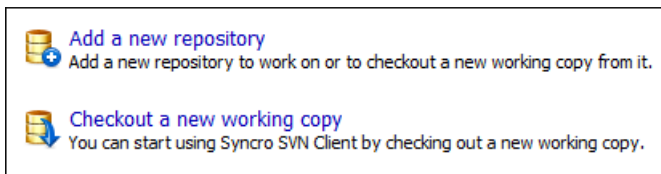


Figure 464: Repositories View Actions

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 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-ci		●	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 465: 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).

UserGuide										
All Files					Modified		Incoming	Outgoing	Conflicts	
Name					Remote date	Remote revision	Remote author	Size	Type	
UserGuide					Today 11:44 AM	12368	sorin		Folder	
DITA									Folder	
img									Folder	
references									Folder	
tasks									Folder	
topics									Folder	
ignore-resources-working-copy.dita								2 KB	dita	
oxygenEntitiesDictionary.dita					Today 11:44 AM	12368	sorin	2 KB	dita	
preferences-svn-working-copy.dita								4 KB	dita	
properties-view.dita								2 KB	dita	
revert-changes.dita								3 KB	dita	
show-history.dita								1 KB	dita	
svn-main-menu.dita								26 KB	dita	
svn-main-toolbar.dita								3 KB	dita	
svn-toolbar.dita								2 KB	dita	
tree-conflict.dita								1 KB	dita	
update-working-copy.dita								4 KB	dita	
views.dita								3 KB	dita	
working-copy-menu.dita								20 KB	dita	
working-copy-settings.dita					Today 11:28 AM	12366	sorin	2 KB	dita	
working-copy-view.dita								15 KB	dita	
EditorUserManual.ditamap								102 KB	ditamap	
.project						12368		1 KB	File	
build.xml					Nov 17, 2010	12337	sorin	14 KB	xml	
build_part.xml					Nov 16, 2010	12323	sorin	18 KB	xml	
userguide.xpr								2 MB	xpr	

Figure 466: 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:
 -  **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 Developer presents *unversioned* and *ignored* folders with *empty* depth when **Show unversioned directories content** or **Show ignored directories content** options are disabled.
-  **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 Developer.
 - 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 Developer 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 Developer 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 Developer 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 disabled, 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:

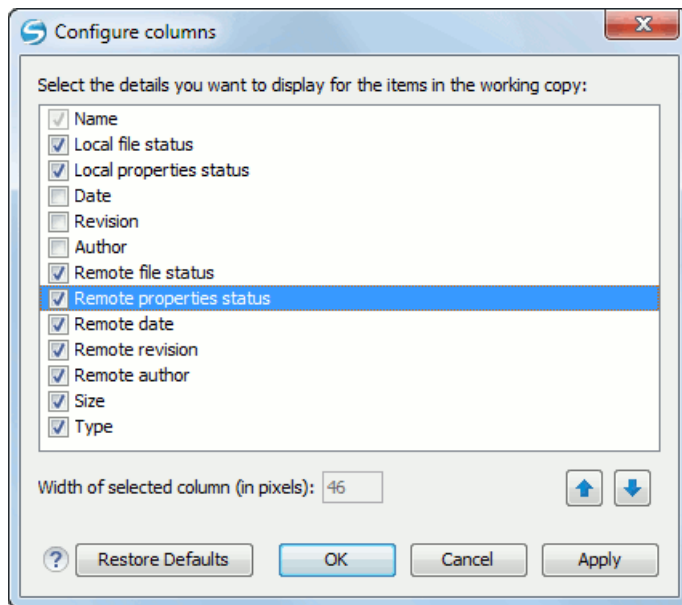


Figure 467: Configure Columns of Working Copy View

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 Developer 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 Developer 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 <i>Always switch to 'Modified' mode option</i> is selected.
Update (Ctrl (Command on OS X) + U)	Updates the selected resources to the <i>HEAD</i> 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 <i>depth</i> for the current folder. You can find out more about the <i>depth</i> term in the <i>sparse checkouts</i> 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 enabled only for modified resources. See <i>Revert your changes</i> 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 <i>Revert your changes</i> 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 <i>Drop incoming modifications</i> .
 Mark Resolved (Ctrl (Command on OS X) + Shift + R)	Instructs the Subversion system that you resolved a conflicting resource. For more information, see <i>Merge conflicts</i> .
 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 <i>Merge conflicts</i> section for more information about how you can solve the pseudo-conflicts.

Create patch (Ctrl (Command on OS X) + Alt + P)

Compare with:

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](#).

- **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 enabled 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:

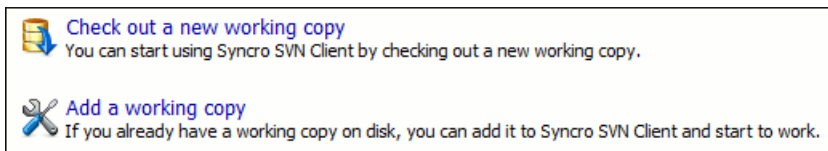


Figure 468: All Files Panel

For **Modified**, **Incoming**, **Outgoing**, **Conflicts** view modes, the following actions may be available, depending on the current working copy state in various contexts:

-  Information message - Informs you why there are no resources presented in the currently selected view mode.
-  **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.

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 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.

Revision	Date	Changes	Author	Message
Today (1 revision)				
16572	2011-11-30 16:15:29	1	mihai	oXygen-users meetup
Last week (1 revision)				
16476	2011-11-22 09:22:33	1	mihai	oXygen-user-meetup
Last month (35 revisions)				
16309	2011-10-26 16:01:09	1	sorin	Fixed DocumentationTest.
16246	2011-10-25 10:32:58	1	bogdan	Reviewed.
16245	2011-10-24 17:39:51	1	mihaela	Small corrections
16244	2011-10-24 17:32:24	1	george	More updates on new features.
16243	2011-10-24 17:21:46	1	george	More updates on new features.
16242	2011-10-24 17:08:52	1	george	Update Saxon, move components section last.
16241	2011-10-24 17:02:52	1	george	Just formatting.
16240	2011-10-24 17:01:39	1	george	More changes on the new features.
16239	2011-10-24 16:45:35	1	sorin	Fixed broken links.

More updates on new features.

Action	Name	Path	Copied from
	site.xml	/www.oxygenxml.com/trunk/xml	

Figure 469: 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.

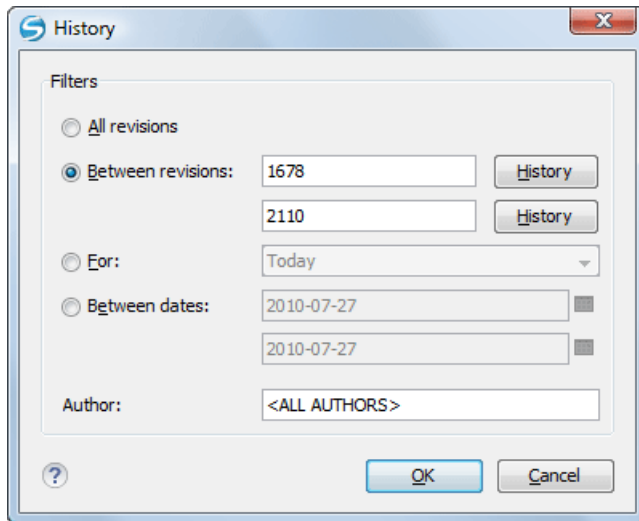


Figure 470: History Filters Dialog Box

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 enabled only when you select a file.
Open	Opens the selected revision of the file into the Editor. This is enabled 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 <i>BASE</i> 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 <i>BASE</i> 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 enabled when the resource history was launched for a local working copy resource.  Note: For items displayed in the Affected Paths section that were <i>added</i> , <i>deleted</i> , or <i>replaced</i> , 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 <i>the Export dialog box</i> 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 <i>the annotations for a file and displays them in the Annotations view</i> , along with the history of the file in the History view.
Change	Allows you to change commit data for a file: • <i>Author</i> - Changes the name of the SVN user that committed the selected revision. • <i>Message</i> - 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 enabled 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.

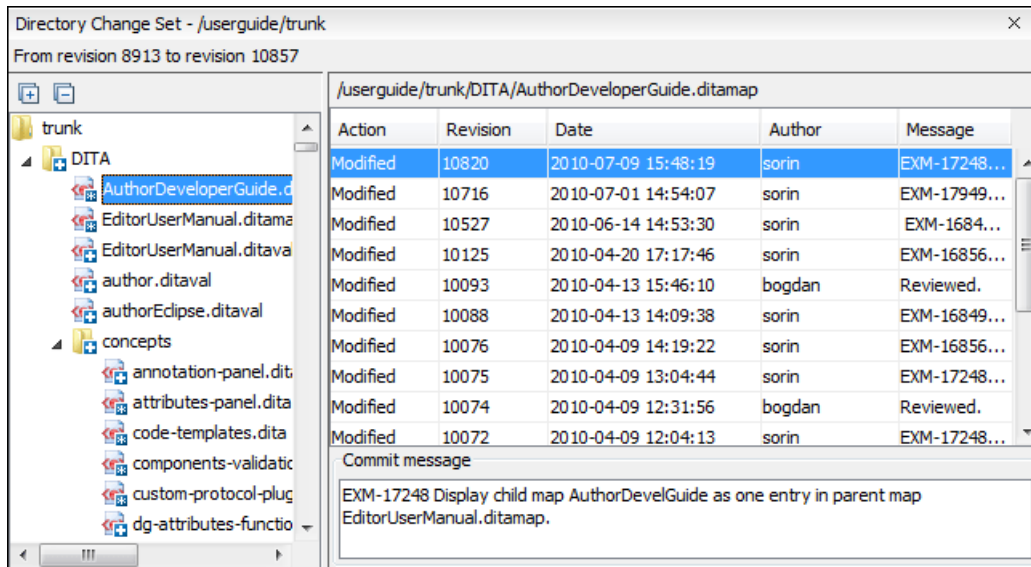


Figure 471: Directory Change Set View

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 <i>the Export dialog box</i> 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 <i>the annotations for a file and displays them in the Annotations view</i> , 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 <i>Obtain information for a resource</i> .

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.

