



Oxygen JSON Editor 26.0

User Guide

Contents

Chapter 1. Introduction.....	12
Chapter 2. Getting Started.....	13
What is Oxygen JSON Editor.....	13
Getting Familiar with the Interface.....	14
Supported Document Types.....	15
Resources to Help You Get Started Using Oxygen JSON Editor	16
Getting Help.....	17
Help Menu.....	17
Frequently Used Shortcut Keys.....	20
Accessibility Support in Oxygen.....	25
Oxygen JSON Editor VPAT Accessibility Conformance Report.....	28
Chapter 3. Installation.....	53
Installing Oxygen JSON Editor on Windows.....	53
Installing Oxygen JSON Editor on macOS.....	58
Installing Oxygen JSON Editor on Linux.....	59
Installing Oxygen JSON Editor on Windows Server.....	64
Installing Oxygen JSON Editor on a Linux / UNIX Server.....	66
Site-Wide Deployment.....	68
Licensing.....	69
License Types.....	69
Upgrading.....	70
Upgrading Oxygen JSON Editor on Windows/Linux.....	71
Upgrading Oxygen JSON Editor on macOS.....	71
Privacy Options.....	71
Uninstalling.....	72
Chapter 4. Configuration.....	74
Preferences.....	74
Global Preferences.....	76
Appearance Preferences.....	79
Application Layout Preferences.....	85
Add-ons Preferences.....	86

Project Level Settings Preferences.....	86
Document Type Association Preferences.....	87
Document Templates Preferences.....	114
Encoding Preferences.....	114
Editor Preferences.....	115
CSS Validator Preferences.....	160
SVN Preferences.....	161
Diff Preferences.....	166
Archive Preferences.....	170
Plugins Preferences.....	171
External Tools Preferences.....	172
Menu Shortcut Keys Preferences.....	174
File Types Preferences.....	177
Open/Find Resource Preferences Page.....	178
Custom Editor Variables Preferences.....	180
Network Connection Settings Preferences.....	180
Views Preferences.....	185
Messages Preferences.....	185
Configuring Options.....	186
Storing Global and Project Level Options.....	187
Sharing Application Settings.....	188
Importing/Exporting/Resetting Global Options.....	189
Configuring the Layout of the Views and Editors.....	190
Configuring Toolbars.....	194
Import/Export Validation Scenarios.....	197
Editor Variables.....	197
Custom System Properties.....	205
Localizing the User Interface.....	210
Setting a Java Virtual Machine Parameter when Launching Oxygen JSON Editor.....	211
Setting Parameters for the Application Launchers.....	211
Setting Parameters in the Command-Line Scripts.....	214
Creating Custom Startup Parameters File.....	214
How to Increase the Amount of Available Memory.....	215

Chapter 5. Working With Documents.....	216
Getting Familiar with the Interface.....	216
Configuring the Layout of the Views and Editors.....	217
Configuring Toolbars.....	222
Creating, Opening, Saving, and Closing Documents.....	224
Creating New Documents and Templates.....	224
Opening Documents.....	234
Saving Documents.....	235
Auto Recover Documents.....	235
Closing Documents.....	236
Working with Remote Documents.....	236
Open URL.....	237
Changing File Permissions on a Remote FTP Server.....	240
WebDAV over HTTPS.....	240
HTTP Authentication Schemes.....	243
Switching, Moving, or Hiding Editor Tabs.....	243
Contextual Menu of the Current Editor Tab.....	245
Viewing File Properties.....	246
Simple Text Editor.....	247
Using Projects to Group Documents.....	248
Creating a New Project.....	248
Project View.....	251
Sharing a Project - Team Collaboration.....	261
Search and Find/Replace Features.....	263
Open/Find Resource View.....	264
Open/Find Resource Dialog Box.....	267
Searching in Content.....	270
Searching in File Paths.....	272
Searching in Reviews.....	272
Find/Replace Dialog Box.....	273
Find/Replace in Multiple Files.....	277
Find All Elements Dialog Box.....	283
Find and Invoke Actions.....	284

Quick Find Toolbar.....	286
Keyboard Shortcuts for Finding the Next and Previous Match.....	287
Regular Expressions Syntax.....	287
Spell Checking.....	288
Spell Check Dictionaries and Term Lists.....	290
Learned Words.....	296
Ignored Words (Elements).....	297
Automatic Spell Check.....	297
Spell Check Multiple Files.....	299
AutoCorrect Misspelled Words.....	300
Add Dictionaries for the AutoCorrect Feature.....	302
Working with Special Characters and Encoding.....	302
Unicode Support.....	303
Opening and Saving Documents with Unsupported Characters.....	304
Unicode Fallback Font Support.....	305
Inserting Special Characters with the Character Map.....	306
Loading Large Documents.....	308
Optimize Loading for Large Files.....	309
Optimize Loading for Huge Files.....	310
Documents with Long Lines.....	310
Handling Read-Only Files.....	311
Scratch Buffer.....	311
Compare Files or Directories	312
Compare Files Tool.....	313
Compare Directories Tool.....	332
Compare Directories Against a Base (3-Way) Tool.....	338
Generate HTML Report for Directory Comparison.....	345
Viewing Status Information.....	348
Editor Highlights.....	348
Printing a Document.....	349
Chapter 6. Editing Supported Document Types.....	352
Editing CSS Stylesheets.....	352
Validating CSS Stylesheets.....	352

Content Completion in CSS Stylesheets.....	353
Syntax Highlighting in CSS Files.....	354
CSS Outline View.....	354
Folding in CSS Stylesheets.....	355
Formatting and Indenting CSS Stylesheets (Pretty Print).....	355
Minifying CSS Stylesheets.....	355
Editing LESS Stylesheets.....	356
Validating LESS Stylesheets.....	357
Content Completion in LESS Stylesheets.....	357
Syntax Highlighting in LESS Files.....	357
Compiling LESS Stylesheets to CSS.....	358
Editing JSON Documents.....	358
JSON Editor.....	359
Navigating References in JSON Documents.....	360
Validating JSON Documents.....	363
Content Completion Assistant in JSON.....	371
Associating a Schema to JSON Documents.....	373
Syntax Highlighting in JSON Documents.....	378
Folding in JSON.....	378
JSON Outline View.....	378
JSON to XML Converter.....	381
XML to JSON Converter.....	384
JSON to YAML Converter.....	387
YAML to JSON Converter.....	387
Contextual Menu Actions in JSON Documents.....	388
Transforming and Querying JSON Documents.....	392
Editing JSON Schema Documents.....	395
JSON Schema Editor.....	395
JSON Schema Design Mode (JSON Schema Diagram Editor).....	396
Generating JSON Schema from a JSON File.....	417
Generating Sample JSON Files from a JSON Schema.....	419
JSON Schema Converter.....	420
Validating JSON Schema Documents.....	421

Syntax Highlighting in JSON Schema Documents.....	423
Flatten JSON Schema.....	423
Editing JSON Lines Documents.....	423
Editing YAML Documents.....	424
YAML Editor.....	424
Validating YAML Documents.....	425
Content Completion Assistant in YAML.....	429
Syntax Highlighting in YAML Documents.....	430
Folding in YAML Documents.....	430
Formatting/Indenting YAML Documents.....	430
YAML Outline View.....	431
YAML to JSON Converter.....	433
JSON to YAML Converter.....	433
Contextual Menu Actions in YAML Documents.....	434
Editing JavaScript Documents.....	439
JavaScript Editing Actions.....	439
Validating JavaScript Files.....	441
Content Completion in JavaScript Documents.....	441
Syntax Highlighting in JavaScript Documents.....	442
JavaScript Outline View.....	443
Editing HTML Documents.....	444
HTML Editor.....	444
HTML Validation.....	445
HTML Content Completion Assistant.....	446
Syntax Highlighting in HTML Documents.....	447
Folding in HTML.....	447
Minifying HTML Documents.....	447
HTML Outline View.....	448
Querying HTML Documents with XPath.....	448
Editing Markdown Documents.....	449
Markdown Editor.....	449
Creating New Markdown Documents.....	450
Actions Available in the Markdown Editor.....	450

Syntax Highlighting in the Markdown Editor.....	456
Markdown Editor Syntax Rules and Specifications.....	457
Chapter 7. Built-in Frameworks (Document Types).....	470
OpenAPI (Swagger) Document Type (Framework).....	470
OpenAPI Test Scenario Document Type (Framework).....	471
AsyncAPI Document Type (Framework).....	472
JSON-LD Document Type (Framework).....	473
Chapter 8. Working with XPath Expressions.....	474
XPath Builder View.....	474
XPath Expression Results View.....	477
XPath and XML Catalogs.....	478
Chapter 9. Working with Archives.....	480
Browsing Archives.....	480
Working with Archive Files.....	483
Creating an Archive.....	485
Migrating Archives to DITA or TEI.....	485
Chapter 10. Add-ons.....	486
Collaboration.....	486
Git Client Add-on.....	486
Migration/Conversions.....	500
Batch Documents Converter Add-on.....	500
Terminology.....	507
Terminology Checker Add-on.....	507
Vale Linter for Markdown and HTML Validation Add-on.....	516
Productivity.....	517
Oxygen AI Positron Assistant.....	517
Oxygen AI Positron Assistant Enterprise.....	528
Oxygen Emmet Plugin.....	532
Development.....	536
XSD to JSON Schema Converter.....	536
JSON Schema Documentation Generator.....	540
OpenAPI Tester.....	543
OpenAPI Documentation Generator.....	546

Chapter 11. Tools.....	551
JSON Tools.....	551
Generate Sample JSON Files.....	551
Generate JSON Schema.....	553
JSON to YAML.....	554
YAML to JSON.....	555
JSON to XML.....	556
XML to JSON.....	559
XSD to JSON Schema Converter.....	562
JSON Schema Converter.....	566
OpenAPI Tester.....	567
Run OpenAPI Test Scenario.....	570
Format and Indent Files.....	571
Generate Documentation.....	572
JSON Schema Documentation Generator.....	572
OpenAPI Documentation Generator.....	575
Large File Viewer.....	579
Hex Viewer.....	581
Compare Files.....	582
Starting File Comparison Tool from a Command Line.....	590
How to Integrate the File Comparison Tool with Git.....	591
How to Integrate the File Comparison Tool with Sourcetree.....	592
Toolbar and Contextual Menu Actions of the Compare Files Tool.....	594
Compare Files Tool Menus.....	597
Compare Directories.....	602
Toolbar and Contextual Menu Actions of the Compare Directories Tool.....	604
Compare Directories Tool Menus.....	605
Compare Images.....	607
Compare Directories Against a Base (3-Way).....	607
Generate HTML Report for Directory Comparison.....	615
External Tools.....	618
Chapter 12. Troubleshooting.....	620
Performance Problems and Solutions.....	620

Display Problems on Linux or Solaris.....	620
Too many nested apply-templates calls Error When Running a Transformation.....	620
Performance Issues with Large Documents.....	621
Performance Issues when Using Oxygen JSON Editor with Remote Desktop.....	621
Misc Problems and Solutions.....	622
Address Family Not Supported by Protocol Family.....	622
Application Reports Errors During Startup After Installing a New Version.....	622
Application Takes Several Minutes to Start.....	623
Archive Distribution Fails to Run on macOS 10.12 (Sierra).....	624
Blank Window is Shown When Starting the App Over an RDP Connection on Linux.....	624
Cannot Open Files from Desktop/Downloads/OneDrive on macOS.....	625
Cannot Uninstall Oxygen JSON Editor in Windows.....	625
Compatibility Issue Between Java and Certain Graphics Card Drivers.....	626
Crash at Startup on Windows with an Error About the nvoglv32.dll File.....	626
Damaged File Associations on macOS.....	627
Details to Submit in a Request for Technical Support Using the Online Form.....	627
Dialog Boxes Cannot Be Resized on Mac.....	629
Fonts Installed in Windows Do Not Appear in Fonts Preferences Page.....	629
Handshake Failure Error When Accessing an HTTPS (SSL) Resource.....	629
Hunspell Spell Checker is Unusable on Your Platform Error.....	630
High Resolution Scaling Issues.....	630
High Resolution Scaling Issues on Linux.....	631
Java Virtual Machine (JVM) Crash on macOS.....	631
Keyboard Language Resets to Default on Windows.....	632
Keyboard Shortcuts Do Not Work At All.....	632
Keyboard Shortcuts Result in Unexpected Behavior.....	632
Mac Touch Bar Function Keys Do Not Work.....	633
Server Signature Mismatch Error.....	633
Out Of Memory Error When Opening Large Documents.....	634
Scroll Function of my Notebook Trackpad is Not Working.....	635
Special Characters are Replaced with a Square.....	635
Text on Buttons and Labels is Invisible for Linux Installer.....	636
Text Rendering Issues on macOS.....	636

Chapter 13. Glossary.....	637
Active Cell.....	637
Alternate CSS Style.....	637
Apache Ant.....	637
Block Element.....	637
Bookmap.....	637
Canonicalize.....	637
Content Completion Assistant.....	638
Dockable.....	638
Document Type Association.....	638
DITA Map.....	638
Foldable Element.....	638
Framework.....	639
Global Options.....	639
IDML.....	639
Inline Element.....	639
Java Archive.....	639
Keystore.....	639
Main CSS Style.....	639
Plugin.....	640
Pretty-Print.....	640
Project Options.....	640
Quick Assist.....	640
Working Set.....	640
XML Catalog.....	640
Index.....	a
Copyright.....	p

1.

Introduction

Welcome to the User Manual of Oxygen JSON Editor 26.0.

Oxygen JSON Editor is a specialized tool designed for editing JSON documents. It offers a wide range of features and views, including Text, Grid, and Author editing modes, along with Design mode for JSON Schema. The intuitive interface and comprehensive set of tools make it easy to navigate, understand, and modify your JSON, JSON Schema files, as well as YAML and OpenAPI files.