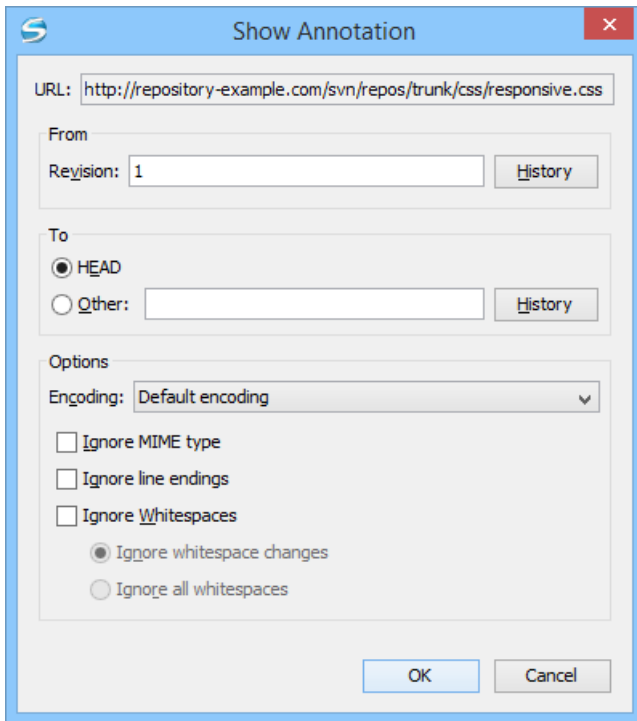


Figure 472: Show Annotation Options Dialog Box

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.

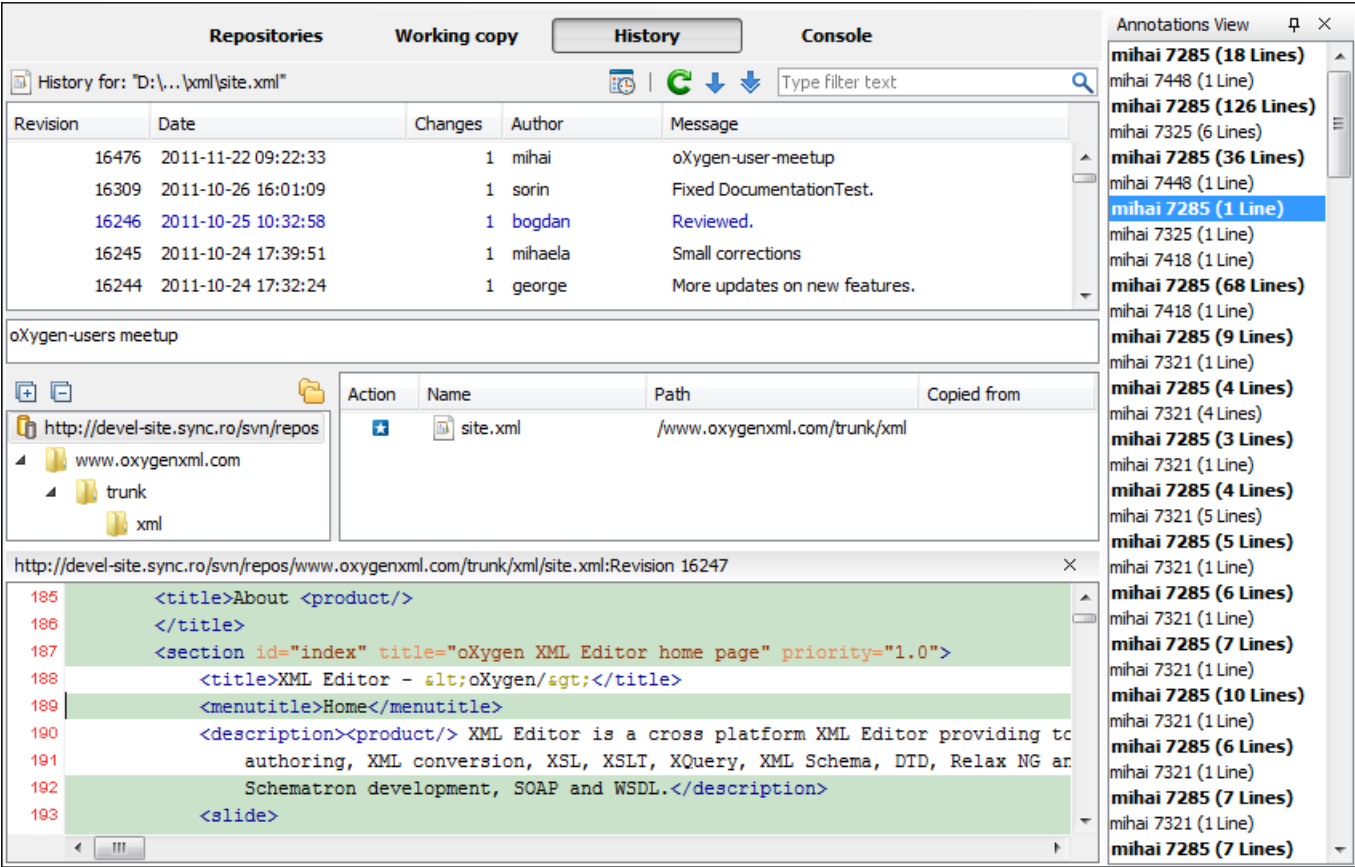


Figure 473: Annotations View

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, <i>the History dialog box</i> 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, <i>the History dialog box</i> 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 enabled, the file is treated as a text file and ignores what the SVN system infers from the <code>svn:mime-type</code> property.
Ignore line endings	If enabled, the differences in line endings are ignored when the annotation is computed.
Ignore whitespaces	If enabled, it allows you to specify how the whitespace changes should be handled. When enabled, 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: Enabling 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 Developer, 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.

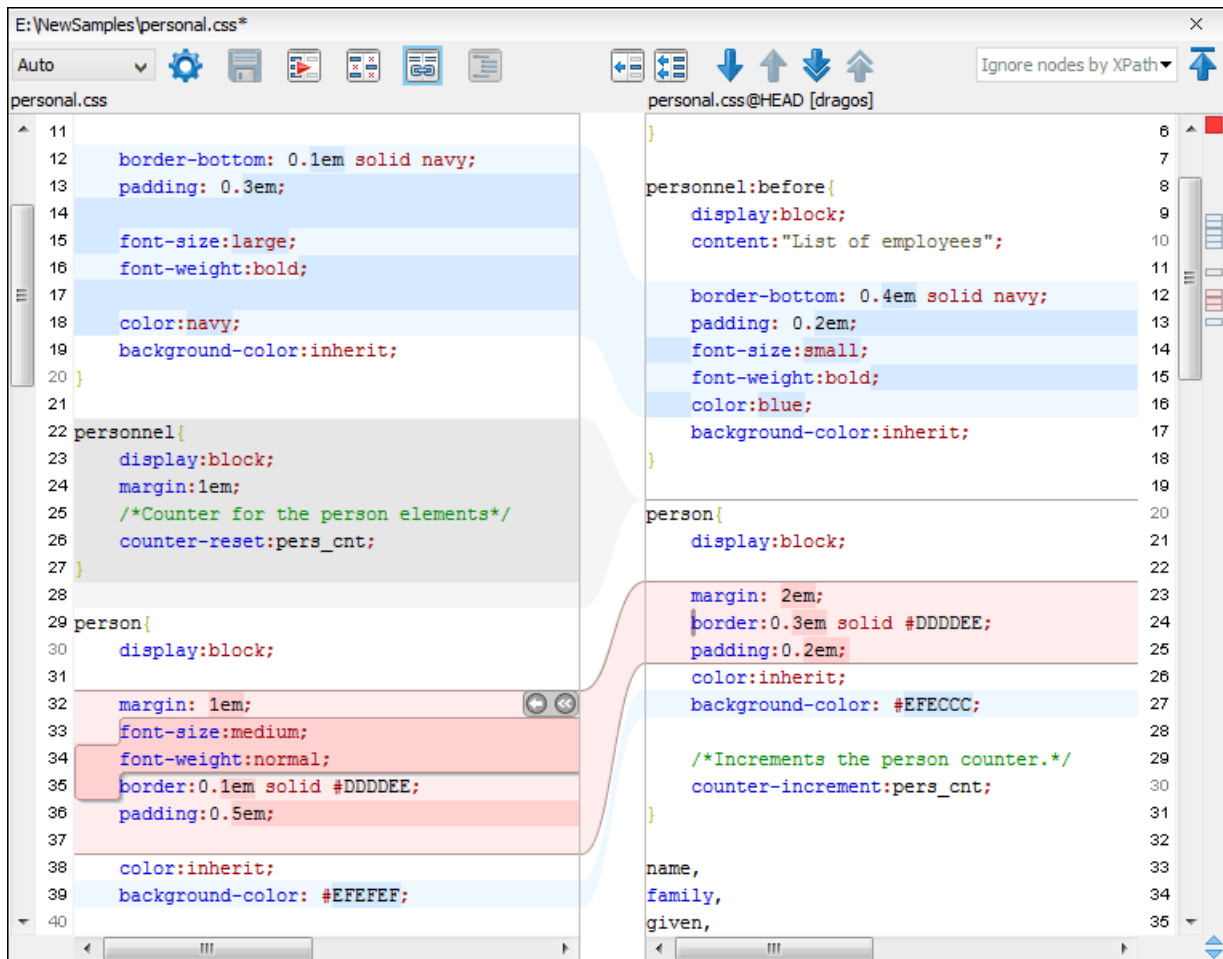


Figure 474: Compare View

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 475: 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 of the application window.



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

Synchronizes scrolling so that a selected difference can be seen on both sides of the application window. This action enables/disables the previously described behavior.



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 the same in both files to make it easier to compare the files.



Copy Change from Right to Left

Copies the selected difference from the target file in the right side to the source file in the left side.



Copy All Changes from Right to Left

Copies all changes from the target file in the right side to the source file in the left side.

 Next Block of Changes (Ctrl + Period (Command + Period on OS X))

Jumps to the next block of changes. This action is disabled 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 disabled 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 disabled 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 disabled 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 2.0 expression to ignore certain nodes from the comparison. This field is only available when comparing XML documents using the **XML Fast** or **XML Accurate** algorithms. If an XPath 2.0 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 from the **Window > Show View** menu.

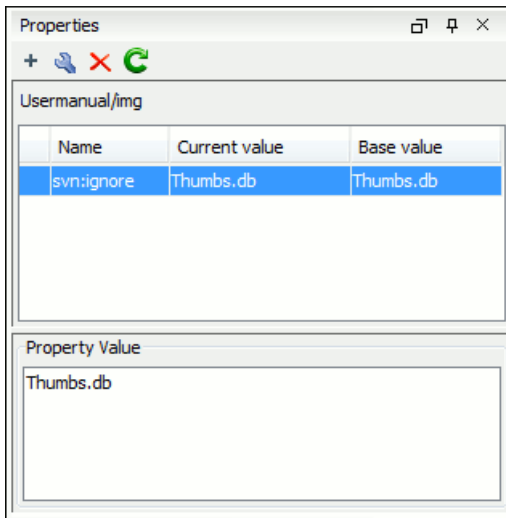


Figure 476: Properties View

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 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 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](#).

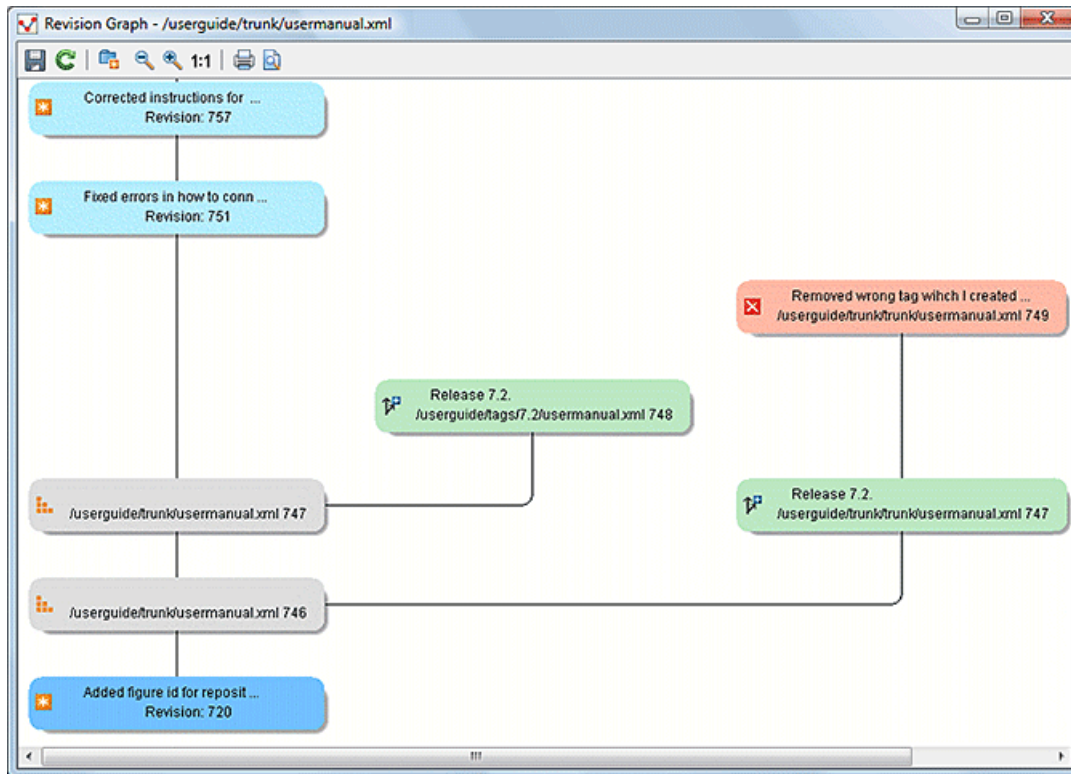


Figure 477: Revision Graph of a File Resource

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.

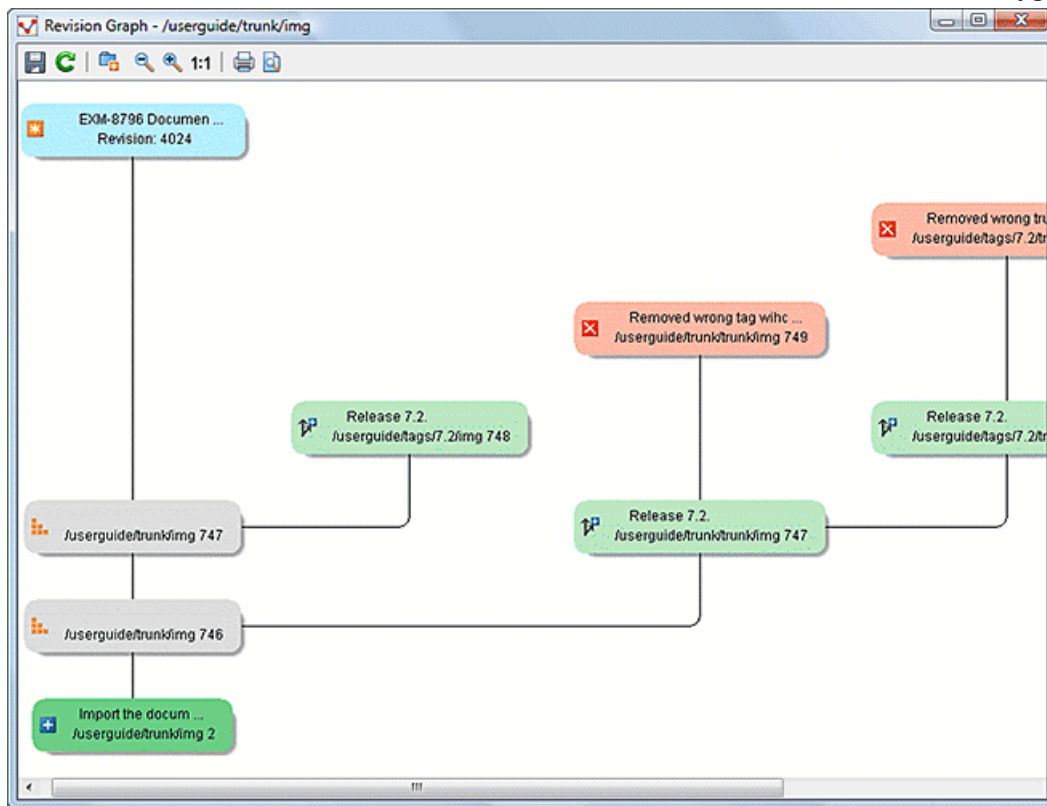


Figure 478: Revision Graph of a Directory (Direct Changes)

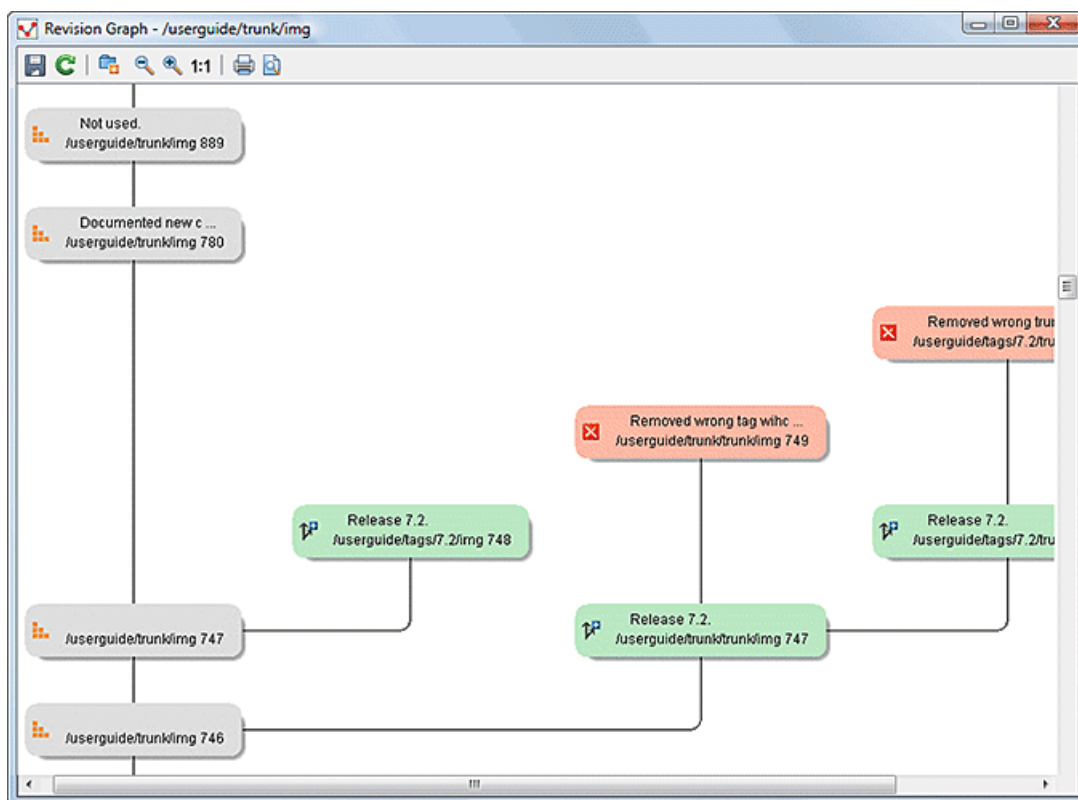


Figure 479: Revision Graph of a Directory (Including Indirect Changes)

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.
1:1 Reset 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 Developer SVN Preferences

The options used in the SVN client are saved and loaded independently from the Oxygen XML Developer options. However, if Oxygen XML Developer 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 Developer 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 Developer 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 Developer expects absolute paths for local file system items.
- *Relative Paths* - The Oxygen XML Developer only accepts relative paths in the form `~/...`, where `~` is the user home directory.
- *Path Validation* - Oxygen XML Developer 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 Developer 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 Developer expects absolute paths for local file system items.
 - *Relative Paths* - The Oxygen XML Developer 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 Developer 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.

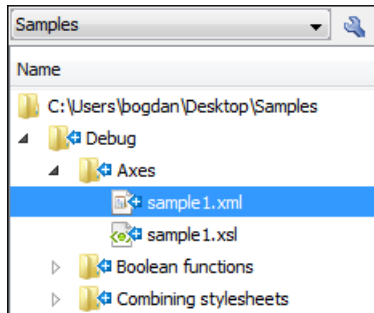


Figure 480: Incoming Changes

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 Developer 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).

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 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 Developer user interface as if from the command line of the operating system shell. Oxygen XML Developer 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 enabled 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 Developer, the *Save all files before calling external tools option* (in the **Open/Save** preference page) should be enabled 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.

Integrating the Ant Tool

This is an example for integrating *the Ant build tool* into Oxygen XML Developer:

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 **Command results** panel:

```
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
```

Chapter 17

Common Problems

Topics:

- [Performance Problems](#)
- [Common Problems and Solutions](#)

A compilation of common problems and their solutions.

This section provides a variety of common problems and their solutions.

Performance Problems

This section contains solutions for some common performance problems that may appear when running Oxygen XML Developer.

Related information

[Editing Documents with Long Lines](#) on page 513

[Loading Large Documents](#) on page 511

[External Tools](#) on page 997

Performance Issues with Large Documents

When starting Oxygen XML Developer 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 Developer from a command-line script, the default maximum memory is 512 MB.

While editing large documents in Oxygen XML Developer, if you see that performance slows down considerably over time, then 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 Developer 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 Developer 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.