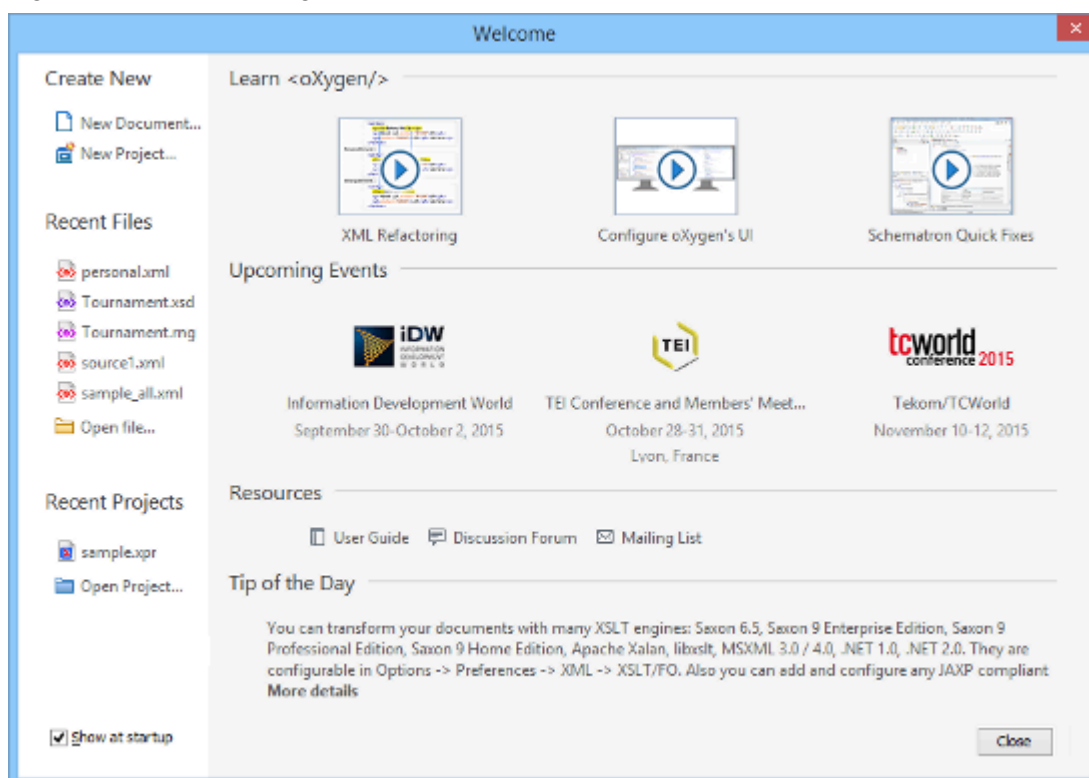
2.

Getting Started

This section provides a variety of resources to help you get the most out of the application. Typically, the first step of getting started with Oxygen JSON Editor would be to install the software. For detailed information about that process, see the [Installation chapter \(on page 53\)](#).

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

Figure 1. Welcome Dialog Box



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

What is Oxygen JSON Editor

Oxygen JSON Editor is a specialized tool designed for editing JSON documents. It offers a wide range of features and views, including Text, Grid, and Author editing modes, along with Design mode for JSON Schema. The intuitive interface and comprehensive set of tools make it easy to navigate, understand, and modify your JSON, JSON Schema files, as well as YAML and OpenAPI files. It also supports other technologies, including

HTML, CSS, LESS, Python, JavaScript, AsyncAPI, Shell, SQL, and more. You can even edit XML files in **Text** mode.

With powerful validation assistance, you can ensure that your JSON and YAML documents are well-formed and accurately validated against custom JSON Schema. Additionally, Oxygen JSON Editor provides convenient conversion tools to transform XML structures or YAML instances into valid JSON data, and vice versa.

For a list of many of the features and technologies that are included in Oxygen JSON Editor, see the [Oxygen JSON Editor website](#).

Getting Familiar with the Interface

Oxygen JSON Editor includes several editing modes (**Text**, **Grid**, **Author**, and **Design**) and a large variety of helper views, menu actions, toolbars, and contextual menu functions.

There are various ways that you can [configure the layout of the views or editors \(on page 217\)](#), and you can [customize the toolbars \(on page 222\)](#).

Regardless of the editing mode that you are working with, the default layout consists of the following areas:

Menus

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

Toolbars

Easy access to common and frequently used functions. Each icon is a button that acts as a shortcut to a related function. Some of the toolbars are common for all *perspectives*, editing modes, and types of documents, while others are specific to the particular *perspective* or mode. Some toolbars are also *framework*-specific, depending on the type of document that is being edited. All the [toolbars can be configured \(on page 222\)](#) to suit your specific needs.

Helper Views

Oxygen JSON Editor includes a large variety of [dockable \(on page 638\)](#) 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 \(on page 217\)](#) 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 various contextual menu actions and other features to help streamline your editing tasks. Each file that has been opened has a

tab at the top of the editing pane and there are [several ways to switch between tabs or move them \(on page 243\)](#).

Status Bar

The status bar at the bottom of the application contains some useful information when you are 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 \(on page 303\)](#) 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 \(on page 76\)](#) is selected, this section will show you when new messages have been received. The types of messages include the addition of new videos on the Oxygen JSON Editor website, the announcement of upcoming webinars and conferences where the Oxygen JSON Editor team will participate, and more.
- The memory consumption, including the memory used by the application and the maximum amount that is allocated to the application.
- If the [Show memory status option \(on page 78\)](#) is selected, a  **Free unused memory** icon is displayed in the bottom-right corner and you can use this icon to free up unused memory.

Supported Document Types

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

The supported document types include the following:

-  - JavaScript documents
-  - Python documents
-  - CSS documents
-  - LESS documents
-  - SQL documents
-  - JSON documents
-  - JSON Schema documents
-  - YAML documents
-  - Markdown documents
- Additional supported document types include: Text, Java, Properties, Batch, Shell, PowerShell, Dockerfile, and PHP.

Resources to Help You Get Started Using Oxygen JSON Editor

Configuring Oxygen JSON Editor

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

See the [Configuring Oxygen section \(on page 74\)](#) for details on the various ways that you can configure the application and its features.

Video Tutorials and Webinars

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

Go to the [Oxygen Videos page](#) to see the list of video tutorials.

Go to the [Oxygen Events page](#) to see all the upcoming and past webinars, conferences, and other events.

Oxygen JSON Editor Documentation

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

- [JSON Schema Diagram Editor \(on page 396\)](#) - Provides information about the schema design mode.
- [Editing Specific Document Types Chapter \(on page 352\)](#) - Includes information about editing numerous different types of documents.
- [Tools Chapter \(on page 551\)](#) - Details about the various built-in tools that are available in Oxygen JSON Editor.
- [Add-ons Chapter \(on page 486\)](#) - Information about how to extend the functionality of Oxygen JSON Editor through add-ons.

Sample Documents

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

- **Sample project file ([sample.xpr](#))** - 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 [view \(on page 251\)](#).
- **Sample files (*personal.json, etc.*)** - A collection of interrelated sample files that will allow you to experiment with the structure and relationship between files, stylesheets, and schemas.
- **Various document type folders** - The various folders contain sample files for numerous document types.

Other Resources

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

- See the [Oxygen JSON Editor Blog Site](#) for a large variety of current and archived blogs regarding numerous features, requests, and instructional topics.
- Take advantage of the [Oxygen JSON Editor Forum](#) to see various announcements and learn more about specific issues that other users have experienced.
- For more information about various additional tools that are integrated into Oxygen JSON Editor, see the [Tools section \(on page 551\)](#).
- See the [External Resource Page](#) for links to various other helpful resources, such as discussion lists, external tutorials, and more.
- For a list of new features that were implemented in the latest version of Oxygen JSON Editor, see the [What's New Section on the website](#).
- You can select the **Tip of the Day** ([on page 19](#)) action in the **Help** menu ([on page 17](#)) to display a dialog box that includes a variety of tips for using Oxygen JSON Editor.
- You can select **Show Dynamic Help view** ([on page 18](#)) from the **Help** menu ([on page 17](#)) to dynamically opens a topic that is relevant to the focused editor, view, or dialog box.

Getting Help

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

- [The Oxygen JSON Editor Support Section of the Website](#)
- [The Oxygen JSON Editor Forum](#)
- [The Oxygen JSON Editor Video Tutorials](#)
- [The Common Problems and Solutions Section of the User Manual \(on page 620\)](#)
- [The Online Technical Support Form](#)

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

Help Menu

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

This menu includes the following actions or options:

Welcome

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

Help (F1)

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

Show Dynamic Help view

Use this action to open a view that loads the latest online WebHelp version of the Oxygen JSON Editor User Manual, and dynamically opens a topic that is relevant to the focused editor, view, or dialog box.

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

Download User Manual

Opens the appropriate HTML page for downloading the Oxygen JSON Editor User Manual. This action is helpful if your internet access is restricted and you need to access the user guide.

Install new add-ons

Opens a dialog box that allows you to install new [add-ons \(on page 640\)](#) to extend the functionality of Oxygen JSON Editor.

Check for add-ons updates

Opens a dialog box that allows you to check for updates on installed [add-ons \(on page 640\)](#).

Manage add-ons

Opens a dialog box that allows you to manage installed [add-ons \(on page 640\)](#).

Check for a New Version

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

Check for notifications

Use this action to have the application check **Oxygen's** website for news and events that will be displayed in a dialog box.

Browse Oxygen Website

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

Register

If you encounter problems with your Oxygen JSON Editor 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 *Class Loader URLs* with your report.
- **System properties** - You can choose whether or not to include the listed system property details with your report.



Tip:

You are able to change the URL where the reported problem is sent by using the `com.oxygenxml.report.problems.url` system property. The report is sent in XML format through the `report` parameter of the POST HTTP method.

- **Plugins** - You can choose whether or not to include details about your installed *plugins* (on page 640) with your report.
- **Frameworks** - You can choose whether or not to include details about your installed *frameworks* (on page 639) with your report.

Support Center

Use this option to open the [Oxygen JSON Editor Support Section of the Website](#).

Support Tools > Clipboard Inspector

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

Support Tools > Randomize XML text content

Use this action when you need to send samples to the *Oxygen* support team and you want to keep the text content confidential. It opens a dialog box that allows you to select the resources that will have the text content randomized. You can then save the resources and send them to the *Oxygen* support team without fear of compromising sensitive or private data.



Warning:

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

Tip of the Day

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

About

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

- **Copyright** - This tab contains general information about the product and the version of the product you are using, along with contact details and the SGN number. Details regarding the memory usage are also presented at the bottom of the dialog box.
- **Libraries** - This tab presents the list of third-party libraries that Oxygen JSON Editor uses. To view the End-User Licence Agreement of each library, double-click it.
- **Frameworks** - This tab contains a list with the *frameworks (on page 639)* that are bundled with Oxygen JSON Editor.
- **System Properties** - This tab contains a list with system properties and their values. The contextual menu allows you to select and copy the properties.

Related information

[Details to Submit in a Request for Technical Support Using the Online Form \(on page 627\)](#)

Frequently Used Shortcut Keys

Oxygen JSON Editor includes numerous shortcut keys that are assigned to actions to help you edit content. All the shortcuts that are assigned to actions are displayed in the table in the [Menu Shortcut Keys preference page \(on page 174\)](#).

For information about how to assign or configure shortcut keys, see [How to Assign a Shortcut Key or Edit an Existing Shortcut \(on page 176\)](#).

Table 1. Frequently Used Shortcut Keys in Oxygen JSON Editor

Action	Windows/Linux Shortcut Keys	macOS Shortcut Keys	Description of Default Assigned Action
Attribute Editor	<u>Alt + Enter</u>	<u>Option + Enter</u>	Opens the in-place attribute editor
Beginning	<u>Ctrl + Home</u>	<u>Command + Home</u>	Navigates to the beginning of the document
Check Spelling	<u>F7</u>	<u>F7</u>	Opens the spell checking dialog box
Check Well-Formedness	<u>Ctrl + Shift + W</u>	<u>Command + Shift + W</u>	Check well-formedness of current document

Table 1. Frequently Used Shortcut Keys in Oxygen JSON Editor (continued)

Action	Windows/Linux Shortcut Keys	macOS Shortcut Keys	Description of Default Assigned Action
Content Completion / New Line	<u>Enter</u>	<u>Enter</u>	• Author mode - Opens the content completion window • Text mode - Moves cursor to the next line
Content Completion (Text Mode)	<u>Ctrl + Space</u>	<u>Command + Space</u>	Opens the content completion window in Text mode
Create Bookmark #	<u>Ctrl + Shift + 1-9</u>	<u>Command + Shift + 1-9</u>	Create bookmarks numbered 1 through 9
Create Next Bookmark	<u>F9</u>	<u>F9</u>	Create bookmark numbered whatever is next in sequence
Delete Next Word	<u>Ctrl + Delete</u>	<u>Command + Delete</u>	Deletes the next word or whitespace
Delete Previous Word	<u>Ctrl + Backspace</u>	<u>Command + Backspace</u>	Deletes the previous word or whitespace
Delete Tags	<u>Alt + Shift + X</u>	<u>Command + Option + X</u>	Deletes the start and end tag of the current element
Duplicate Lines Up (Text Mode)	<u>Ctrl + Shift + UpArrow</u>	<u>Option + Shift + UpArrow</u>	Duplicates the selected lines (or current line) and inserts it above the current selection/line
Duplicate Lines Down (Text Mode)	<u>Ctrl + Shift + DownArrow</u>	<u>Option + Shift + DownArrow</u>	Duplicates the selected lines (or current line) and inserts it below the current selection or line
End	<u>Ctrl + End</u>	<u>Command + End</u>	Navigates to the end of the document
Exit	<u>Ctrl + Q</u>	<u>Command + Q</u>	Exit the application
Find	<u>Ctrl + F</u>	<u>Command + F</u>	Opens Find/Replace dialog box