External Processes

The [Memory available to the built-in FOP option](#) controls the amount of memory allocated to generate PDF output with the built-in Apache FOP processor. If Oxygen XML Developer throws an *Out Of Memory* error, [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 115

[Custom Engines Preferences](#) on page 119

[External Tools Preferences](#) on page 137

Common Problems and Solutions

This chapter presents common problems that may appear when running the application and the solutions for these problems.

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

What details should I add to my request for technical support on the online form in the product website?

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.

To generate an Oxygen XML Developer 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.

XML Document Takes a Long Time to Open

If Oxygen XML Developer takes a long time to open a document, check the following:

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 enabling 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 Developer](#).

Oxygen XML Developer Takes Several Minutes to Start

Problem: Some anti-virus software can cause Java applications, such as Oxygen XML Developer, to start very slowly due to scanning of compressed archives (such as the JAR libraries that all Java applications use).

Solution: The solution is to add the Oxygen XML Developer folder to the list of exceptions in the anti-virus software settings.

Out Of Memory Error When Opening Large Documents

I am trying to open a file larger than 100 MB to edit it in Oxygen XML Developer, but it keeps telling me it runs out of memory (**OutOfMemoryError**). What can I do?

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 Developer](#).

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](#).

Special Characters are Replaced with a Square in Editor

My file was created with other application and it contains special characters (such as é, ©, ®, etc.) Why does Oxygen XML Developer display a square for these characters when I open the file in Oxygen XML Developer?

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**.

XSLT Debugger Is Very Slow

When I run a transformation in the **XSLT Debugger** perspective it is very slow. Can I increase the speed?

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 enabled.
- 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).

Scroll Function of my Notebook Trackpad is Not Working

I got a new notebook (Lenovo Thinkpad™ with Windows) and noticed that the scroll function of my trackpad is not working in Oxygen XML Developer.

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:

```
* , * , oxygen18.0.exe , * , * , WheelStd , 1 , 9
* , * , oxygenAuthor18.0.exe , * , * , WheelStd , 1 , 9
* , * , oxygenDeveloper18.0.exe , * , * , WheelStd , 1 , 9
* , * , syncroSVNClient.exe , * , * , WheelStd , 1 , 9
* , * , diffDirs.exe , * , * , WheelStd , 1 , 9
* , * , diffFiles.exe , * , * , WheelStd , 1 , 9
```

Crash at Startup on Windows with an Error Message About a File nvogl32.dll

I try to start Oxygen XML Developer on Windows but it crashed with an error message about “Fault Module Name: nvogl32.dll”. What is the problem?

It is an OpenGL driver issue that can be avoided by creating an empty file called `opengl32.dll` in the Oxygen XML Developer install folder (if you start Oxygen XML Developer 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 Developer (if you start Oxygen XML Developer with the `oxygen.bat` script). The home folder of the Java virtual machine that runs Oxygen XML Developer 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).

Oxygen XML Developer Crashed on My Mac OS X Computer

Oxygen XML Developer crashed the Apple Java virtual machine/Oxygen XML Developer could not start up on my OS X computer due to a JVM crash. What can I do?

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.

Wrong Highlights of Matched Words in a Search in User Manual

When I do a keyword search in the User Manual that is included with the Oxygen XML Developer 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?

This does not happen when Oxygen XML Developer runs with a built-in Java virtual machine, that is a JVM that was installed by Oxygen XML Developer in a subfolder of the installation folder (for example, on Windows and Linux when installing Oxygen XML Developer with the installation wizard specific for that platform). When Oxygen XML Developer 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).

Keyboard Shortcuts Do Not Work

The keyboard shortcuts listed in the [Menu Shortcut Keys preferences](#) do not work. What can I do?

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.

After Installing Oxygen XML Developer I Cannot Open XML Files in Internet Explorer Anymore

Before installing Oxygen XML Developer 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 Developer. How can I load XML files in Internet Explorer again?

XML files are opened in Oxygen XML Developer because Internet Explorer uses the Windows system file associations for opening files and you associated XML files with Oxygen XML Developer 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 Developer 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()
```

I Cannot Associate Oxygen XML Developer With a File Type on My Windows Computer

I cannot associate the Oxygen XML Developer 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 oxygenDeveloper18.0.exe. When I select the file oxygenDeveloper18.0.exe in the Windows file browser dialog box, the Oxygen XML Developer application is not added to the list of applications in the **Open With** dialog box. What can I do?

The problem is due to some garbage Windows registry entries remained from versions of Oxygen XML Developer older than version 9.0. Uninstall all your installed versions of Oxygen XML Developer and run a registry cleaner application for cleaning these older entries. After that just reinstall your current version of Oxygen XML Developer and try again to create the file association.

Gray Window on Linux With the Compiz / Beryl Window Manager

I try to run Oxygen XML Developer 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 Developer window responds to user actions but after opening and closing an Oxygen XML Developer dialog or after resizing the Oxygen XML Developer window or a view of the Oxygen XML Developer window the content of this window becomes gray and it does not respond to user actions. What is wrong?

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 Developer application or switch to other window manager.

Drag and Drop Without Initial Selection Does Not Work

When I try to drag with the mouse an unselected file from the **Project** 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?

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 Developer application.

Set Specific JVM Version on Mac OS X

How do I configure Oxygen XML Developer to run with the version X of the Apple Java virtual machine on my Mac OS X computer?

Oxygen XML Developer 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**.

Segmentation Fault Error on Mac OS X

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?

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 `OxygenDeveloper.app/Contents/MacOS` folder. For browsing the `.app` folder you have to (**CMD+click**) the Oxygen XML Developer icon and select **Show Package Contents**.

Damaged File Associations on OS X

After upgrading OS X and Oxygen XML Developer, it is no longer associated to the appropriate file types (such as XML, XSL, XSD, etc.) How can I create the file associations again?

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 Developer installations on your hard drive.
2. Delete them by dragging them to the Trash.
3. Clear the Trash.
4. Unpack the Oxygen XML Developer 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 Developer icon.
9. Double-click the file.
10. Accept the confirmation.

When you start Oxygen XML Developer the file associations should work correctly.

I Cannot Connect to SVN Repository from Repositories View

I cannot connect to a SVN repository from the **Repositories** view of SVN Client. How can I find more details about the error?

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.

Signature Verification Failed Error on Open or Edit a Resource from Documentum

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.
```

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.jar=false` parameter, as explained in the [Setting a Parameter in the Launcher Configuration File / Startup Script](#) topic.

Compatibility Issue Between Java and Certain Graphics Card Drivers

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. 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.

Image Appears Stretched Out in the PDF Output

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.

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.

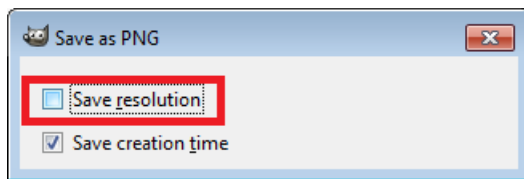
The PDF processor that comes out of the box with the application is the open-source Apache FOP processor. Here is what Apache FOP does when deciding the image size:

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.

DITA PDF Transformation Fails

To generate the PDF output, Oxygen XML Developer uses the DITA Open Toolkit.

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 Developer 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).

However, the execution of the transformation scenario may still fail. Reported errors include:

- [exec] HHC5010: Error: Cannot open "fileName.chm". Compilation stopped. - 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.
- [exec] HHC5003: Error: Compilation failed while compiling fileName - 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.

DITA Map Transformation Fails (Cannot Connect to External Location)

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](#)

Topic References Outside the Main DITA Map Folder

Referencing to a DITA topic, map or to a binary resource (for example: image) that is located outside of the folder where the main DITA map is located usually leads to problems when publishing the content using the DITA Open Toolkit. The DITA OT does not handle it well when links to topics that 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.

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.

PDF Processing Fails to Use the DITA OT and Apache FOP

There are cases when publishing DITA content fails when creating a PDF file. This topic lists some common problems and solutions.

- 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)
```

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.

- 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
```