Table 1. Frequently Used Shortcut Keys in Oxygen JSON Editor (continued)

Action	Windows/Linux Shortcut Keys	macOS Shortcut Keys	Description of Default Assigned Action
Find Next	<u>F3</u>	<u>Command + G</u>	Finds next occurrence of the last searched term
Find Previous	<u>Shift + F3</u>	<u>Command + Shift + G</u>	Finds previous occurrence of the last searched term
Go To Bookmark	<u>Ctrl + 1-9</u>	<u>Command + 1-9</u>	Go to specific bookmark
Go To Definition	<u>Shift + Ctrl + Enter</u>	<u>Shift + Command + Enter</u>	Go to the definition of the selected item in the associated schema.
Help	<u>F1</u>	<u>F1</u>	Opens help documentation
Insert Para / Format Indent	<u>Ctrl + Shift + P</u>	<u>Command + Shift + P</u>	• Author mode - Inserts a paragraph at the cursor position • Text mode - Formats and indents current document
Move Tab Left	<u>Ctrl + Alt + Comma</u>	<u>Ctrl + Option + Comma</u>	Moves the current file tab one position to the left
Move Tab Right	<u>Ctrl + Alt + Period</u>	<u>Ctrl + Option + Period</u>	Moves the current file tab one position to the right
Move Node Down (Author)	<u>Alt + DownArrow</u>	<u>Option + DownArrow</u>	Moves the selected XML node down in Author mode
Move Node Down (Text)	<u>Ctrl + Alt + DownArrow</u>	<u>Command + Option + DownArrow</u>	Moves the selected XML node down in Text mode
Move Node Up (Author)	<u>Alt + UpArrow</u>	<u>Option + UpArrow</u>	Moves the selected XML node up in Author mode
Move Node Up (Text)	<u>Ctrl + Alt + UpArrow</u>	<u>Command + Option + UpArrow</u>	Moves the selected XML node up in Text mode
New File	<u>Ctrl + N</u>	<u>Command + N</u>	Opens wizard for creating new documents

Table 1. Frequently Used Shortcut Keys in Oxygen JSON Editor (continued)

Action	Windows/Linux Shortcut Keys	macOS Shortcut Keys	Description of Default Assigned Action
Next Word	<u>Ctrl + RightArrow</u>	<u>Command + RightArrow</u>	Navigates to next word
Open/Find Resource	<u>Ctrl + Shift + R</u>	<u>Command + Shift + R</u>	Opens the Open/Find Resource dialog box
Previous Word	<u>Ctrl + LeftArrow</u>	<u>Command + LeftArrow</u>	Navigates to previous word
Print Preview	<u>Ctrl + P</u>	<u>Command + P</u>	Opens the print preview (page setup) dialog box
Quick Assist	<u>Alt + 1</u>	<u>Command + Option + 1</u>	Opens Quick Assist menu if actions are available in the current context (usually indicated with a bulb  icon in the left stripe)
Quick Find	<u>Alt + Shift + F</u>	<u>Option + Shift + F</u>	Opens the Quick Find mechanism at the bottom of the editor
Redo	<u>Ctrl + Y</u> (Windows) - <u>Ctrl + Shift + Z</u> (Linux)	<u>Command + Shift + Z</u>	Redo last editing action
Refresh	<u>F5</u>	<u>F5</u>	Refresh
Remove Bookmarks	<u>Ctrl + F7</u>	<u>Command + F7</u>	Removes all bookmarks
Reopen Last Closed Editor	<u>Ctrl + Alt + T</u>	<u>Command + Option + T</u>	Reopens the editor tab that was closed most recently
Reset Zoom	<u>Ctrl + NumPad0</u>	<u>Command + NumPad0</u>	Resets zoom (default font size)
Save	<u>Ctrl + S</u>	<u>Command + S</u>	Saves current document
Save All	<u>Ctrl + Shift + S</u>	<u>Command + Shift + S</u>	Saves all open files
Scroll Down	<u>Ctrl + DownArrow</u>	<u>Command + DownArrow</u>	Scrolls the editor down
Scroll Up	<u>Ctrl + UpArrow</u>	<u>Command + Up Arrow</u>	Scrolls the editor up
Select Content of Element	<u>Alt + [Mouse Triple Click]</u>	<u>Option + [Mouse Triple Click]</u>	Selects the content of an element in Author mode.

Table 1. Frequently Used Shortcut Keys in Oxygen JSON Editor (continued)

Action	Windows/Linux Shortcut Keys	macOS Shortcut Keys	Description of Default Assigned Action
Shift Left	<u>Shift + Tab</u>	<u>Shift + Tab</u>	• Author mode - Moves the cursor to the previous XML node • Text mode - Shifts content to the left
Shift Right	<u>Tab</u>	<u>Tab</u>	• Author mode - Moves cursor to the next XML node • Text mode - Shifts content to the right
Split Element	<u>Alt + Shift + D</u>	<u>Ctrl + Option + D</u>	Splits the element the cursor position
Surround With	<u>Ctrl + E</u>	<u>Command + E</u>	Surrounds selected content with specified tag
Switch Tabs	<u>Ctrl + Tab / Ctrl + Shift + Tab</u>	<u>Command + Tab / Command + Shift + Tab</u>	Switches between open tabs
Transform	<u>Ctrl + Shift + T</u>	<u>Command + Shift + T</u>	Opens a dialog box for selecting a transformation scenario
Underline / Open URL	<u>Ctrl + U</u>	<u>Command + U</u>	• Underlines selected content (in the main editor) • Opens the URL (when focus is outside the main editor)
Undo	<u>Ctrl + Z</u>	<u>Command + Z</u>	Undo last editing action
Zoom In	<u>Ctrl + NumPad+</u>	<u>Command + NumPad+</u>	Zooms in (increase font size)
Zoom Out	<u>Ctrl + NumPad-</u>	<u>Command + NumPad-</u>	Zooms out (decrease font size)

**Troubleshooting:**

If you encounter problems with keyboard shortcuts not working as expected, see [Keyboard Shortcuts Result in Unexpected Behavior \(on page 632\)](#) or [Keyboard Shortcuts Do Not Work At All \(on page 632\)](#).

Accessibility Support in Oxygen

The **Oxygen** team is dedicated to developing software products that are usable for everyone, including those with physical challenges and disabilities. Oxygen JSON Editor is designed to adhere to the U.S. Government Section 508 accessibility standards: https://www.oxygenxml.com/xml_editor/section508.html.

Adjusting Fonts and Colors

If you have low vision, go to **Options > Preferences > Appearance > Fonts** where you can adjust the font styles and sizes used in the entire application, both for the editing areas and UI labels. If you have color blindness, you can also adjust most of the colors used in Oxygen JSON Editor by going to **Options > Preferences > Appearance** and changing the current color theme. You can also search for other color-related settings in the **Preferences** dialog box.

Installing Oxygen JSON Editor

Installation kits for Windows and Linux are made using the Install4j product. If you have problems navigating the Install4j installation wizard, you can run the installation from a [command-prompt application using the -c flag \(on page 56\)](#) like this:

```
C:\Users\your_user_name\Downloads\oxygenAuthor-64bit.exe -c
```

Screen Reader Software (Windows OS)

If you are using a text-to-speech narrator, Oxygen JSON Editor supports this since it is a Java application and it is periodically tested on Windows using both the NVDA and JAWS screen readers on the Windows operating system.

Using the JAWS Screen Reader (Windows)

The JAWS (Job Access With Speech) screen reader can be downloaded from: <http://www.freedomscientific.com/Products/Blindness/JAWS>.

For JAWS to work, you need to enable the Java access bridge in Oxygen JSON Editor: http://docs.oracle.com/javase/7/docs/technotes/guides/access/enable_and_test.html.

To enable the Java access bridge:

1. Since Oxygen JSON Editor comes bundled with its own Java VM, you need to open a command-prompt application and use the **cd** command to go to the Oxygen JSON Editor installation directory (for example, in Windows, it would be something like this:

```
cd C:\Program Files\Oxygen XML Editor 21.1
```

2. Then run the following command:

```
jre\bin\jabswitch -enable
```

3. Press **Enter** and you should receive a notification that the access bridge has been enabled.

Once the Java access bridge is enabled and as long as the JAWS narrator is active, when Oxygen JSON Editor starts, the narrator will start reading content from Oxygen JSON Editor and you can interact with the application and read menus, content from open XML documents, and UI components from dialog boxes and side views.

Using the NVDA Screen Reader (Windows)

The NVDA screen reader can be downloaded for free from: <https://www.nvaccess.org/>.

For NVDA to work, you need to enable the Java access bridge in Oxygen JSON Editor: http://docs.oracle.com/javase/7/docs/technotes/guides/access/enable_and_test.html.

To enable the Java access bridge:

1. Since Oxygen JSON Editor comes bundled with its own Java VM, you need to open a command-prompt application and use the **cd** command to go to the Oxygen JSON Editor installation directory (for example, in Windows, it would be something like this:

```
cd C:\Program Files\Oxygen XML Editor 21.1
```

2. Then run the following command:

```
jre\bin\jabswitch -enable
```

3. Press **Enter** and you should receive a notification that the access bridge has been enabled.

Once the Java access bridge is enabled and as long as the NVDA narrator is started, when Oxygen JSON Editor starts, the narrator will start reading content from Oxygen JSON Editor and you can interact with the application and read menus, content from open XML documents, and UI components from dialog boxes and side views.



Important:

If after these steps the narrator still does not read anything from a started Oxygen JSON Editor application, please go to the folder `C:\Windows\SysWOW64\` and make sure the library `WindowsAccessBridge-32.dll` is present there. If it is not present, try to search online, download the library file and copy it to the folder. Then restart Oxygen JSON Editor.

Besides the main editing area, Oxygen JSON Editor also has side views (for example, the **Attributes**, **Outline**, **Elements** views) that help with editing the XML content. NVDA versions **2020.1** and older have a [registered bug](#) that makes the narrator read content from the side views when editing in the main editing area. Because of this problem, when using NVDA versions **2020.1** or older, the following workflow is suggested:

1. Start Oxygen JSON Editor.
2. Go to the **Window** menu and select **Maximize Editing Area** (or **hold Alt, then W, then M**). This action will hide all side views and allow you to properly edit in the main editing area.

3. Whenever you want to open a side view, go to **Window > Show View** (or **hold Alt, then W, then S**) and choose the view you want to open. For example, to show the **Elements** view, you can **hold Alt, then W, then S, then E**.
4. When you are done using the side view, go to the **Window** menu and select **Hide current view** (or **hold Alt, then W, then H**) to hide the side view and return the focus to the main editing area.

Hints for the Visually Impaired

Here are a few hints for using Oxygen JSON Editor if you are visually impaired:

- The top main menu contains actions to open, save, and close documents, switch between open documents, or switch between the various editing modes for XML documents that are already open. All actions in the main menu bar should have mnemonics making it possible to memorize various shortcuts. For example, using the **alt-w-s-e** shortcut should open the **Window** menu, open the **Show view** submenu from it and show the **Elements** view,
 - The **File** menu contains actions to open, save, or close the currently edited XML document.
 - The **Edit** menu contains actions to undo/redo or cut/copy/paste content. They also have the usual shortcuts that can be used instead of directly invoking the actions from the menu.
 - The **Find** menu contains an action to show the **Find/Replace** dialog box. Sometimes the **JAWS** narrator overloads the **CTRL+F** shortcut and presents its own find/replace window but the Oxygen JSON Editor **Find/Replace** dialog box provides the ability to perform complex find/replace operations in the open file.
 - In the **Options** menu, you have access to the **Preferences** dialog box that contains global application settings and access to the **Menu Shortcut Keys** table where you can configure shortcuts for the most commonly used actions.
 - The **Window** menu includes actions to switch between open XML documents. Also, you can use the **Show view** submenu to open a particular side view and move the focus to that view.
- An open XML document can be edited with accessibility support either in the **Text** editing mode (where the XML tags are accessible in the edited content) or in the visual **Author** editing mode (where the XML tags are hidden and only the text content is shown). You can switch between these editing modes by using the **Document > Edit Mode** menu.
 - **Text** mode provides access to the entire source document with all of its **XML** content, just like you have in any text editing application.

Pressing the **⌘** key will present a list of [available XML elements \(on page 638\)](#). If you do not want to choose from the list whenever you want to insert an XML element, you have two choices:

- After the list of available XML elements is shown, you can press the **ESC** key to close it and continue to manually insert the XML tag.
- You can disable the content completion list from the **Options > Preferences > Editor / Content Completion** page by deselecting **Enable Content Completion**. After the content completion is disabled, you can force it to be displayed by using the **Ctrl+Space** keyboard shortcut.

In addition, using the **Window > Show view** submenu, you can change focus to the **Attributes**, **Elements**, or **Outline** view. The **Attributes** view presents the existing and possible attributes that can be inserted in an XML tag. The **Elements** view shows you the list of XML elements that can be inserted at the cursor position (also, pressing **F2** on a selected element presents its annotation). The **Outline** view shows the current path in the XML structure.

- **Author** mode is useful for reviewing written XML content because it has support for change tracking and for adding comments. Editing in the **Author** visual editing mode, you have access to only the text content in the XML document.

Pressing **Shift+F2** will read the current element context where the cursor is located. Pressing **Ctrl+Shift+F3** will read the current element context and the entire path in the XML structure where the cursor is located. You can also use the **Outline** view to better understand the XML structure.

In the **Author** editing mode, you can also use the **Attributes** and **Elements** views similar to using them in the **Text** editing mode. Pressing **Enter** in the **Author** visual editing mode can also be used to present a list of allowed elements at the current position.