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 Developer, 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.
- 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.
```

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 Developer, and search in it for the *unique_4_Connect_42_wrongID* id.
5. 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.

Navigation to the web page was canceled when viewing CHM on a Network Drive

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.

As a workaround, copy the CHM file on your local system and view it there.

Alignment Issues of the Main Menu on Linux Systems Based on Gnome 3.x

On some Linux systems based on Gnome 3.x (Ubuntu 11.x, 12.x), the main menu of Oxygen XML Developer has alignment issues when you navigate it using your mouse.

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 Developer, go to the installation directory of Oxygen XML Developer and rename or move the `jre` folder. If Oxygen XML Developer 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.

SVG 1.2 Rendering Issues

Oxygen XML Developer 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 Developer. 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.

MSXML 4.0 Transformation Issues

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.

Error Message

Could not create the 'MSXML2.DOMDocument.4.0' object.
Make sure that MSXML version 4.0 is correctly installed on the machine.

To fix this issue, go to the Microsoft website and get the latest MSXML 4.0 service pack.

'Address Family Not Supported by Protocol Family; Connect' Error

Question: 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 Developer 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 fix for this problem is to set the `java.net.preferIPv4Stack` Java property to `true` (`java.net.preferIPv4Stack=true`).

In order to do this, follow this procedure:

1. Edit the `custom_commons.vmoptions` file that is located in the Oxygen XML Developer installation folder, by adding the Java property `java.net.preferIPv4Stack=true`, and save it in the same location.
2. Restart Oxygen XML Developer.
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`.

Chapter 18

Glossary

Topics:

- [Active cell](#)
- [Apache Ant](#)
- [Block element](#)
- [Bookmap](#)
- [DITA Map](#)
- [DITA_OT_DIR](#)
- [Inline element](#)
- [Java Archive](#)
- [Named User](#)

Active cell

The selected cell in which data is entered when you begin typing. Only one cell is active at a time. The active cell is bounded by a heavy border.

Apache Ant

Apache Ant (Another Neat Tool) is a software tool for automating software build processes.

Ant

Block element

A block element is intended to be visually separated from its siblings, usually vertically. For instance, a paragraph or a list item 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.

DITA Map

A DITA map is 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 is 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 1.8, the default DITA OT directory is:

`[OXYGEN_INSTALL_DIR]/frameworks/dita/DITA-OT` (or if you are using DITA 2.x, `[OXYGEN_INSTALL_DIR]/frameworks/dita/DITA-OT2.x`). You can also specify a custom directory.

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

JAR (Java ARchive) 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.

Java Archive (JAR)

JAR

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 Developer) 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 Developer [End User License Agreement](#).

Notice

Copyright

Oxygen XML Developer User Manual

Syncro Soft SRL.

Copyright © 2002-2016 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 Developer End User License Agreement, as well as information regarding support for this product, while under warranty, is available through the [Oxygen XML Developer 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 Developer website](#).

Downloading documents. For the most current versions of documentation, see the [Oxygen XML Developer 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 Developer website](#).

Index

A

- Actions subtab (Author Tab) (Document type configuration dialog box) 63
- Add button 666
- Add favicon to WebHelp 639, 664
- Add items to a project 20, 231
- Add master file 244
- Address Family Not Supported by Protocol Family; Connect 1009
- Advanced Tab (DITA OT Transformations) 576
- Annotations preferences 85
- Archives 695, 696, 699
 - browse 696
 - edit 699
 - file browser 696
 - modify 696
- Association Rules Tab (Document type configuration dialog box) 61
- Author Editor 242, 249
 - edit markup 249
 - whitespace handling 242
 - versions differences 242
- Author Tab (Document type configuration dialog box) 63
- Automating WebHelp Output 669
- WebHelp plugin 669

B

- BaseX 740
- XQJ 740
- XQuery 740
- BaseX contextual menu actions 739
- BaseX database connections 739
- Batch transformations 600
- Berkeley DB XML contextual menu actions 719
- Berkeley DB XML database connection 718
- Berkeley DB XML debugging 722, 748
- Bidirectional text 192, 195
- Grid Mode 195
- Text Mode 192

C

- Catalogs Tab (Document type configuration dialog box) 71
- Check for add-on updates 48
- Classpath Tab (Document type configuration dialog box) 62
- Common Problems 45, 1000
- floating license server 45
- Comparing and Merging Documents 519, 525, 526, 527, 528, 530, 531, 532, 533, 888, 894, 895, 896, 897, 899, 900, 901, 902
 - Compare Directories tool 530, 531, 532, 533, 899, 900, 901, 902
 - compare images view 533, 902
 - Compare menu 531, 532, 900, 901
 - user interface 531, 900
 - Compare Files tool 525, 526, 528, 894, 895, 897
 - Compare menu 526, 895

- Comparing and Merging Documents (*continued*)
 - Compare Files tool (*continued*)
 - compare toolbar 528, 897
 - Edit menu 526, 895
 - File menu 525, 894
 - Find menu 526, 895
 - main menu 525, 894
 - Compare Files tool Compare Directories tool 527, 528, 531, 532, 896, 897, 900, 901
 - Help menu 528, 532, 897, 901
 - Options menu 527, 531, 896, 900
- Compare Files tool compare fragments 519, 888
- Compile LESS to CSS 463
- Configuration 99
- CSS validator 99
- Configure Application 94
 - Editor preferences 94
 - spell check 94
- Configure BaseX Connection 739
- Configure Berkeley DB XML database connection 718
- Configure Calabash with XEP 603
- Configure Documentum xDB database connection 731
- Configure eXist database connection 723
- Configure generic JDBC database connection 737
- Configure IBM DB2 database connection 706
- Configure MarkLogic connection 727
- Configure MarkLogic database connection 726
- Configure Microsoft SQL Server database connection 709
- Configure MySQL database connection 735
- Configure Oracle database connection 712
- Configure PostgreSQL database connection 716
- Configure the application 57
 - Appearance Preferences 57
 - Colors 57
- Configure the Application 55, 58, 59, 73, 74, 75, 76, 77, 78, 79, 80, 83, 84, 87, 88, 89, 90, 94, 96, 97, 99, 101, 104, 105, 106, 107, 109, 110, 111, 112, 114, 115, 118, 119, 120, 121, 122, 124, 127, 128, 129, 130, 131, 132, 134, 135, 136, 137, 139, 140, 142, 143, 144, 145, 147, 149, 152, 153, 154
- (S)FTP 144
- Accessibility 152
- archive 135
- BOM 73
- byte order mark 73
- certificates 122
- character encoding 73
- configure toolbars 152
- customize default options 147
- custom validation 97
 - data sources 124, 127
 - download links for database drivers 127
 - table filters 127
- Diff 132
- Diff appearance 134
- Diff directories 134
- Diff directories appearance 135
- document type association 59

Configure the Application (*continued*)

- editor preferences 74
 - Editor preferences 74, 75, 76, 77, 78, 79, 80, 83, 84, 87, 88, 89, 90, 94, 96
 - code templates 90
 - content completion 84
 - document checking 96
 - document templates 94
 - elements and attributes by prefix 88
 - format 79
 - format - CSS 83
 - format - JavaScript 83
 - format - XML 80
 - grid 77
 - open/save 88
 - pages 75
 - print 74
 - save hooks 89
 - schema design 78
 - syntax highlight 87
 - text 76
 - text/diagram 77
 - validation preferences 96
- editor variables 154
- encoding 73
- external tools 137
- file types 140
- fonts 58
- global 55
- HTTP(S)/WebDAV preferences 143
 - import 120, 121
 - date/time patterns 121
- import/export global options 149
- menu shortcut keys 139
- messages 131, 145
- outline 145
- perspectives layout 58
- plugins 136
- proxy preferences 142
- reset global options 149
- Sample XML Files Generator 104
 - scenarios management 153
 - Export Global Transformation Scenarios 153
 - Export Global Validation Scenarios 153
 - Import Global Transformation Scenarios 153
 - Import Global Validation Scenarios 153
- SSH 145
- SVN 128
- SVN Diff 130
- UTF-8 73
- views 145
- working copy 129
- XML 99
- XML catalog 99
- XML parser 101
- XProc engines 105
- XSLT 106
- XSLT/FO/XQuery 106
 - XSLT/FO/XQuery preferences 107, 109, 110, 111, 112, 114, 115, 118, 119
 - custom engines 119
 - debugger 114

Configure the Application (*continued*)

- XSLT/FO/XQuery preferences (*continued*)
 - FO Processors 115
 - MSXML 110
 - MSXML.NET 111
 - profiler 115
 - Saxon6 107
 - Saxon HE/PE/EE 107, 112
 - Saxon-HE/PE/EE 107
 - Saxon HE/PE/EE advanced options 109, 114
 - XPath 118
 - XQuery 112
 - XSLTProc 109
- Configure the interface (Change interface colors) (Change color theme) 56
- Accessibility 56
- Configure Transformation Scenario Dialog Box 598
- Configure WebDAV connection 741
- Configuring Documentum (CMS) Support 750
- Content Completion in Schematron 490
- Content Completion subtab (Author Tab) (Document type configuration dialog box) 69
- Content Management System 749
- CMS integration 749
- Context sensitive WebHelp 668
- Contextual Menu subtab (Author Tab) (Document type configuration dialog box) 68
- Copy/Paste 278
- grid editor 278
- Create custom document templates 212
- Create custom editor variables 159
- create edit document type association 61
- Create New Project 20, 231
- Create Validation Scenario 287, 290
- Create validation scenario XSLT stylesheets 336
- Creating Documents 209
- New Document Wizard 209
- CSS subtab (Author Tab) (Document type configuration dialog box) 63
- Customize assert error messages 284
- Customize document templates 213
- Customize WebHelp Classic Mobile 667
- Customize WebHelp Responsive output 628
- Customizing Schematron Quick Fixes 498
- SQF 498
- Customizing WebHelp Responsive Systems 632, 663
 - add logo image in title area 632, 663
- Customizing WebHelp Systems 638, 639, 653, 656, 662, 663, 664, 665, 667
 - add videos to WebHelp output 639, 662
 - change style of ordered lists 639, 665
 - changing the style of WebHelp Mobile pages 667
 - customizing headers / footers 664
 - Customizing icons in the Table of Contents 663
 - Customizing WebHelp output with a custom CSS 656
 - Editing Scoring Properties for Search Results 638, 662
 - remove previous / next links 664
 - Table of contents customization 663
 - custom startup parameters 163
 - vmoptions reset to default values 163
- Custom term file 509
- Custom XML Refactoring Operations 325, 846

D

- Databases 689, 702, 705, 728, 730, 741, 745, 747, 748, 754
- debugging with MarkLogic 728, 730, 747, 748
- limitations of the MarkLogic debugger 730, 748
- Native XML databases 705
- Relational databases 705
- SharePoint connection 754
- WebDAV connection 741
 - XQuery 689, 745, 747
 - debugging 747
 - drag and drop from the Data Source Explorer 745
 - transformation 745
 - validation 689
- Data Source Explorer view 702
- Debugging imported modules 729, 748
- MarkLogic 729, 748
- Debugging XSLT/XQuery Documents 771, 774, 783, 787
- Java extensions 787
 - layout 771, 774, 783
 - information views 774
 - multiple output documents in XSLT 2.0 783
- XSLT/XQuery debugger 783
- Debugging XSLT / XQuery Documents 772
 - layout 772
 - Control toolbar 772
- Debugging XSLT XQuery 769
- Debug PDF transformation scenario 603
- deploying WebHelp Classic with Feedback 649
- deploying WebHelp Responsive with Feedback 615
- Detect Master Files 243
- Develop a Plugin 820, 834
- example - **UppercasePlugin** 834
- introduction 820
- Digital Signature 513, 515, 516, 518, 876, 878, 879
- canonicalizing files 515, 876
- certificates 515
- signing files 516, 878
- verifying the signature 518, 879
- Digital Signatures 513, 518
- digital signatures overview 513
- how to digitally sign XML content 518
- Disable browser caching 665
- DITA map document type 810
- schema 810
- DITA Map document type 810
 - Author extension 810
 - catalogs 810
- DITA Map Document Type 810, 813
 - Author extension 813
 - templates 813
- DITA MAP document type 810
- association rules 810
- DITA MAP Document Type 544, 810
 - Author extension 544, 810
 - transformation scenarios 544, 810
- DITA Map PDF - WISIWYG - Experimental transformations scenario 557
- Antenna House Formatter 557
- Prince Print with CSS 557

- DITA Maps 1007
 - transforming DITA Maps 1007
 - running an ANT transformation 1007
- DITA Map Transformation Scenario WebHelp Output 545, 810
- DITA Map WebHelp Classic Mobile transformation scenario 553
- DITA Map WebHelp Classic transformation scenario 550
- DITA Map WebHelp Classic with Feedback transformation scenario 555
- DITA Map WebHelp Responsive transformation scenario 545
- DITA Map WebHelp Responsive with Feedback transformation scenario 548
- DITA Topics document type 809
- association rules 809
 - Author extensions 809
 - catalogs 809
- schema 809
- DITA Topics Document Type 544, 808, 809
 - Author extensions 544, 809
 - templates 809
 - transformation scenarios 544, 809
- DocBook 799, 806
- Insert olink 799, 806
- DocBook 4 to WebHelp transformations scenarios 537, 795
- DocBook 5 to WebHelp transformation scenarios 541, 802
- DocBook Targetset document type 818
- association rules 818
- schema 818
- DocBook Targetset Document Type 818
- DocBook to DITA Transformation Scenario 540, 544, 798, 805
- DocBook to EPUB transformations scenario 540, 543, 798, 805
- DocBook V4 document type 795, 799
- association rules 795
 - Author extensions 795, 799
 - catalogs 795
 - templates 795
- schema 795
- DocBook V4 Document Type 537, 795
 - Author extensions 537, 795
 - transformation scenarios 537, 795
- DocBook V5 document type 802, 806
- association rules 802
 - Author extensions 802, 806
 - catalogs 802
 - templates 806
- schema 802
- DocBook V5 Document Type 540, 802
 - Author extensions 540, 802
 - transformation scenarios 540, 802
- Document type configuration dialog box 61
- Documentum (CMS) Support 749, 750, 751
- actions 751
- configuring a Documentum (CMS) data source 750
- Documentum CMS known issues 751
- Documentum xDB contextual menu actions 732
- Documentum xDB database connection 731
- Documentum xDB parser configuration 734
- Documentum xDB troubleshooting 735
- Duplicate transformation scenario 597

E

- Edit 150, 161, 205, 206, 209, 214, 215, 216, 217, 225, 230, 231, 295, 298, 299, 507, 510, 512, 513, 699
- archives 699
- associating a file extension 150
- change the user interface language 161
- change user interface language 161
- Character map 206
- check spelling 507
- check spelling in files 510
- close documents 230
- create new documents 209
- edit documents with long lines 513
- file properties 231
- find/replace 295
- find/replace (keyboard shortcuts) 298
- find and replace text in multiple files 299
- interface localization file 161
- localize the user interface 161
- open and close documents 209
- open documents 214
- open read-only files 513
- open remote documents (FTP/SFTP/WebDAV) 217
- open resource 225
- open the currently document in the system application 215
- Quick Find toolbar 298
- save documents 216
- scratch buffer 512
- Unicode documents 206
- Unicode support 206
- Unicode toolbar 206
- Editing Ant Build Files 365
- Editing CSS Stylesheets 459, 460, 461, 462
- Content Completion Assistant 460
- folding 461
- format and indent (pretty print) 462
- Outline view 461
- validation 460
- Editing Documents 230, 231, 253
- contextual menu of current editor tab 230
- drag and drop 253
- Using projects 231
- Editing JavaScript Documents 484
- Editing JavaScript Files 484, 486, 487
- Content Completion Assistant 486
- Outline view 487
- Text mode 484
- validating JavaScript files 486
- Editing JSON Documents 479, 480, 481, 482, 865
- convert XML to JSON 482, 865
- folding 480
- Grid mode 481
- Outline view 482
- syntax highlight 480
- Text mode 479
- Validating JSON Documents 482
- Editing LESS stylesheets 462
- Editing Modes 188
- Results View 188
- Editing NVDL Schemas 475, 476, 477, 478
- Component Dependencies view 477
- Editing NVDL Schemas (*continued*)
- Outline view 477
 - schema diagram 475, 476
 - actions in the diagram view 476
 - full model view 476
- searching and refactoring actions 478
- Editing RelaxNG Schemas 472
- Component Dependencies View 472
- Editing Relax NG Schemas 464, 465, 466, 467, 468, 469, 473
- Resource Hierarchy/Dependencies View 469
 - schema diagram 464, 465, 466, 467, 468
 - actions 467
 - full model view 465
 - logical model view 465
 - Outline view 468
 - symbols 466
- searching and refactoring actions 473
- Editing Schematron 497
- Quick Assist 497
- Editing Schematron Documents 495
- searching and refactoring operations 495
- Editing Schematron Schemas 294, 489, 490, 492, 493, 496
- contextual editing 490
- Master Files context 490
- Resource Hierarchy/Dependencies View 493
- Schematron Outline view 492
- Search and Refactoring Operations Scope 496
- validation against Schematron 294
- Editing StratML Documents 483
- Editing SVG Documents 504, 505, 506, 886
- preview result pane 506
- standalone SVG viewer 505, 886
- Editing WSDL Document 456, 879
 - SOAP request 456, 879
 - composing a SOAP request 456, 879
- Editing WSDL Documents 440, 441, 442, 445, 448, 449, 450, 451, 452, 455, 456, 458, 874, 879, 881
- Component Dependencies view 448
- component occurrences 448
- composing web service calls with WSDL SOAP analyzer 456, 879
- content completion 442
- contextual editing 441
- generate documentation for WSDL documents 452, 874
- generate documentation for WSDL documents from command line 456
- generate documentation for WSDL documents in a custom format 455
- Outline view 442
- Quick Assist 451
- Resource Hierarchy/Dependencies view 445
- searching and refactoring operations 449
- searching and refactoring operations scope 450
 - SOAP request 458, 881
 - testing remote WSDL files 458, 881
 - UDDI registry browser 458, 881
- Editing XLIFF document 484
- 1.2 484
- 2.0 484
- Editing XML Documents 174, 182, 184, 187, 191, 233, 240, 245, 247, 252, 254, 259, 260, 261, 262, 265, 270, 280, 282,

- Editing XML Documents (*continued*)
 - 283, 285, 286, 293, 304, 305, 307, 308, 310, 312, 313, 315, 428, 860
- against a schema 282
 - associate a schema to a document 304, 305, 307
 - add schema association in XML instance 305
 - learning a document structure 307
 - setting a default schema 304
 - supported schema types 304
- checking XML well-formedness 280
- code templates 261
- content completion 261
- converting between schema languages 428, 860
 - document navigation 182, 252, 262
 - folding 252
 - outline view 182, 262
 - editor specific actions 247
 - smart editing 247
 - grouping documents in XML projects 174, 233, 312
 - large documents 312
 - new project 174, 233
 - project view 174, 233
- image preview 240
- including document parts with XInclude 313
- locking and unlocking XML markup 265
- making a persistent copy of results 315
- markup transparency 265
- Resource Hierarchy/Dependencies view 308
 - Sharing projects 240
 - team collaboration - Subversion 240
- status information 315
 - streamline with content completion 184, 187, 254, 259, 260
 - the Attributes view 184, 259
 - the Elements view 187, 260
 - the Entities view 187, 260
- Text mode contextual menu actions 270
- Text Mode specific actions 245
 - validation against a schema 191, 282, 283, 285, 286, 293
 - automatic validation 282
 - custom validation 285
 - marking validation errors 191, 283
 - references to XML Schema specification 293
 - resolving references to remote schemas with an XML Catalog 293
 - validation actions 282
 - validation example 293
 - validation scenario 286
- working with XML Catalogs 310
- Editing XML files in Master Files context 308
- Editing XML Schemas 373, 404, 409, 411, 413, 415, 419, 420, 423, 427, 433, 435, 855, 859, 867, 882
- Component Dependencies view 411
- contextual editing 404
 - generate documentation for XML Schema 420, 423, 427, 867
 - from command line 427
 - output formats 423
 - Custom format 423
 - DocBook format 423
 - HTML format 423
 - PDF format 423
- Resource Hierarchy/Dependencies view 409
- schema files generator 415, 855

- Editing XML Schemas (*continued*)
 - schema instance generator 419, 859
 - running from command line 419, 859
- schema regular expressions builder 433, 882
- searching and refactoring actions 413
- XML Schema 1.1 435
- Editing XProc Scripts 488
- Editing XQuery Documents 437, 438, 439, 873
- folding 438
- generate HTML documentation 439, 873
- Editing XSL Stylesheets 349, 358, 364
- Component Dependencies view 349
- quick assist support 358
- XSpec 364
- Editing XSLT Schemas 335
- contextual editing 335
- Editing XSLT Stylesheets 335, 337, 338, 342, 345, 346, 350, 351, 354, 356, 357, 361, 870
 - content completion 337, 338
 - in XPath expressions 338
- find XSLT references and declarations 350
- generate documentation for XSLT stylesheets 351, 870
 - generate documentation for XSLT Stylesheets 354, 356, 357
 - as HTML 354
 - from command line 357
 - in custom format 356
- Outline view 342
- refactoring actions 361
- Resource Hierarchy/Dependencies view 346
 - validation 335, 337
 - custom validation 337