Screen Reader Software (macOS)

On **macOS**, the application can be used with **VoiceOver** but is not rigorously tested with it. Because of processing limitations of the **VoiceOver** application, various limitations may be encountered. To avoid blocking the application, when tree-structure user interface controls are used (e.g. the **Project** or **Outline** view), if a node in the tree contains more than 500 child nodes, only the first 500 child nodes are accessible to the screen reader.

Oxygen JSON Editor VPAT Accessibility Conformance Report

A Voluntary Product Accessibility Template (VPAT) is a document that explains how information and communication technology (ICT) products such as software, hardware, electronic content, and support documentation meet (conform to) the [Revised 508 Standards](#) for IT accessibility. VPAT documents help Federal agency contracting officials and government buyers to assess ICT for accessibility when doing market research and evaluating proposals.

This document provides information about how Oxygen JSON Editor addresses the accessibility requirements defined in the international standards.

International Edition

VPAT[®] Version 2.3 – April 2019

Name of Product/Version

Oxygen JSON Editor 26.0

Product Description

Oxygen JSON Editor is a cross-platform application designed to accommodate all of your XML editing, authoring, developing, and publishing needs.

Date

March 2023

Contact Information

support@oxygenxml.com

Notes

Oxygen JSON Editor has been designed and enhanced to adhere to the [U.S. Government Section 508 accessibility standards](#) and the [Web Content Accessibility Guidelines \(WCAG\)](#). For details, see [Accessibility \(on page 25\)](#).

Evaluation Methods Used:

The following applications were used for testing **Oxygen JSON Editor**:

- NVDA assistive technology
- JAWS assistive technology

Applicable Standards/Guidelines

This report covers the degree of conformance for the following accessibility standards/guidelines:

Standard/Guideline	Included In Report
Web Content Accessibility Guidelines 2.0	Level A - Yes Level AA - Yes Level AAA - No
Web Content Accessibility Guidelines 2.1	Level A - Yes Level AA - Yes Level AAA - No
Revised Section 508 standards published January 18, 2017 and corrected January 22, 2018	Yes
EN 301 549 Accessibility requirements suitable for public procurement of ICT products and services in Europe - V2.1.2 (2018-08)	No

Terms

The terms used in the Conformance Level information are defined as follows:

- **Supports:** The functionality of the product has at least one method that meets the criterion without known defects or meets with equivalent facilitation.
- **Partially Supports:** Some functionality of the product does not meet the criterion.
- **Does Not Support:** The majority of product functionality does not meet the criterion.
- **Not Applicable:** The criterion is not relevant to the product.
- **Not Evaluated:** The product has not been evaluated against the criterion. This can be used only in WCAG 2.0 Level AAA.

WCAG 2.x Report

Tables 1 and 2 also document conformance with:

Revised Section 508: Chapter 5 – 501.1 Scope, 504.2 Content Creation or Editing, and Chapter 6 – 602.3 Electronic Support Documentation.



Note:

When reporting on conformance with the WCAG 2.x Success Criteria, they are scoped for full pages, complete processes, and accessibility-supported ways of using technology as documented in the [WCAG 2.0 Conformance Requirements](#).

Table 1: Success Criteria, Level A

Criteria	Conformance Level	Remarks and Explanations
<u>1.1.1 Non-text Content</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Text alternatives are provided for non-text content.
<u>1.2.1 Audio-only and Video-only (Prerecorded)</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Not Applicable	The product does not play audio and video content to end users.

Criteria	Conformance Level	Remarks and Explanations
<u>1.2.2 Captions (Prerecorded)</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Not Applicable	The product does not provide pre-recorded media that requires captions.
<u>1.2.3 Audio Description or Media Alternative (Prerecorded)</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Not Applicable	The product does not provide pre-recorded media that requires alternate descriptions.
<u>1.3.1 Info and Relationships</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text, with exceptions that include: • Grid editing mode. • Schema Design editing mode. • The editing of profiling attributes using the Edit profiling attributes and Insert/Edit Topic Reference dialog boxes. • The editing of a DITA map in the Author editing mode.
<u>1.3.2 Meaningful Sequence</u> (Level A) Also applies to: Revised Section 508	Supports	The product presents content in a meaningful sequence.

Criteria	Conformance Level	Remarks and Explanations
• 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		Authors should use Unicode right-to-left mark (RLM) or left-to-right mark (LRM) to mix text direction inline.
<u>1.3.3 Sensory Characteristics</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	The product provides textual identification for understanding and operating content.
<u>1.4.1 Use of Color</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.
<u>1.4.2 Audio Control</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Not Applicable	There is no sound that plays automatically by default.
<u>2.1.1 Keyboard</u> (Level A) Also applies to: Revised Section 508	Partially Supports	Most of the content is operable through a keyboard interface, with exceptions that include:

Criteria	Conformance Level	Remarks and Explanations
• 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		• Some toolbars in the application not being accessible via a keyboard. • Combo boxes and buttons located inside the table from the dialog box used to create or edit a validation scenario.
2.1.2 No Keyboard Trap (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	The product does not usually have user interface elements that trap the keyboard focus. Exceptions include some trees located in side views and dialog boxes.
2.1.4 Character Key Shortcuts (Level A 2.1 only) Also applies to: Revised Section 508 – Does not apply	Not Applicable	The product does not include character key shortcuts.
2.2.1 Timing Adjustable (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Not Applicable	The product does not include time limits.
2.2.2 Pause, Stop, Hide (Level A) Also applies to: Revised Section 508	Supports	Side views update their content automatically as a result of the user interaction with the open documents. They can be hidden or this functionality can be inhibited. The product does not in-

Criteria	Conformance Level	Remarks and Explanations
• 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		clude other elements that automatically move, blink, or scroll.
<u>2.3.1 Three Flashes or Below Threshold</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	The product does not have content that flashes more than three times in any one second.
<u>2.4.1 Bypass Blocks</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) – Does not apply to non-web software • 504.2 (Authoring Tool) • 602.3 (Support Docs) – Does not apply to non-web docs	Not Applicable	The application does not contain blocks of repeated content.
<u>2.4.2 Page Titled</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Each side view and dialog box in the application has a title.
<u>2.4.3 Focus Order</u> (Level A) Also applies to: Revised Section 508	Supports	Focusable components receive focus in an order that preserves meaning and operability.

Criteria	Conformance Level	Remarks and Explanations
• 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
<u>2.4.4 Link Purpose (In Context)</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	The purpose of each link can be determined from the link text alone or from the link text together with its programmatically-determined link context.
<u>2.5.1 Pointer Gestures</u> (Level A 2.1 only) Also applies to: Revised Section 508 – Does not apply	Not Applicable	The product does not have functionality that requires multi-point or path-based gestures.
<u>2.5.2 Pointer Cancellation</u> (Level A 2.1 only) Also applies to: Revised Section 508 – Does not apply	Partially Supports	Almost all pointer operations in the product are activated on Up events. Exceptions may include selection changes in the dialog box for new files and in the side views.
<u>2.5.3 Label in Name</u> (Level A 2.1 only) Also applies to: Revised Section 508 – Does not apply	Supports	The names of the user interface components contain the text that is presented visually.
<u>2.5.4 Motion Actuation</u> (Level A 2.1 only) Also applies to: Revised Section 508 – Does not apply	Not Applicable	The product does not contain functionality that can be operated by device or user motion.
<u>3.1.1 Language of Page</u> (Level A) Also applies to:	Does Not Support	The product does not report the default language for each open document.

Criteria	Conformance Level	Remarks and Explanations
Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
<u>3.2.1 On Focus</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	No changes of context occur when any component receives focus.
<u>3.2.2 On Input</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Changing the setting of any user interface component does not automatically cause a change of context.
<u>3.3.1 Error Identification</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Product shows error messages when user input is invalid.
<u>3.3.2 Labels or Instructions</u> (Level A) Also applies to: Revised Section 508	Partially Supports	Most input areas in the product provide labels and instructions. Exceptions include the content completion windows in the Text , Grid , Author , and schema Design modes.

Criteria	Conformance Level	Remarks and Explanations
• 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
<u>4.1.1 Parsing</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	All labels presenting HTML content in the product have valid HTML content.
<u>4.1.2 Name, Role, Value</u> (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	Almost all visual components in the product have identifiable names and roles with exceptions that include: • Grid editing mode. • Schema Design editing mode.

Table 2: Success Criteria, Level AA

Criteria	Conformance Level	Remarks and Explanations
<u>1.2.4 Captions (Live)</u> (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Not Applicable	The product does not contain live audio content.
<u>1.2.5 Audio Description (Prerecorded)</u> (Level AA) Also applies to:	Not Applicable	The product does not provide prerecorded video content that requires audio description.

Criteria	Conformance Level	Remarks and Explanations
Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
<u>1.3.4 Orientation</u> (Level AA 2.1 only) Also applies to: Revised Section 508 – Does not apply	Supports	Content does not restrict its view and operation to a single display orientation.
<u>1.3.5 Identify Input Purpose</u> (Level AA 2.1 only) Also applies to: Revised Section 508 – Does not apply	Not Applicable	The content does not contain input fields that collect information about the user.
<u>1.4.3 Contrast (Minimum)</u> (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	The product has sufficient contrast between the foreground and background colors, with few exceptions, including: • Change tracking content in the document for some of the author colors. • Change tracking content in some review panel items, when the item is selected. • Placeholders shown in empty elements. • Comments marked as done.
<u>1.4.4 Resize text</u> (Level AA) Also applies to: Revised Section 508	Partially Supports	In most situations, text in the main editing area, side views, and dialog boxes can be resized to reasonable dimensions by increasing the font without loss of content or functionality

Criteria	Conformance Level	Remarks and Explanations
• 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		and without using assistive technology. Sizes of certain text components cannot be increased.
<u>1.4.5 Images of Text</u> (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	The few buttons with icons containing text characters also provide text alternatives.
<u>1.4.10 Reflow</u> (Level AA 2.1 only) Also applies to: Revised Section 508 – Does not apply	Partially Supports	The majority of the user interface controls can be presented without loss of information or functionality, and without requiring scrolling in two dimensions. In the editor area, the text will re-flow depending on the editor mode (Text/Grid/Author/Design) and both horizontal and vertical scrolls may be needed.
<u>1.4.11 Non-text Contrast</u> (Level AA 2.1 only) Also applies to: Revised Section 508 – Does not apply	Supports	User interface components and states have sufficient contrast against adjacent colors.
<u>1.4.12 Text Spacing</u> (Level AA 2.1 only) Also applies to: Revised Section 508 – Does not apply	Supports	There is no loss of content or functionality occurs by setting line height (line spacing), spacing following paragraphs, letter spacing, and word spacing.
<u>1.4.13 Content on Hover or Focus</u> (Level AA 2.1 only)	Partially Supports	The tooltips for most user interface simple components (buttons) are not hoverable and persistent.

Criteria	Conformance Level	Remarks and Explanations
Also applies to: Revised Section 508 – Does not apply		
<u>2.4.5 Multiple Ways</u> (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) – Does not apply to non-web software • 504.2 (Authoring Tool) • 602.3 (Support Docs) – Does not apply to non-web docs	Supports	There are multiple ways to navigate between the open documents.
<u>2.4.6 Headings and Labels</u> (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Headings and labels describe the topic or purpose.
<u>2.4.7 Focus Visible</u> (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	The product has a visible indication of focus for almost all user interface controls (buttons, text fields, combo boxes, etc.) Exceptions include: • Combo boxes located in the DI-TA Maps Manager and Share-Point Browser view. • Filtering buttons located in the Templates tab from the New/Edit scenario and the DITA Reusable Components view. • Tab buttons located in the Insert/Edit Topic Reference dialog box.

Criteria	Conformance Level	Remarks and Explanations
<u>3.1.2 Language of Parts</u> (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Does Not Support	The language of parts is not specified.
<u>3.2.3 Consistent Navigation</u> (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) – Does not apply to non-web software • 504.2 (Authoring Tool) • 602.3 (Support Docs) – Does not apply to non-web docs	Supports	The product has a consistent navigation mechanism.
<u>3.2.4 Consistent Identification</u> (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) – Does not apply to non-web software • 504.2 (Authoring Tool) • 602.3 (Support Docs) – Does not apply to non-web docs	Supports	The product components are identified consistently.
<u>3.3.3 Error Suggestion</u> (Level AA) Also applies to: Revised Section 508	Supports	The product provides suggestions for the input error if there are any available.

Criteria	Conformance Level	Remarks and Explanations
• 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
<u>3.3.4 Error Prevention (Legal, Financial, Data)</u> (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Not Applicable	The product does not process legal commitments or financial transactions or modify user-controllable data.
<u>4.1.3 Status Messages</u>(Level AA 2.1 only) Also applies to: Revised Section 508 – Does not apply	Not Applicable	The product does not contain status messages as defined by this criterion.