- XSLT Input view 345
- XSLT stylesheet documentation 350
- Edit Large Documents 511, 512
 - larger than 300 MB 512
 - smaller than 300 megabytes 511
- Edit Menu 195, 206, 507, 510, 784
- Breakpoints 784
- Change Text Orientation 195
- Check Spelling 507
- Check Spelling in files 510
- Insert from Character Map 206
- editor highlights 316
 - Accessibility 316
- Edit transformation scenario 596
- Edit Validation Scenario 287, 290
- Edit validation scenario XSLT stylesheets 336
- Edit XHTML files 506
- Embedded Schematron Rules 491
- Relax NG 491
- XML Schema 491
- Enable Master Files 243
- EPUB Document Type 817
- Error 1067: Process Terminated Unexpectedly 45
- eXist contextual menu actions 724
- eXist database connection 722
- Expanding Unicode support 208
- fallback font support 208
- Export color themes 56
- share color themes 56
- Extend Oxygen with plugins 820
- implement plugin 820

- Extend Oxygen with Plugins 820, 821, 825, 826, 827, 828, 829, 833
 - implement plugin 821, 825, 826, 827, 828, 829, 833
 - CMS integration plugin 829
 - components validation plugin 825
 - custom protocol plugin 825, 827, 828
 - document plugin 829
 - general plugin 829
 - how to install a plugin 821
 - how to write a custom protocol plugin 833
 - resource locking custom protocol plugin 826
 - selection plugin 829
- Extensions Tab (Document type configuration dialog box) 73

F

- Facebook widget 635, 656
- File Menu 209, 214, 215, 216, 230, 316, 760, 761, 763, 766
 - Close 230
 - Close All 230
 - Close Other Files 230
 - Import Database Data 763
 - Import HTML File 766
 - Import MS Excel File 761
 - Import Text File 760
 - New 209
 - Open 214
 - Open URL 214
 - Print 316
 - Print Preview 316
 - Reopen 214
 - Save 216
 - Save All 216
 - Save as 216
 - Save to URL 216
 - View in Browser System Application 215
- Filters Tab (DITA Transformations) 576
- Find/Replace 295, 297, 298
 - Find All Elements/Attributes dialog box 297
 - keyboard shortcuts 298
 - Quick Find toolbar 298
- Find Menu 225, 295, 297, 298, 299
 - Find/Replace 295
 - Find/Replace in Files 299
 - Find All Elements 297
 - Open/Find Resource 225
 - Quick Find 298
- Flagging content 640, 666
- Floating license key replacement 43, 45
- Floating license report page 43
- Floating license server 40
- FO Processor tab (DITA OT Transformation) 574
- FO Processor Tab (XQuery Transformations) 570, 594
- FO Processor Tab (XSLT Transformations) 565, 586
- Format and indent 265
 - Format and Indent 82, 268
 - set Indent Size to zero 268

G

- Generate Documentation for an XML Schema 426
 - customizing the PDF output 426

- Generate Sample XML Files Advanced tab 419, 859
- Generate Sample XML Files Options tab 416, 857
- Generate Sample XML Files Schema tab 415, 856
- Generating Documentation for WSDL Documents 454
 - generating WSDL documentation in HTML format 454
- Generic database connections 735, 737, 738
- Getting Started 150, 165, 166, 167, 168, 169, 888
 - dockable views and editors 150
 - perspectives 165, 166, 167, 168, 169, 888
 - database 169
 - editor 166
 - tree editor 888
 - XQuery debugger 168
 - XSLT debugger 167
- Getting started resources 19
- Google Analytics integration 634, 659
- Google Plus widget 636, 658
- Google Search Integration 658
- grid editor 195
 - navigation 195
 - collapse all 195
 - collapse children 195
 - collapse others 195
 - expand all 195
 - expand children 195
- Grid Editor 192, 194, 277, 278
 - add nodes 278
 - clear column content 278
 - copy/paste 278
 - drag and drop 278
 - duplicate nodes 278
 - inserting table column 277
 - insert table row 277
 - layouts (grid and tree) 194
 - navigation 194
 - refresh layout 278
 - sort table column 277
 - start and stop editing a cell value 278
- Grid Mode Editor 192

H

- Help menu 21
 - Accessibility 21
 - Check for a New Version 21
 - Check for updates 21
 - install add-ons 21
 - online help 21
 - Register 21
 - Report problem 21
 - Support center 21
 - Tip of the day 21
- Highlight Component Occurrences 495
- HTTP Authentication Schemes 222
- NTLM 222

I

- IBM DB2 connection contextual menu actions 708
- IBM DB2 Database 706
 - Import color themes 56
 - copy color themes 56

- Import from Excel 763
- 2007 763
- 2010 763
- 2013 763
- Importing data 759
- Importing Data 760, 763, 766
 - from a database 763
 - table content as XML document 763
- from HTML files 766
- from text files 760
- Importing data from Excel 761
- Incorrect Function when Started 45
- Information view 315
- Inputs Tab (XProc Transformations) 588
- Installation 27, 29, 30, 31, 32, 33, 34
 - all platforms version 27, 29, 31, 32, 33
 - Linux 30, 33
 - multiple instances (Unix / Linux server) 34
 - OS X installation 29
 - unattended (Windows and Linux only) 27, 30
 - Windows installation 27, 32
 - Windows terminal server 32
- installing WebHelp Classic with Feedback 649
- installing WebHelp Responsive with Feedback 615
- Install new add-ons 48
- Integrate External XProc Engine 604
- integrate WebHelp plugin with DITA 669
- integrate WebHelp plugin with DocBook 674
- Integrating external tools 997
- Integrating Social Media in WebHelp 634, 656
- Integrating the Ant tool 997
- Integrating the WebHelp plugin with DITA OT 669

J

- JATS document templates 817
- JATS NISO Journal Article Tag Suite Document Type 817
- JATS transformation scenarios 560, 817
- Java system properties 159
 - custom system properties 159

K

- Kerberos authentication 222

L

- License 36, 37, 38, 39, 40, 44, 46, 47
 - floating (concurrent) license 38
 - floating license server 40, 44
 - license server installed on OS X Linux Unix 46
 - multiple named-user licenses 39
 - named-user license 37
 - register a license key 36
 - release floating license 39
 - releasing a license key 47
 - transferring a license key 47
 - unregistering a license key 47
- localize WebHelp with Feedback system emails 637, 661
- Localizing WebHelp Output 660

M

- Manage add-ons 48
- Manage IDs 270, 302
 - highlight ID occurrences in Text mode 270
 - search and refactor actions of ID IDREFS 302
- MarkLogic contextual menu actions 730
- MarkLogic database connection 725
- MarkLogic for the developer 727
- Master Files 242
 - benefits 242
 - Master Files contextual menu actions 244
 - Master Files validation transformation scenarios 244
- Menu subtab (Author Tab) (Document type configuration dialog box) 67
- Methods for customizing WebHelp Responsive 628
- Microsoft SQL Server database 709, 711
- Model view 186, 257
 - streamline with content completion 186, 257
 - the Model panel 186, 257
- Moving Renaming Schematron resources 494

N

- Native XML Databases 705
- Native XML Databases (NXD) 718, 719, 723, 724, 727, 732
 - database connections configuration 719, 724, 732
 - Berkeley DB XML 719
 - Documentum xDB (X-Hive/DB) 732
 - eXist 724
 - data sources configuration 718, 723, 727, 732
 - Berkeley DB XML 718
 - Documentum xDB (X-Hive/DB) 732
 - eXist 723
 - MarkLogic 727
- Network settings 142

O

- Open Office documents to DITA 558, 813
- Options Menu 54, 139, 149, 153
 - Export Global Options 149
 - Export Global Transformation Scenarios 153
 - Export Global Validation Scenarios 153
 - Import Global Options 149
 - Import Global Transformation Scenarios 153
 - Import Global Validation Scenarios 153
- Menu Shortcut Keys 139
- Preferences 54
- Reset Global Options 149
 - Options priority 146
- Options Tab (Ant Transformations) 580
- Options Tab (XProc Transformations) 590
- Oracle database connection contextual menu actions 714
- Oracle XML DB 712
- OutOfMemory 115, 1000, 1001
 - Out Of Memory 115, 884, 1000, 1001
 - large documents error 1001
- Large File Viewer 884
 - opening XML documents closed a long time ago 1001
- OutOfMemoryError 115, 1000, 1001
- Output Tab (Ant Transformations) 581

Output Tab (DITA OT Transformations) 578
 Output Tab (XQuery Transformations) 570, 594
 Output Tab (XSLT Transformations) 565, 587

P

Parameters Tab (Ant Transformations) 581
 Parameters Tab (DITA OT Transformations) 575
 Parameters Tab (XProc Transformations) 589
 Performance Problems 1000
 external processes 1000
 large documents 1000
 PostgreSQL database 716
 PostgreSQL database contextual menu actions 717
 Predefined XML refactoring operations 322, 842
 Preferences 54
 Pretty print 265
 Profiling 788
 XSLT stylesheets and XQuery documents 788
 Profiling XSLT Stylesheets and XQuery Documents 788, 789, 790
 profiling information 788, 789
 Hotspots view 789
 Invocation tree view 788
 XSLT/XQuery profiler 790
 Project Menu 179, 238
 Change Search and Refactor operations scope 179, 238
 Enable Master Files Support 179, 238
 Filters 179, 238
 New Project 179, 238
 Open Project 179, 238
 Reopen Project 179, 238
 Save Project As 179, 238
 Show Project View 179, 238
 Validate all project files 179, 238

Q

Querying Documents 437, 680, 681, 685, 686, 688, 689, 690
 running XPath and XQuery expressions 681
 XPath/XQuery Builder view 681
 running XPath expressions 680
 XPath toolbar 680
 XQuery 437, 685, 686, 688, 689, 690
 Input view 688
 Outline view 437, 686
 syntax highlight and content completion 685
 transforming XML documents; advanced Saxon B/SA options 690
 validation 689
 Quick Assist Support 264

R

Rectangular Selection 253
 Text Mode 253
 Refactoring 264
 Quick Assist 264
 Refactoring XML Documents 319, 840
 XML Refactoring Tool 319, 840
 Refresh WebHelp Classic with Feedback 651
 Refresh WebHelp Responsive with Feedback 617
 register license for WebHelp plugin 670

Relational Databases 703, 705, 707, 708, 709, 710, 712, 713, 716, 717, 735, 736, 737, 738, 743, 744
 connections configuration 708, 710, 713, 717, 736, 738
 generic JDBC 738
 IBM DB2 connection 708
 JDBC-ODBC connection 738
 Microsoft SQL Server 710
 MySQL 736
 Oracle 11g 713
 PostgreSQL 8.3 717
 data sources configuration 707, 709, 712, 716, 735, 737
 generic JDBC data source 737
 IBM DB2 707
 Microsoft SQL Server 709
 MySQL 735
 Oracle 11g 712
 PostgreSQL 8.3 716
 SQL execution support 743, 744
 drag and drop from the Data Source Explorer 743
 executing SQL statements 744
 SQL validation 744
 Table Explorer view 703
 Relax NG Schema Editor 464
 contextual editing 464
 Remove master file 244
 Resolve schema through xml catalog mappings 312

S

schema annotations 256
 Schema Tab (Document type configuration dialog box) 62
 Schematron Quick Fixes 319
 Selecting content 253
 Text Mode 253
 Sequence view 691
 Set XML Schema version 436
 SharePoint Browser 755
 SharePoint Connection 754, 755
 configuration 755
 SharePoint contextual menu actions 757
 Data Source Explorer 757
 Sharing preferences 148
 global options 148
 project options 148
 Sharing Preferences 240
 project options file 240
 Sharing Transformation Scenarios 240
 Sharing Validation Scenarios 240
 Skins Tab (DITA OT transformations) 572
 Social media 635, 656
 Social Media 634, 635, 636, 656, 657, 658, 659
 Social plugin 635, 656