Table 3: Success Criteria, Level AAA

Criteria	Conformance Level	Remarks and Explanations
<u>1.2.6 Sign Language (Prerecorded)</u> (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
<u>1.2.7 Extended Audio Description (Prerecorded)</u> (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
<u>1.2.8 Media Alternative (Prerecorded)</u> (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
<u>1.2.9 Audio-only (Live)</u> (Level AAA)	Not Evaluated	

Criteria	Conformance Level	Remarks and Explanations
Revised Section 508 – Does not apply		
<u>1.3.6 Identify Purpose</u> (Level AAA 2.1 only)	Not Evaluated	
Revised Section 508 – Does not apply		
<u>1.4.6 Contrast Enhanced</u> (Level AAA)	Not Evaluated	
Revised Section 508 – Does not apply		
<u>1.4.7 Low or No Background Audio</u> (Level AAA)	Not Evaluated	
Revised Section 508 – Does not apply		
<u>1.4.8 Visual Presentation</u> (Level AAA)	Not Evaluated	
Revised Section 508 – Does not apply		
<u>1.4.9 Images of Text (No Exception) Control</u> (Level AAA)	Not Evaluated	
Revised Section 508 – Does not apply		
<u>2.1.3 Keyboard (No Exception)</u> (Level AAA)	Not Evaluated	
Revised Section 508 – Does not apply		
<u>2.2.3 No Timing</u> (Level AAA)	Not Evaluated	
Revised Section 508 – Does not apply		
<u>2.2.4 Interruptions</u> (Level AAA)	Not Evaluated	
Revised Section 508 – Does not apply		
<u>2.2.5 Re-authenticating</u> (Level AAA)	Not Evaluated	
Revised Section 508 – Does not apply		

Criteria	Conformance Level	Remarks and Explanations
2.2.6 Timeouts (Level AAA 2.1 only) Revised Section 508 – Does not apply	Not Evaluated	
2.3.2 Three Flashes (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.3.3 Animation from Interactions (Level AAA 2.1 only) Revised Section 508 – Does not apply	Not Evaluated	
2.4.8 Location (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.4.9 Link Purpose (Link Only) (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.4.10 Section Headings (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.5.5 Target Size (Level AAA 2.1 only) Revised Section 508 – Does not apply	Not Evaluated	
2.5.6 Concurrent Input Mechanisms (Level AAA 2.1 only) Revised Section 508 – Does not apply	Not Evaluated	
3.1.3 Unusual Words (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
3.1.4 Abbreviations (Level AAA)	Not Evaluated	

Criteria	Conformance Level	Remarks and Explanations
Revised Section 508 – Does not apply		
3.1.5 Reading Level (Level AAA)	Not Evaluated	
Revised Section 508 – Does not apply		
3.1.6 Pronunciation (Level AAA)	Not Evaluated	
Revised Section 508 – Does not apply		
3.2.5 Change on Request (Level AAA)	Not Evaluated	
Revised Section 508 – Does not apply		
3.3.5 Help (Level AAA)	Not Evaluated	
Revised Section 508 – Does not apply		
3.3.6 Error Prevention (All) (Level AAA)	Not Evaluated	
Revised Section 508 – Does not apply		

Revised Section 508 Report

N/A

Chapter 3: Functional Performance Criteria (FPC)

Criteria	Conformance Level	Remarks and Explanations
302.1 Without Vision	Partially Supports	Much of the product is operable without vision. As noted in 1.3.1 Info and Relationships , some structural and hierarchical information is not communicated to screen readers. Also, as noted in 2.1.1 Keyboard , some components are not accessible via keyboard.
302.2 With Limited Vision	Supports	The product is operable with limited vision.

Criteria	Conformance Level	Remarks and Explanations
302.3 Without Perception of Color	Supports	Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.
302.4 Without Hearing	Supports	The product does not require hearing for use.
302.5 With Limited Hearing	Supports	The product does not require hearing for use.
302.6 Without Speech	Supports	The product does not require speech for use.
302.7 With Limited Manipulation	Partially Supports	Most content of the product is operable for users with limited manipulation who rely on keyboard access. As noted in 2.1.1 Keyboard , some components are not accessible via keyboard.
302.8 With Limited Reach and Strength	Supports	The product is functional with limited reach and limited strength. It supports operating system tools such as <i>Sticky-Keys</i> and <i>FilterKeys</i> .
302.9 With Limited Language, Cognitive, and Learning Abilities	Partially Supports	The product is operable by users with limited language, cognitive, and learning abilities. To accommodate users with limited cognition, the UI provides icons, text, or a combination of both, for controls. It provides the ability to change the interface language to 6 languages.

Chapter 4: Hardware

Notes: Not Applicable - **Oxygen JSON Editor** is not a hardware product.

Chapter 5: Software

501 General

Criteria	Conformance Level	Remarks and Explanations
501.1 Scope – Incorporation of WCAG 2.0 AA	See WCAG 2.x section (on page 30)	See information in WCAG section

502 Interoperability with Assistive Technology

Criteria	Conformance Level	Remarks and Explanations
502.2.1 User Control of Accessibility Features	Not Applicable	The product is not platform software.
502.2.2 No Disruption of Accessibility Features	Supports	The product does not disrupt platform features that are defined in the platform documentation as accessibility features.

502.3 Accessibility Services

Criteria	Conformance Level	Remarks and Explanations
502.3.1 Object Information	Partially Supports	Most of the object information can be programmatically determined. As noted in 1.3.1 Info and Relationships , the content from the Grid and schema Design editing modes is not exposed programmatically.
502.3.2 Modification of Object Information	Partially Supports	States and properties that can be set by the user can be set programmatically. As noted in 1.3.1 Info and Relationships , there are few exceptions.
502.3.3 Row, Column, and Headers	Partially Supports	The insert table feature in the editing area of the product does not communicate information about the headers. Row and column information is presented in a way that can be used by assistive technology.
502.3.4 Values	Partially Supports	The current values of an object can be programmatically determined. As noted in 1.3.1 Info and Relationships , there are few exceptions. The con-

Criteria	Conformance Level	Remarks and Explanations
		tent from the Grid and schema Design editing modes is not exposed programmatically.
502.3.5 Modification of Values	Partially Supports	Values that can be set by the user are capable of being set programmatically. As noted in 1.3.1 Info and Relationships , there are few exceptions. The content from the Grid and schema Design editing modes cannot be set programmatically.
502.3.6 Label Relationships	Partially Supports	Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. As noted in 1.3.1 Info and Relationships , there are few exceptions.
502.3.7 Hierarchical Relationships	Partially Supports	The hierarchical relationships are, in general, accessible programmatically, but as noted in 1.3.1 Info and Relationships , there are few exceptions.
502.3.8 Text	Partially Supports	The content of text objects, text attributes, and the boundary of text rendered to the screen is programmatically determinable. As noted in 1.3.1 Info and Relationships , there are few exceptions, including the text from the Grid and schema Design modes.
502.3.9 Modification of Text	Partially Supports	Text can be set programmatically, including through assistive technology. Text in the editor can be modified using the keyboard. As noted in 1.3.1 Info and Relationships , the text content from the Grid and schema Design editing modes cannot be accessed programmatically.

Criteria	Conformance Level	Remarks and Explanations
502.3.10 List of Actions	Supports	Actions that can be executed on an object can be determined programmatically from the context menu or content completion menu.
502.3.11 Actions on Objects	Supports	Actions on objects can performed by users, including those using assistive technologies.
502.3.12 Focus Cursor	Partially Supports	The focus location, selection state, and text insertion point information can be determined programmatically. As noted in 1.3.1 Info and Relationships , the content from the Grid and schema Design editing modes is not exposed programmatically.
502.3.13 Modification of Focus Cursor	Partially Supports	The focus location, selection state and text insertion can be controlled programmatically or through the keyboard. As noted in 1.3.1 Info and Relationships , the content from the Grid and schema Design editing modes cannot be accessed programmatically.
502.3.14 Event Notification	Supports	The changes in states and other properties are communicated through notifications that are communicated to assistive technology.
502.4 Platform Accessibility Features	Not Applicable	This product is not platform software.

503 Applications

Criteria	Conformance Level	Remarks and Explanations
503.2 User Preferences	Partially Supports	The product permits some of the user preferences from platform settings. Exceptions include

Criteria	Conformance Level	Remarks and Explanations
		• Cursor thickness is not modified in Text, Author, and Grid editing modes. • Some elements may not use the platform font size. The product provides custom preferences for changing the color, font type, and font size.
503.3 Alternative User Interfaces	Not Applicable	The application does not provide an alternative user interface that functions as assistive technology.

503.4 User Controls for Captions and Audio Description

Criteria	Conformance Level	Remarks and Explanations
503.4.1 Caption Controls	Not Applicable	The product does not provide controls for volume adjustment.
503.4.2 Audio Description Controls	Not Applicable	The product does not provide controls for program selection.

504 Authoring Tools

Criteria	Conformance Level	Remarks and Explanations
504.2 Content Creation or Editing (if not authoring tool, enter “not applicable”)	See the WCAG 2.x section (on page 30)	See information in WCAG section
504.2.1 Preservation of Information Provided for Accessibility in Format Conversion	Partially Supports	For the main XML vocabulary supported in the application (DITA), the accessibility information is preserved in the generated main formats (Web-Help, PDF).
504.2.2 PDF Export	Supports	The product is capable of publishing PDF files that conform to PDF/UA-1.
504.3 Prompts	Partially Supports	For DITA documents, there is an optional Schematron that performs accessibility checks on the content and

Criteria	Conformance Level	Remarks and Explanations
		prompts the authors whenever it detects errors.
504.4 Templates	Does Not Support	The product does provide several templates. However, these templates offer only minimal structure and do not mark content in ways that promote following the WCAG success criteria. The existing templates can be customized.

Chapter 6: Support Documentation and Services

601.1 Scope

602 Support Documentation

Criteria	Conformance Level	Remarks and Explanations
602.2 Accessibility and Compatibility Features	Partially Supports	The documentation of the product lists and explains the accessibility and compatibility features of the product.
602.3 Electronic Support Documentation	Partially Supports	The self-service documentation is generated with Oxygen XML Web-Help . You can find its VPAT statement here .
602.4 Alternate Formats for Non-Electronic Support Documentation	Not Applicable	Documentation is not provided in non-electronic formats.

603 Support Services

Criteria	Conformance Level	Remarks and Explanations
603.2 Information on Accessibility and Compatibility Features	Supports	The support services cover the accessibility features.
603.3 Accommodation of Communication Needs	Supports	Support is provided over a variety of channels including email and phone.

Legal Disclaimer

This report describes **Oxygen JSON Editor** ability to support the stated VPAT Standards/Guidelines, subject to Syncro Soft's interpretation of the same. This accessibility report is provided for informational purposes only, and the contents hereof are subject to change without notice. SYNCRO SOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT. For more information regarding the accessibility status, please contact us at sales@oxygenxml.com.

© 2020 Syncro Soft SRL. All rights reserved.

3.

Installation

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

Choosing How Oxygen JSON Editor Runs

You can install Oxygen JSON Editor to run in several ways:

- As a desktop application (running standalone or as an Eclipse plugin) on Windows, Linux, or macOS.
- 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 also have a choice of several different installers:

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

Choosing a License Option

You must [obtain and register a license \(on page 69\)](#) to run Oxygen JSON Editor.

Upgrading, Transferring, and Uninstalling.

You can also [upgrade \(on page 70\)](#) Oxygen JSON Editor or [uninstall \(on page 72\)](#) Oxygen JSON Editor.

Getting Help With Installation

If you need help, email support at: support@oxygenxml.com.

Installing Oxygen JSON Editor on Windows

System Requirements

Operating Systems

The product has been fully tested on Windows versions 10 and 11. The latest version of Oxygen JSON Editor might work on other versions of Windows, but they have not been officially tested.

CPU

- Minimum - Intel/AMD Dual-core class CPU, 2 GHz
- Recommended - Quad-core class processor

Memory

- Minimum - 3 GB of RAM
- Recommended - 8 GB of RAM

Storage

- Minimum - 1 GB free disk space
- Recommended - 2 GB free disk space

Java

Oxygen JSON Editor only officially supports Java Virtual Machines with version 17 from Oracle or Eclipse Adoptium. If you use the native Windows installer, Oxygen JSON Editor will be installed with its own copy of Java with the specific update version that has been thoroughly tested.

All Platforms Package

If you use the all platforms package, your system must have a compatible Java 17 virtual machine installed. To see the exact Java update version that is supported, go to www.oxygenxml.com, navigate to the **Download** page for the particular product you are installing, and click on the tab for your particular platform.



Note:

Oxygen JSON Editor may work with other versions of Java, but since Oxygen JSON Editor has only been thoroughly tested with specific versions, there is no guarantee that it will be stable with any other Java version.

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

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

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

Windows Installer

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

1. Make sure that your system meets the [system requirements \(on page 53\)](#).
2. [Download](#) the Windows installer.
3. [Optional] 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 JSON Editor using one of the following methods:
 - Use one of the shortcuts created by the installer.
 - Run `oxygenJSONEditor.bat`, which is located in the installation directory.
6. To license your copy of Oxygen JSON Editor, go to **Help > Register** and enter your [license information \(on page 69\)](#).

Windows 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 JSON Editor with the default options and does not overwrite existing files. You can change various options for the unattended installer using the installer command-line parameters.

Windows Installer Command-Line Reference

The Oxygen JSON Editor installer for Windows supports a variety of command-line parameters:

-q

Instructs the installer to run in unattended mode. The installer will not prompt the user for input during the install. Default settings will be used for all options unless a `response.varfile` ([on page 57](#)) is specified using the `-varfile` option.

-overwrite

In unattended mode, the installer does not overwrite files with the same name if a previous version of the Oxygen JSON Editor is installed in the same folder. The **-overwrite** parameter added after the **-q** parameter forces the overwriting of these files.

-console

Displays a console during an unattended installation.

**Note:**

If you want the installer to run in the foreground, you need to use the **start /wait** command (for example, `start /wait oxygen.exe -q -console`). Otherwise, it will run in the background.

-varfile

Specifies the location of a **response.varfile** ([on page 57](#)), normally to be used during an unattended installation.

-c

Allows users to configure the installation by inputting answers to installation questions in the command line.

**Tip:**

Using this parameter is the best way to use the installer for people who are visually impaired.

-V[variable name]=[variable value]

This command-line parameter can be used to define any of the variables listed below to be used by an installation.

EXAMPLE:

```
oxygen.exe -q -overwrite -console -VautoVersionChecking=false
```

Command-Line Variables for Preconfiguring License Server Details

For organizations that use a license server to manage user licenses, the Oxygen JSON Editor installer also supports the following command-line variables used for preconfiguring license server details:

autoVersionChecking

Used for automatic version checking. Possible values are **true** (default) or **false**.

backup.license.servlet.url

Specifies the URL of the backup HTTP license server.

backup.license.servlet.user.name

Specifies the user name for the backup HTTP license server.

backup.license.servlet.password

Specifies the password for the backup HTTP license server, in clear form (will be stored encrypted).

backup.license.servlet.password.encrypted

Specifies the password for the HTTP license server, in encrypted form. Can be obtained from an entry with the same name in an existing `license.xml` file (found in: `[user_home_directory]\AppData\Roaming\com.oxygenxml.jsoneditor`).

downloadResources

Used to download resources (links to video demonstrations, webinars, and upcoming events) from <https://www.oxygenxml.com> to populate the application welcome screen. Possible values are **true** (default) or **false**.

license.servlet.url

Specifies the URL of the HTTP license server.

license.servlet.user.name

Specifies the user name for the HTTP license server.

license.servlet.password

Specifies the password for the HTTP license server, in clear form (will be stored encrypted).

license.servlet.password.encrypted

Specifies the password for the HTTP license server, in encrypted form. Can be obtained from an entry with the same name in an existing `license.xml` file (found in: `[user_home_directory]\AppData\Roaming\com.oxygenxml.jsoneditor`).

reportProblem

Used to report a problem encountered while using Oxygen JSON Editor. Possible values are **true** (default) or **false**.

EXAMPLE:

```
oxygen.exe "-Vlicense.servlet.url=http://
main.licenseserver:8080/oxygenLicenseServlet/license-servlet"
"-Vlicense.servlet.user.name=user" "-Vlicense.servlet.password=mypass"
"-Vbackup.license.servlet.url=http://
backup.licenseserver:8080/oxygenLicenseServlet/license-servlet"
"-Vbackup.license.servlet.user.name=user" "-Vbackup.license.servlet.password=mypass"
```

Windows Installer *response.varfile*

The Oxygen JSON Editor installer for Windows also creates a file called `response.varfile`, which records the choices that the user made when running the installer interactively. The generated response file is found in the `[OXYGEN_INSTALL_DIR]/.install4j` folder. You can use the `response.varfile` to set the options for an **unintended install** (*on page 55*). For more information about the `response.varfile` format, see [install4j site](#).

Variables (can be used in the `response.varfile` or from the command line)

The following variables are supported in the `response.varfile` (or from the command line):

autoVersionChecking

Used for automatic version checking. Possible values are **true** (default) or **false**.

downloadResources

Used to download resources (links to video demonstrations, webinars, and upcoming events) from <https://www.oxygenxml.com> to populate the application welcome screen. Possible values are **true** (default) or **false**.

reportProblem

Used to report a problem encountered while using Oxygen JSON Editor. Possible values are **true** (default) or **false**.

Installing Oxygen JSON Editor on macOS

System Requirements

Operating system

The product has been fully tested on macOS 11 (Big Sur), 12 (Monterey), 13 (Ventura), and 14 (Sonoma). The latest version of Oxygen JSON Editor might work on older versions of macOS, but they have not been officially tested.

CPU

- Minimum - Intel-based Dual-core Mac, 2 GHz
- Recommended - Apple M1 or newer

Memory

- Minimum - 4 GB of RAM
- Recommended - 8 GB of RAM

Storage

- Minimum - 1 GB free disk space
- Recommended - 2 GB free disk space

macOS Installation

To install Oxygen JSON Editor on macOS, follow these steps:

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

**Warning:**

If you receive a warning that an *Oxygen JSON Editor* installation folder already exists in the **Applications** folder, do not attempt to merge the two installations. Instead, move the old installation folder to the trash bin before installing the application. If you are prompted to **Replace** the old folder, cancel the installation, move the old folder to the trash bin, and restart the installation process.

**Important:**

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

5. Start Oxygen JSON Editor, using one of the following methods:
 - Double-click **Oxygen JSON Editor.app**.
 - Run `sh oxygenJSONEditor.sh` in the command-line interface.
6. To license your copy of Oxygen JSON Editor, go to **Help > Register** to enter your [license key \(on page 69\)](#).

macOS Unattended Installation

To install Oxygen JSON Editor on macOS in unattended mode, follow these steps:

1. [Download](#) the macOS installation package (oxygenJSONEditor.dmg).
2. Mount the oxygenJSONEditor.dmg file in the command line.
3. Copy the **oxygen** folder for the particular version from the mounted volume to the **Applications** folder (or another folder where you want to install it), as in the following example:

```
hdiutil attach oxygen.dmg|oxygenAuthor.dmg|oxygenDeveloper.dmg
```

```
cp -aR "/Volumes/Oxygen JSON Editor 26.0/Oxygen JSON Editor" /Applications/|
```

4. Eject the mounted disc image:

```
hdiutil detach "/Volumes/Oxygen JSON Editor 26.0"|
```

Installing Oxygen JSON Editor on Linux

System Requirements

Operating System

The product has been fully tested on Ubuntu 22.04. The latest version of Oxygen JSON Editor might work on other flavors/versions of Linux, but they have not been officially tested.

CPU

- Minimum - Intel/AMD Dual-core class CPU, 2 GHz
- Recommended - Quad-core class processor

Memory

- Minimum - 3 GB of RAM
- Recommended - 8 GB of RAM

Storage

- Minimum - 1 GB free disk space
- Recommended - 2 GB free disk space

Java

Oxygen JSON Editor only officially supports Java Virtual Machines with version 17 from Oracle or Eclipse Adoptium. If you use the Linux installer, Oxygen JSON Editor will be installed with its own copy of Java with the specific update version that has been thoroughly tested.

All Platforms Package

If you use the all platforms package, your system must have a compatible Java 17 virtual machine installed. To see the exact Java update version that is supported, go to www.oxygenxml.com, navigate to the **Download** page for the particular product you are installing, and click on the tab for your particular platform.



Note:

Oxygen JSON Editor may work with other versions of Java, but since Oxygen JSON Editor has only been thoroughly tested with specific versions, there is no guarantee that it will be stable with any other Java version.



Attention:

Oxygen JSON Editor does not work with the **GNU libgcj** Java Virtual Machine.

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

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

You can also change the version of the Java Virtual Machine that runs Oxygen JSON Editor by editing the script file, `oxygenJSONEditor.sh`.

X.org

The version of Java bundled with Oxygen JSON Editor requires **X.org** (Wayland is not supported).

Linux Installer

To install Oxygen JSON Editor using the Linux installer, follow these steps:

1. Make sure that your system meets the [system requirements \(on page 59\)](#).
2. [Download](#) the Linux installer.
3. [Optional] 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.



Note:

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.



Warning:

If you are running the installer as root and your Linux distribution uses Wayland (such as **Ubuntu 17.10** or **Fedora 25**), before running the installer, the local user must first allow the root user to access the X server by running the following command (as the local user):

```
xhost +SI:localuser:root
```

5. Start Oxygen JSON Editor using one of the following methods:

- Use the `jsoneditor` shortcut created by the installer.

**Note:**

For **Ubuntu 17.10** (or later), a security dialog box will appear the first time you start the application where you need to select **Trust and Launch** to continue. This dialog box will not appear on subsequent launches.

- From a command line, type `sh oxygenJSONEditor.sh`. This file is located in the installation folder.
6. To license your copy of Oxygen JSON Editor go to **Help > Register** and enter your [license information \(on page 69\)](#).

Linux 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 JSON Editor with the default options and does not overwrite existing files. You can change various options for the unattended installer using the installer command-line parameters.

Linux Installer Command-Line Reference

The Oxygen JSON Editor installer for Linux supports a variety of command-line parameters:

-q

Instructs the installer to run in unattended mode. The installer will not prompt the user for input during the install. Default settings will be used for all options unless a `response.varfile` ([on page 64](#)) is specified using the `-varfile` option.

-overwrite

In unattended mode, the installer does not overwrite files with the same name if a previous version of the Oxygen JSON Editor is installed in the same folder. The `-overwrite` parameter added after the `-q` parameter forces the overwriting of these files.

-console

Displays a console during the installation.

-varfile

Specifies the location of a `response.varfile` ([on page 64](#)), normally to be used during an unattended installation.

-V

Used to define a `variable parameter` ([on page 64](#)) to be used by an installation.

EXAMPLE:

```
oxygen.sh -q -overwrite -console -VautoVersionChecking=false
```

Command-Line Variables for Preconfiguring License Server Details

For organizations that use a license server to manage user licenses, the Oxygen JSON Editor installer also supports the following command-line variables used for preconfiguring license server details:

autoVersionChecking

Used for automatic version checking. Possible values are **true** (default) or **false**.

backup.license.servlet.url

Specifies the URL of the backup HTTP license server.

backup.license.servlet.user.name

Specifies the user name for the backup HTTP license server.

backup.license.servlet.password

Specifies the password for the backup HTTP license server, in clear form (will be stored encrypted).

backup.license.servlet.password.encrypted

Specifies the password for the HTTP license server, in encrypted form. Can be obtained from an entry with the same name in an existing `license.xml` file (found in: `[user_home_directory]\AppData\Roaming\com.oxygenxml.jsoneditor`).

downloadResources

Used to download resources (links to video demonstrations, webinars, and upcoming events) from <https://www.oxygenxml.com> to populate the application welcome screen. Possible values are **true** (default) or **false**.

license.servlet.url

Specifies the URL of the HTTP license server.

license.servlet.user.name

Specifies the user name for the HTTP license server.

license.servlet.password

Specifies the password for the HTTP license server, in clear form (will be stored encrypted).

license.servlet.password.encrypted

Specifies the password for the HTTP license server, in encrypted form. Can be obtained from an entry with the same name in an existing `license.xml` file (found in: `[user_home_directory]\AppData\Roaming\com.oxygenxml.jsoneeditor`).

reportProblem

Used to report a problem encountered while using Oxygen JSON Editor. Possible values are **true** (default) or **false**.

EXAMPLE:

```
oxygen.sh "-Vlicense.servlet.url=http://
main.licenseserver:8080/oXygenLicenseServlet/license-servlet"
"-Vlicense.servlet.user.name=user" "-Vlicense.servlet.password=mypass"
"-Vbackup.license.servlet.url=http://
backup.licenseserver:8080/oXygenLicenseServlet/license-servlet"
"-Vbackup.license.servlet.user.name=user" "-Vbackup.license.servlet.password=mypass"
```

Linux Installer *response.varfile*

The Oxygen JSON Editor installer for Linux also creates a file called **response.varfile**, which records the choices that the user made when running the installer interactively. The generated response file is found in the `[OXYGEN_INSTALL_DIR]/.install4j` folder. You can use the **response.varfile** to set the options for an **unintended install** (*on page 62*). For more information about the **response.varfile** format, see [install4j site](#).

Variable Parameters (can be used in the *response.varfile* or from the command line)

The following variable parameters are supported in the **response.varfile** (or from the command line):

autoVersionChecking

Used for automatic version checking. Possible values are **true** (default) or **false**.

reportProblem

Used to report a problem encountered while using Oxygen JSON Editor. Possible values are **true** (default) or **false**.

downloadResources

Used to download resources (links to video demonstrations, webinars, and upcoming events) from <https://www.oxygenxml.com> to populate the application welcome screen. Possible values are **true** (default) or **false**.

Installing Oxygen JSON Editor on Windows Server

System Requirements

Operating systems

Windows Server 2012 or Windows Server 2012 R2

CPU

- Minimum - Intel/AMD Dual-core class CPU, 2 GHz
- Recommended - Quad-core class processor

Memory

- Minimum values per user - 1 GB of RAM
- Recommended values per concurrent user - 2 GB of RAM

Storage

- Minimum - 1 GB free disk space
- Recommended - 2 GB free disk space

Java

Oxygen JSON Editor only officially supports Java Virtual Machines with version 17 from Oracle or Eclipse Adoptium. If you use the native Windows installer, Oxygen JSON Editor will be installed with its own copy of Java with the specific update version that has been thoroughly tested.

All Platforms Package

If you use the all platforms package, your system must have a compatible Java 17 virtual machine installed. To see the exact Java update version that is supported, go to www.oxygenxml.com, navigate to the **Download** page for the particular product you are installing, and click on the tab for your particular platform.



Note:

Oxygen JSON Editor may work with other versions of Java, but since Oxygen JSON Editor has only been thoroughly tested with specific versions, there is no guarantee that it will be stable with any other Java version.

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

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

If you run Oxygen JSON Editor using the batch file, **oxygenJSONEditor.bat**, you can edit the batch file to specify a particular version to use.

Windows Installer

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

1. Make sure that your system meets the [system requirements \(on page 64\)](#).
2. [Download](#) the Windows installer.
3. [Optional] 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 JSON Editor using one of the following methods:
 - Use one of the shortcuts created by the installer.
 - Run `oxygenJSONEditor.bat`, which is located in the installation directory.
6. To license your copy of Oxygen JSON Editor go to **Help > Register** and enter your [license information \(on page 69\)](#).

Configuring Windows Terminal Server

1. Install Oxygen JSON Editor on the server and make its shortcuts available to all users.
2. Make sure you allocate sufficient memory to Oxygen JSON Editor by adding the `-Xmx` parameter either in the `.bat` startup script [\(on page 211\)](#), or in the `.vmoptions` configuration file [\(on page 214\)](#) (if you start it from an executable launcher).

Installing Oxygen JSON Editor on a Linux / UNIX Server

System Requirements

Operating system

The product has been fully tested on Ubuntu 22.04. The latest version of Oxygen JSON Editor might work on other flavors/versions of Linux, but they have not been officially tested.

CPU

- Minimum - Intel/AMD Dual-core class CPU, 2 GHz
- Recommended - Quad-core class processor

Memory

- Minimum - 3 GB of RAM
- Recommended - 8 GB of RAM

Storage

- Minimum - 1 GB free disk space
- Recommended - 2 GB free disk space

Java

Oxygen JSON Editor only officially supports Java Virtual Machines with version 17 from Oracle or Eclipse Adoptium. If you use the Linux installer, Oxygen JSON Editor will be installed with its own copy of Java with the specific update version that has been thoroughly tested.