 specify schema for Content Completion 256
 Spell Checking 510
 automatic spell check 510
 SQF 319
 Startup parameter 162, 163
 application launchers parameters 162
 command line scripts parameters 163
 Startup parameters 162
 Storing preferences 148
 global options 148

- Storing preferences (*continued*)
- project options 148
- Supported document types 794
- frameworks 794
- SVN Branches/Tags 936, 938, 940, 941, 942, 944, 948, 949
- create a branch/tag 936
 - merging 938, 940, 941, 942, 944
 - merge changes 940
 - merge two different trees 944
 - reintegrate a branch 942
 - reverse merge 940
 - synchronize branch 941
- relocate a working copy 949
- switch the repository location 948
- SVN Client 902, 903, 911, 912, 913, 914, 916, 918, 919, 924, 934, 935, 936, 950, 961, 962, 963, 964, 965, 973, 978, 980, 981, 982, 983, 986, 987, 988, 990, 993
- Annotations view 983
 - Compare view 986, 987, 988
 - Compare images view 988
 - toolbar 987
- Console view 990
 - define a repository location 912, 913
 - add/edit/remove repository locations 912
 - authentication 913
- define a working copy 916
- Help view 990
 - History view 978, 980
 - history filter dialog box 980
 - history filter field 980
- image preview 988
 - main window 902, 903, 911, 912
 - main menu 903
 - status bar 912
 - toolbar 911
 - views 902
 - obtain information regarding a resource 934, 935
 - request history 935
 - request status information 934
- Preferences 993
 - Properties view 988, 990
 - toolbar and contextual menu 990
 - Repositories view 964, 965
 - contextual menu actions 965
 - toolbar 964
 - Resource History view 981, 982
 - contextual menu on double selection 981
 - Directory Change Set view 982
- Revision Graph 990
- share a project 914
- sparse checkouts 963
 - SVN branches/tags 936, 950
 - patch 950
 - SVN properties 935, 990
 - Add / Edit / Remove 990
- SVN working copy resources 919
- synchronize with the SVN repository 924
- use an existing working copy 918
 - Working Copy view 973
 - contextual menu actions 973
 - working with repositories 961, 962
 - copy/move/delete resources 962

- SVN Client (*continued*)
 - working with repositories (*continued*)
 - export resources 961
- SVN working copy resources 923
 - lock / unlock resources 923
 - locked items 923
- SVN Working Copy Resources 919, 921, 922, 923, 924
- add resources to version control 919
- copy resources 922
- delete resources 921
- edit files 919
- ignore resources 921
 - lock/unlock resources 922, 923, 924
 - locking a file 924
 - scanning for locks 923
 - unlocking a file 924
- move resources 922
- rename resources 922
- Synchronize with the SVN Repository 924, 926, 927, 928, 929, 930, 931, 934
- commit changes 931
- integration with Bug Tracking tools 934
 - resolve conflicts 926, 927, 928, 929, 930
 - content conflicts vs property conflicts 926
 - drop incoming modifications 930
 - edit real content conflicts 927
 - merge conflicted resources 929
 - real conflicts vs mergeable conflicts 926
 - revert changes 928
 - tree conflicts 930
- update the working copy 930
- view differences 924
- Synchronize With The SVN Repository 924
- syntax highlight depending on namespace prefix 190

T

- TEI ODD document type 814, 815
- association rules 814
 - Author extensions 815
 - catalogs 815
- schema 815
- TEI ODD Document Type 559, 814, 815
 - Author extensions 559, 815
 - templates 815
 - transformation scenarios 559, 815
- TEI P4 document type 815
- association rules 815
 - Author extensions 815
 - catalogs 815
- schema 815
- TEI P4 Document Type 559, 815, 816
 - Author extensions 559, 815, 816
 - templates 816
 - transformation scenarios 559, 815
- TEI P5 document type 816
- association rules 816
 - Author extensions 816
 - catalogs 816
- schema 816

- TEI P5 Document Type 559, 816
 - Author extensions 559, 816
 - templates 816
 - transformation scenarios 559, 816
- Templates Tab (DITA OT transformations) 573
- Templates Tab (Document type configuration dialog box) 71
- Text Editing Mode 172, 223, 248, 264, 269, 316
 - compare file 264
 - insert file at cursor position 264
 - Manage highlighted content 269
 - open file at cursor position 264
 - print a file 316
 - shortcut actions 248
 - switch between opened tabs 223
- Text mode editor 172
- Text Mode Editor 172, 245
- Accessibility 172, 245
- breadcrumb 172, 245
- navigation 172, 245
- Toolbar subtab (Author Tab) (Document type configuration dialog box) 69
- Tools 839
- Tools Menu 269, 319, 415, 420, 428, 430, 431, 433, 456, 482, 505, 515, 516, 518, 519, 530, 840, 855, 860, 862, 863, 865, 866, 867, 876, 878, 879, 882, 884, 886, 888, 899, 902, 997
- Canonicalize 515, 876
- Compare Directories 530, 899
- Compare Files 519, 888
- Convert DB Structure to XML Schema 430, 862
- External Tools 997
- Flatten Schema 431, 863
- Format and Indent Files 269, 866
- Generate/Convert Schema 428, 860
 - Generate Documentation 420, 867
 - XML Schema Documentation 420, 867
- Generate Sample XML Files 415, 855
- Hex Viewer 886
- Large File Viewer 884
- Sign 516, 878
- SVG Viewer 505, 886
- SVN Client 902
- Tree Editor 888
- Verify Signature 518, 879
- WSDL SOAP Analyzer 456, 879
- XML Refactoring 319, 840
- XML Schema Regular Expression Builder 433, 882
- XML to JSON 482, 865
- Transformation Scenario 536, 560, 562, 566, 567, 568, 571, 579, 582, 583, 584, 588, 590, 591, 592, 595, 600
- ANT 579
 - built-in transformation scenarios 536
- DITA OT 571
 - new transformation scenario 560, 562, 567, 568, 583, 584, 591, 592
 - additional XSLT stylesheets 562, 584
 - configure transformation scenario 560
 - XML transformation with XSLT 560
 - XQuery extensions 568, 592
 - XQuery parameters 567, 591
 - XSLT extensions 562, 584
 - XSLT parameters 562, 583

- Transformation Scenario (*continued*)
 - sharing transformation scenarios 600
- SQL 595
- XML Transformation with XQuery 566
- XProc 588
- XQuery 590
- XSLT 582
- Transformation Tab (Document type configuration dialog box) 71
- Transforming Documents 535, 536, 601, 604, 606, 607
 - custom XSLT processors 606
 - supported XSLT processors 604
 - transformation scenario 536
- Transformation Scenarios view 601
- XSL-FO processors 607
- XSLT processors extensions paths 606
- Tweet Button 635, 657
- Twitter Widget 635, 657

U

- Unicode 82
- Uninstalling the application 49
- Upgrade 47
 - check for new version 47
- Upgrade HTTP floating license server 42
- upgrade WebHelp plugin for DITA 670
- upgrading WebHelp plugin for DocBook 675

V

- Validating Schematron Files 490
- Validating XML Documents 280
- Validating XML Schema Files 404, 441, 467, 477
- validation linked output messages 285
- Validation Scenario 292
 - sharing validation scenarios 292
- Validation Tab (Document type configuration dialog box) 72
- views to assist Content Completion 257

W

- WebDAV Connection 741
- WebDAV contextual menu actions 741
- WebHelp 669
 - command line 669
 - external process 669
 - outside oxygen 669
- WebHelp Classic Administrative page 651
- Admin Panel 651
- WebHelp Classic System 642
- WebHelp Classic with Feedback 651
- comment management 651
- manage comments 651
- WebHelp GET parameters 639, 665
- WebHelp Internationalization 637, 660
 - DITA WebHelp localization 637, 660
 - WebHelp i18n 637, 660
 - DocBook WebHelp localization 660
 - WebHelp i18n 660
- WebHelp Output 655
- WebHelp Skin Builder 655
- WebHelp plugin to run external DITA transformations 670

- WebHelp Responsive 611
- WebHelp Responsive Administrative page 618
- Admin Panel 618
- WebHelp Responsive Templates 619
- WebHelp Responsive Template types 619
- WebHelp Responsive with Feedback 613, 618
- comment management 618
- manage comments 618
- WebHelp with Classic Feedback System 645
- whitespace 82
- Whitespace handling 265
- Window Menu 150, 165, 190
- Export Layout 150
- Hide current view 150
- Load Layout 150
- Maximize/Restore Editor Area 150
- Next editor 150
- Open Perspective 165
- Previous editor 150
- Reset Layout 150
 - Results 190
 - Close 190
 - Close All Tabs 190
 - Close Other Tabs 190
 - Print Results 190
 - Save Results 190
- Show View 150
- Split Editor Horizontally 150
- Split Editor Vertically 150
- Stack Editors 150
- Synchronous Scrolling 150
- Tile Editors Horizontally 150
- Tile Editors Vertically 150
- Unsplit Editor 150
- Workspace Access 822
- Workspace Access JavaScript 823

X

- X-Hive/DB 731
- XHTML document type 814
- association rules 814
 - Author extensions 814
 - catalogs 814
 - schema 814
- XHTML Document Type 559, 814
 - Author extensions 559, 814
 - templates 814
 - transformation scenarios 559, 814
- XML Outline View 182, 183, 184, 262, 263
 - document structure change 184, 263
 - contextual menu 184, 263
- outline filters 183, 263
- XML Quick Fixes 318
- Accessibility 318
- XML Schema 198, 406
- Outline view 198, 406
- XML Schema Diagram Editor 196, 197, 200, 201, 202, 373, 374, 375, 383, 386, 388, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 407
- Attributes view 200, 407
- editing actions 375

- XML Schema Diagram Editor (*continued*)
- edit schema namespaces 402
- Facets view 201, 403
 - group schema components 400, 401
 - attributes 400
 - constraints 400
 - substitutions 401
- navigation 197, 374
 - schema components 383, 386, 388, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399
 - xs:alternative 391
 - xs:any 395
 - xs:anyAttribute 396
 - xs:assert 398
 - xs:attribute 386
 - xs:attributeGroup 388
 - xs:complexType 388
 - xs:element 383
 - xs:field 398
 - xs:group 392
 - xs:import 393
 - xs:include 393
 - xs:key 397
 - xs:keyRef 397
 - xs:notation 394
 - xs:openContent 399
 - xs:override 394
 - xs:redefine 393
 - xs:schema 383
 - xs:selector 398
 - xs:sequence, xs:choice, xs:all 394
 - xs:simpleType 390
 - xs:unique 396
- the Palette view 202, 402
- validation 401
- XML Schema Text Editor 404, 405, 431, 863
- content completion 405
- flatten an XML Schema 431, 863
- references to XML Schema specification 405
- XML serialization 265
- XProc Tab (XProc transformations) 589
- XProc Tab (XProc Transformations) 588
- XQJ Connection 691
- XQJ configuration 691
- XQJ Support 690
- XQJ processor configuration 690
- XQuery Tab (XQuery Transformations) 567, 591
- XSLT/XQuery Debugger 774, 775, 776, 777, 778, 779, 780, 781, 783, 784
- debug steps 783
- determining what XSLT/XQuery expression generated particular output 784
 - using breakpoints 784
 - inserting breakpoints 784
 - removing breakpoints 784
 - viewing processing information 774, 775, 776, 777, 778, 779, 780, 781
- breakpoints view 775
- context node view 774
- messages view 776
- node set view 781
- output mapping stack view 778

XSLT/XQuery Debugger (*continued*)
 viewing processing information (*continued*)
 stack view 777
 templates view 780
 trace history view 779

XSLT/XQuery Debugger (*continued*)
 viewing processing information (*continued*)
 variables view 781
 XPath watch view 775
XSLT tab (XSLT Transformations) 561, 582