All Platforms Package

If you use the all platforms package, your system must have a compatible Java 17 virtual machine installed. To see the exact Java update version that is supported, go to www.oxygenxml.com, navigate to the **Download** page for the particular product you are installing, and click on the tab for your particular platform.



Note:

Oxygen JSON Editor may work with other versions of Java, but since Oxygen JSON Editor has only been thoroughly tested with specific versions, there is no guarantee that it will be stable with any other Java version.



Attention:

Oxygen JSON Editor does not work with the **GNU libgcj** Java Virtual Machine.

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

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

You can also change the version of the Java Virtual Machine that runs Oxygen JSON Editor by editing the script file, `oxygenJSONEditor.sh`.

Linux Installer

To install Oxygen JSON Editor using the Linux installer, follow these steps:

1. Make sure that your system meets the [system requirements \(on page 66\)](#).
2. [Download](#) the Linux installer.
3. [Optional] 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.

**Note:**

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.

5. Start Oxygen JSON Editor using one of the following methods:
 - Use the `jsoneditor` shortcut created by the installer.
 - From a command line, type `sh oxygenJSONEditor.sh`. This file is located in the installation folder.
6. To license your copy of Oxygen JSON Editor go to **Help > Register** and enter your [license information \(on page 69\)](#).

Unix/Linux Server Configuration

1. Install Oxygen JSON Editor on the server and make sure the `oxygenJSONEditor.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 JSON Editor by setting an appropriate value for the `-Xmx` parameter in the `.sh` startup script.

**Note:**

The default value of the `-Xmx` parameter is about a quarter of the maximum internal memory available on the machine. To avoid [performance issues with large documents \(on page 621\)](#), 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 JSON Editor by typing `sh oxygenJSONEditor.sh` from the command line. This file is located in the installation folder.

Site-Wide Deployment

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

Creating default project files

Oxygen JSON Editor project files [\(on page 248\)](#) are used to configure a project. You can [create and deploy default project files \(on page 248\)](#) 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 \(on page 261\)](#) 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 \(on page 55\)](#) or [Linux \(on page 62\)](#) installs.

Licensing

This section contains information about licensing Oxygen JSON Editor.

Registering License Keys

To help you comply with the **Oxygen** EULA (terms of licensing), all floating or named-user licenses must be registered. 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 deployment of the license server is sent to the **Oxygen** servers. The servers will then sign the license key.

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

- You can purchase one or more licenses from the Oxygen JSON Editor website at or through one of the [authorized resellers](#). A license key will be sent to you by email.
- If your company or organization has already purchased licenses, contact your license administrator to obtain a license key or configuration details to connect to a license server.
- If you want to evaluate the product, you can obtain a trial license key for 30 days from the Oxygen JSON Editor website at .

License Types

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

The following license types are available:

- A **Subscription** license that allows you to use the application for a specific period of time (either 6 months or 1 year). This type of license is user-based and is covered by a Support and Maintenance Pack, which means that during the subscription period you will get free upgrades to all major and minor releases and priority technical support.

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

Subscription Licenses

A **Subscription** license that allows you to use the application for a specific period of time (either 6 months or 1 year). This type of license is user-based and is covered by a Support and Maintenance Pack, which means that during the subscription period you will get free upgrades to all major and minor releases and priority technical support.

Registering a Subscription License

To register a *Subscription License*, follow these steps:

1. Purchase a license from the [Oxygen JSON Editor website](#). You will receive an email that contains your license key.
2. Save a backup copy of your email message that contains the new license key.
3. Start Oxygen JSON Editor.
If this is a new installation of Oxygen JSON Editor, the registration dialog box is displayed. If the registration dialog box is not displayed, go to **Help > Register**.
4. Paste your license key into the registration dialog box. The license key is composed of nine lines of text between two text markers.
5. Click **OK**.

Related information

[Oxygen JSON Editor End-User License Agreement](#)

Automatic Subscription Renewal

The **Oxygen License Server** has a mechanism that tries to detect when you purchase a renewal of your current subscription and automatically updates the license key.

To determine that a license key you purchased is a renewal of the license key you have currently installed in the License Server, it uses the **Previous order reference number** if you inserted it in the checkout process.

This automatic renewal mechanism makes an HTTP request to `https://oxygenxml.com/subscription_management/check_renewal.php` and passes the following as parameters:

- The SGN field of the existing license key.
- The server signature that uniquely identifies the License Server installation.

This request is made by the License Server automatically (or manually by pressing the **Check now** link).

This mechanism can be disabled by deselecting the **Automatically check for subscription renewal** checkbox.

Upgrading

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

By default, Oxygen JSON Editor automatically checks for new versions at startup (or every 24 hours from startup). If a newer version is detected, a dialog box will automatically be displayed that provides information about the type of upgrade or update that is available. If a new patch of a new version is detected, the dialog box will be displayed only when the new patch version includes a critical bug fix or when the license allows upgrading to the new version.

To disable this check, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#), go to **Global**, and deselect **Automatic Version Checking**.

To check for new versions manually, go to **Help > Check for New Versions**. This opens a dialog box that displays information about whether or not a newer version is available.

Upgrading Oxygen JSON Editor on Windows/Linux

How to Upgrade Oxygen JSON Editor on Windows or Linux

1. Download and install the new version according to the instructions for your platform and the type of installer you selected.
2. Restart Oxygen JSON Editor.

Upgrading Oxygen JSON Editor on macOS

How to Upgrade Oxygen JSON Editor on macOS

1. [Uninstall the current version of Oxygen JSON Editor \(on page 72\)](#) or rename the installation directory.
2. Download and install the new version in an empty folder according to the instructions for your platform and the type of installer you selected.
3. Restart Oxygen JSON Editor.

Privacy Options

As an on-premise desktop application, Oxygen JSON Editor does not store sensitive user data or user-specific files on remote servers. The full [Privacy Policy](#) is available on the official Oxygen JSON Editor web site.

Whenever it is started, the application may connect to its official web site (<https://www.oxygenxml.com>) to provide notifications about new releases or new events. In the **Global** preferences page, there are some checkboxes that can be disabled if you do not want this type of information retrieved:

Automatic Version Checking

If this option is selected, the application obtains information about new available versions from the official Oxygen JSON Editor web site. It provides the license category (academic/professional/enterprise), the product version, Java version, and operating system name.

Check for Oxygen-related events at startup

If this option is selected, the application obtains information about events that get displayed in the **Welcome** screen. The types of downloaded events include information about the addition of new videos on the website, announcements of upcoming webinars and conferences where the Oxygen JSON Editor team will participate, and more. No specific information about the current installation is passed to the Oxygen JSON Editor web site during this process.

Check for Oxygen-related notifications

If this option is selected, the application obtains information about other various notifications that get displayed [on the right side of the status bar \(on page 217\)](#). No specific information about the current installation is passed to the Oxygen JSON Editor web site during this process.

Providing installation-specific details to the Oxygen JSON Editor technical support team is also possible by using the **Help > Report problem** action from the main menu. Also, if an *unhandled/fatal* error occurs in the application, the user is presented with a **Report problem** dialog box and can choose to submit the error and its context for analysis. The **Report problem** dialog box shows the exact specific details that are sent to the Oxygen JSON Editor technical support team and can be examined by the end user before being sent. No files or confidential information are sent.

When the application is installed on Windows or Linux using the provided installation kit, the last installation step allows you to click a **Privacy Options** link to choose which privacy-related settings should be disabled before the application starts for the first time. The **Privacy Options** can also be disabled when performing an unattended install. See instructions here: [Windows Unattended Installation \(on page 55\)](#).

Uninstalling

How to Uninstall Oxygen JSON Editor



CAUTION:

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

1. Back up any data you want to keep from the Oxygen JSON Editor installation folder.
2. Remove the application according to your operating system:
 - **Windows or Linux** - Use the appropriate uninstaller shortcut provided with your OS.
 - **macOS** - Manually delete the installation folder and all its contents.
3. If you want to remove the user preferences:
 - **Windows** - Remove the directory: `%APPDATA%\com.oxygenxml.jsoneditor`. Note that the `AppData` directory is hidden. If you cannot locate it, type `%APPDATA%` and press ENTER in the File Explorer address bar. (`%APPDATA%` expands to `[user-home-dir]\AppData\Roaming`).
 - **macOS** - Remove the directory: `Library/Preferences/com.oxygenxml.jsoneditor` of the user home folder.
 - **On Linux**, remove the directory: `.com.oxygenxml.jsoneditor` from the user home directory.

Unattended Uninstall

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

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

- **Windows** - The uninstaller executable is called **uninstall.exe** and is located in the *Oxygen* installation directory.
- **Linux** - The uninstaller executable is called **uninstall** and is located in the *Oxygen* installation directory.

4.

Configuring Oxygen JSON Editor

This chapter presents all the user preferences and options that allow you to configure various features and aspects of the application itself. It also includes information about storing and sharing options, importing and exporting options or scenarios, customizing system properties, setting startup parameters, and the [editor variables \(on page 197\)](#) that are available for customizing user-defined commands.

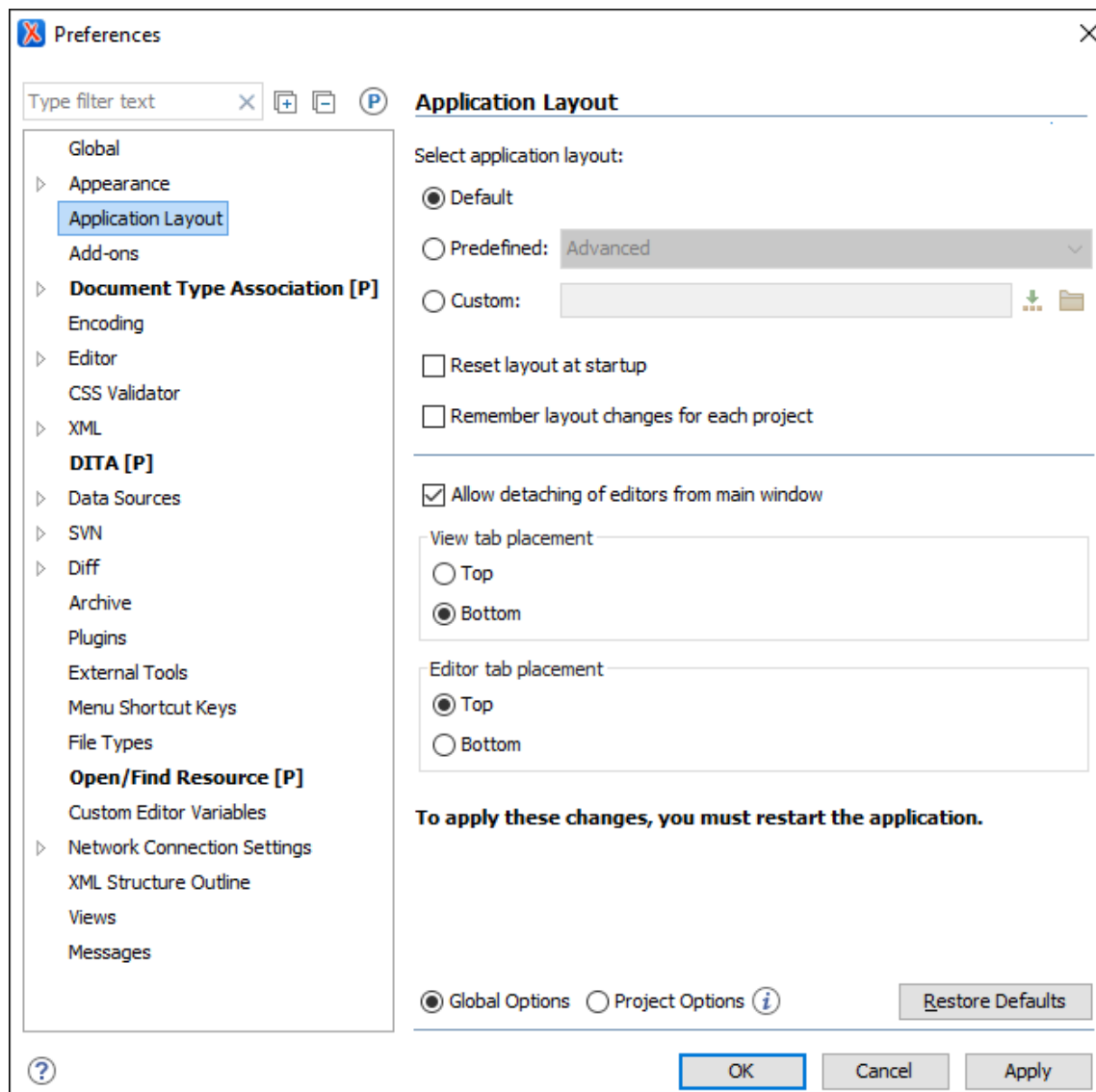
Preferences

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

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

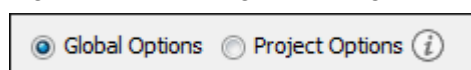
You can select the preference page you are interested in from the tree on the left of the **Preferences** dialog box. You can filter the tree by using the filter text box and the following buttons are available to the right of the text box:

-  **Expand All** - Expands the structure of the tree to show all preference pages.
-  **Collapse All** - Collapses the structure of the tree to show only the 1st level preference pages.
-  **Project-Level Options Only** - If toggled on, it filters the tree to only show the preference pages that are [saved at project level \(on page 188\)](#).

Figure 2. Preferences Dialog Box

Click the  icon or press **F1** for help on any preferences page.

Some preference pages include an option to control how the options are stored, either as **Global Options** ([on page 188](#)) or **Project Options** ([on page 188](#)).

Figure 3. Controlling the Storage of the Preferences

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

Preferences Directory Location

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

- **Windows (7, 8, 10)** - `[user_home_directory]\AppData\Roaming\com.oxygenxml.jsoneditor`
- **macOS** - `[user_home_directory]/Library/Preferences/com.oxygenxml.jsoneditor`
- **Linux/Unix** - `[user_home_directory]/.com.oxygenxml.jsoneditor`

Global Preferences

The global options cover various aspects of the overall operation of Oxygen JSON Editor. To configure the **Global** options, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Global**.

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

Automatic Version Checking

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

Check for Oxygen-related events at startup

If this option is selected, Oxygen JSON Editor will check for various new event updates on the Oxygen JSON Editor website and if any new events are found, they will be presented at startup.

Check for notifications

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

Language

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

Other language

This option sets the language used in the user interface using an interface localization file. For details about creating this file, see [Localizing the User Interface \(on page 210\)](#). You can use this option to set the language of the user interface to a language that is not shipped with Oxygen JSON Editor.



Note:

If some interface labels are not rendered correctly after restarting the application, (for example, 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 specifies the type of line separator to be used when saving files. Use **System Default** to select the normal line separator for your OS. The other two possible selections are **Unix-like** and **Windows-like**.



Notes:

- This option is ignored if the **Detect the line separator on file open** option ([on page 77](#)) is selected AND a line separator is automatically detected.
- When changing the selection in this option, the change does not affect an opened file until you make a modification to the file and save it. At that point, all line separators in the file will change to the type of line separator you chose in this option.

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. If this option is not selected, you can use the **Line separator** option ([on page 76](#)) to choose the type of line separator to be used when saving files.



Tip:

To see the line separator type for the current file, you can use the **Properties** view (**Window > Show View > Properties**).

Default Internet browser

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

- Open (X)HTML or PDF transformation results.
- Open a web page.

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

Open last edited files from project

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

Load file content only when switching to its corresponding editor tab

When selected (default), files that were left open in the previous editing session remain as placeholder tabs but the file content is loaded only when switching to the corresponding editor tab. This helps to improve performance. If the option is deselected, the previously open files are all re-loaded at startup.

Check opened files for file system changes

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

Auto update unmodified editors on file system changes

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

Beep on operation finished

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

Show memory status

When this option is selected, the memory that Oxygen JSON Editor uses is displayed in the status bar. To free memory, click the  **Free unused memory** button located at the right side of the status bar. The memory status bar turns yellow or red when Oxygen JSON Editor uses too much memory. You can change the amount of memory available to Oxygen JSON Editor by [changing the parameters of the application launcher \(on page 211\)](#).

Order of switching between editor tabs

This option specifies the order for [switching between open file tabs when using **Ctrl + Tab** \(**Command + Tab** on macOS\) or **Ctrl + Shift + Tab** \(**Command + Shift + Tab** on macOS\)](#) [\(on page 243\)](#). You can choose between:

- **Recently used order** - Switches to the most recently used tab.
- **Visual order** - Switches to the next tab in visual order.

File Chooser Dialog section

Use platform file chooser (Windows and macOS)

This option is selected by default and it specifies that the native file chooser is used. You can deselect this option if you want the Java Swing file chooser to be used instead. If Oxygen JSON Editor encounters a problem while using the native file manager, it will avoid using it again in the current session, even if this option is selected.

Consider application bundles to be directories when browsing (macOS only)

This option is available only on the macOS 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 macOS).

Show hidden files and directories

If this option is selected, Oxygen JSON Editor shows system hidden files and folders in the file browser dialog box and the folder browser dialog box.



Tip:

On macOS, you need to press **Command + Shift + Period** in the file browser to show hidden files.

File chooser opens

This option specifies the starting directory that the [file browser dialog box \(on page 234\)](#) will open. You can choose between:

- **Directory of the selected file** - The file browser opens the folder where the selected file is stored, depending on the current selection (for example, a file could be selected from the **Project** view, , main editing pane, or another location within the application).
- **Last visited directory** - The file browser opens the last visited folder.

Appearance Preferences

This preferences page contains various options that allow you to change the appearance of the user interface of Oxygen JSON Editor. To configure the **Appearance** options, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) 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 macOS)



Note:

In Windows, if a high contrast theme is detected and the **Theme** option is set to *Classic* and the [Look and Feel option \(on page 79\)](#) is set to *Default* or *Windows*, Oxygen JSON Editor inherits the high contrast theme colors that are set in the operating system.

- **Graphite**

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 JSON Editor. To create a custom theme, follow these steps:

1. Select a **Theme** to use as a base.
2. Configure the desired options in any of the option pages listed in this preferences page.
3. Click **Export** and specify a name for your custom theme. If you save the theme to the default file path, your custom theme will immediately appear in the **Theme** drop-down list. Otherwise, if you save it to another location, you can use the **Import button** ([on page 80](#)) to make it appear in the drop-down list.



Note:

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

Theme preview area

Displays a preview of the current **Theme selection** ([on page 79](#)) (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](#) ([on page 80](#)).

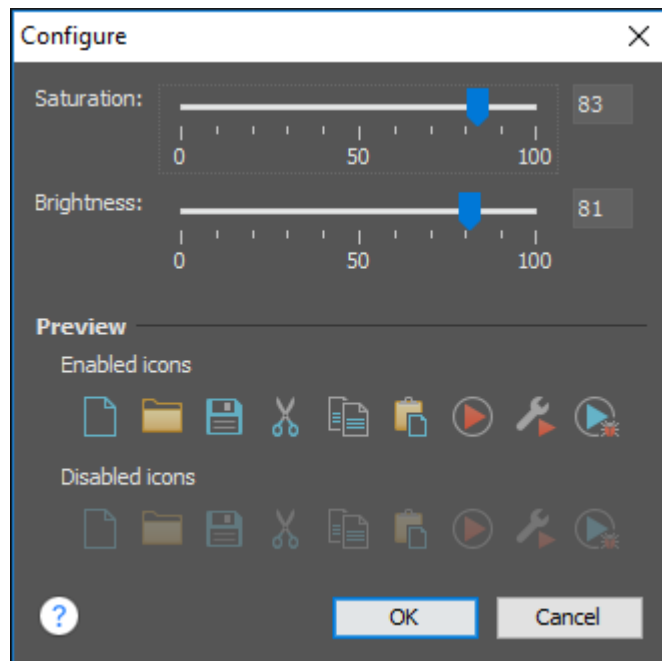
Export

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 JSON Editor.

Configure icon saturation and brightness link

This link is available if you are using the **Graphite theme** ([on page 79](#)). It opens a dialog box where you can configure the saturation and brightness for all the icons in Oxygen JSON Editor.

Figure 4. Configure Icon Saturation and Brightness Dialog Box



Colors Preferences

Oxygen JSON Editor allows you to configure the colors for frames, dialog boxes, controls, and commands. To configure the **Colors**, open the **Preferences** dialog box (**Options > Preferences**) ([on page 74](#)) 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 macOS) Background color for menu selections and toolbar buttons.

View titles background

Background color for the titles of view and tabs.

Status bar background

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

Foreground Colors

Foreground

Foreground color for various general user interface items.

Component selection foreground

Foreground color for the current selection.

Disabled foreground

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

Link foreground

Foreground color for links in views and dialog boxes.

View titles foreground

Foreground color for the title bar of views.

Status bar foreground

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

Other Colors

Borders and table grids

Color for certain borders and table grid lines.

Text component border

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

View/Editor tabs border

Color for the borders of views and tabs.

Scroll bars, chevrons

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

Separator

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



Note:

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

Fonts Preferences

Oxygen JSON Editor allows you to choose the fonts to be used in the **Text**, **Design**, and **Grid** editor modes, and fonts for the **Author** mode that are not specified in the associated CSS stylesheet. To configure the font options, open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Appearance > Fonts**.

The following options are available:

Editor

Specifies the font family, size, and weight to be used in the **Text** mode editor. To change the current values, double-click the text field or click the **Choose** button. This opens a dialog box where you can choose the font family, font size, and whether or not to bold the text. You can enter or paste content in the **Sample** box to see a preview of how it will look in the application. If you select the **Show only the fonts that can render the sample text** option and paste content in the **Sample** box, the application detects fonts that can render the particular character set and filters the fonts that can be selected accordingly.



Note:

On macOS, the default font, Monaco, cannot be rendered in bold.

Author default font

Specifies the default font family, size, and weight to be used in **Author** mode. However, the default font will be overridden by the fonts specified in any CSS file associated with the open document. To change the current values, double-click the text field or click the **Choose** button. This opens a dialog box where you can choose the font family, font size, and whether or not to bold the text. You can enter or paste content in the **Sample** box to see a preview of how it will look in the application. If you select the **Show only the fonts that can render the sample text** option and paste content in the **Sample** box, the application detects fonts that can render the particular character set and filters the fonts that can be selected accordingly.

Schema default font

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

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

To change the current values, double-click the text field or click the **Choose** button. This opens a dialog box where you can choose the font family, font size, and whether or not to bold the text. You can enter or paste content in the **Sample** box to see a preview of how it will look in the application. If you select the **Show only the fonts that can render the sample text** option and paste content in the **Sample** box, the application detects fonts that can render the particular character set and filters the fonts that can be selected accordingly.

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

Specifies the font family, size, and weight to be used in text boxes within the interface. To change the current values, double-click the text field or click the **Choose** button. This opens a dialog box where you can choose the font family, font size, and whether or not to bold the text. You can enter or paste content in the **Sample** box to see a preview of how it will look in the application. If you select the **Show only the fonts that can render the sample text** option and paste content in the **Sample** box, the application detects fonts that can render the particular character set and filters the fonts that can be selected accordingly.

GUI

Specifies the font family, size, and weight to be used for user interface labels. To change the current values, double-click the text field or click the **Choose** button. This opens a dialog box where you can choose the font family, font size, and whether or not to bold the text. You can enter or paste content in the **Sample** box to see a preview of how it will look in the application. If you select the **Show only the fonts that can render the sample text** option and paste content in the **Sample** box, the application detects fonts that can render the particular character set and filters the fonts that can be selected accordingly.

View titles font

Specifies the font family, size, and weight to be used in the titles of the various views within the interface. To change the current values, double-click the text field or click the **Choose** button. This opens a dialog box where you can choose the font family, font size, and whether or not to bold the text. You can enter or paste content in the **Sample** box to see a preview of how it will look in the application. If you select the **Show only the fonts that can render the sample text**

option and paste content in the **Sample** box, the application detects fonts that can render the particular character set and filters the fonts that can be selected accordingly.

**Note:**

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

Application Layout Preferences

Oxygen JSON Editor offers various 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 \(Options > Preferences\) \(on page 74\)](#) 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. 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.

Custom

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

Reset layout at startup

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

Remember layout changes for each project

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

Allow detaching of editors from main window

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

**Note:**

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

View tab placement

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

Editor tab placement

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

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

Resources

For more information about configuring the user interface of Oxygen JSON Editor, watch our video demonstration:

<https://www.youtube.com/embed/anwjepfAdEk>

Add-ons Preferences

You can use [add-ons \(on page 640\)](#) to enhance the functionality of Oxygen JSON Editor. To configure the **Add-ons** options, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Add-ons**.

The following options are available in this preferences page:

Enable automatic updates checking

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

Add-on Sites URLs

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

Automatically install add-ons

You can use this section to specify required add-ons for a project. Then, when a user opens the project, the specified add-ons will be automatically installed after prompting the user. You can use **Ctrl+Space** to open a pop-up window with the list of detected add-ons that you can select to be marked for automatic installation. You can also manually enter the add-on ID and multiple IDs can be entered by separating with either a space or new line.

Project Level Settings Preferences

The **Project Level Settings** preference page allows you to decide whether various settings should be saved in the project configuration file or in the global settings. Settings that are saved at project level can easily be shared with others. To configure these options, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Project Level Settings**.

The following options can be toggled on or off to determine which settings will be saved at project level:

Allow validation scenario associations to be saved at project level

When this option is selected, the associations for custom validation scenarios will be stored according to their storage location.

- If you associate a scenario that is stored at the project or framework level, the association will be saved at project level (in the project configuration file).
- If you associate a scenario at global level, it will be saved globally.

If this option is not selected, the association is not allowed to be saved at project level and will be saved globally (even if the scenario is saved in the project file).

Document Type Association Preferences

Oxygen JSON Editor uses *document type associations* (on page 638) to associate a *document type* (on page 470) with a set of functionality provided by a *framework* (on page 639). To configure the **Document Type Association** options, open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Document Type Association**.

The following actions are available in this preferences page:

Discover more frameworks by using add-ons update sites

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

Document Type Table

This table presents the currently defined *frameworks* (on page 639) (*document type associations* (on page 638)), sorted by priority and alphabetically. Each edited document type has a *set of association rules* (on page 90) (used by the application to detect the proper document type association to use for an open XML document).

New

Opens a **Document type configuration dialog box** (on page 89) that allows you to add a new *framework*.

Edit

Opens a **Document type configuration dialog box** (on page 89) that allows you to edit an existing *framework*.

**Note:**

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

Duplicate

Opens a **Document type configuration dialog box** (on page 89) that allows you to duplicate the configuration of an existing *framework*. This will create a snapshot of the *framework* in its

current form. It is merely a copy of the document type and will not *evolve* along with the base document type as the **Extend** action does.

Extend

Opens a **Document type configuration dialog box** ([on page 89](#)) that allows you to extend an existing *framework*. You can add or remove functionality starting from a base document type. All of these changes will be saved as a patch. When the base document type is modified and evolves (for example, from one application version to another) the extension will evolve along with the base document type, allowing it to use the new actions added in the base document type.

Delete

Deletes the selected *framework* (document type).

Locations Preferences

Oxygen JSON Editor allows you to change the location where *frameworks* ([on page 639](#)) (document types) are stored, and to specify additional *framework* directories. The **Locations** preferences page allows you to specify the main **frameworks** folder location. You can choose between the **Default** directory (`[OXYGEN_INSTALL_DIR]/frameworks`) or a **Custom** specified directory. You can also change the current **frameworks** folder location value using the `com.oxygenxml.editor.frameworks.url` system property set in either the *.vmoptions* configuration files ([on page 211](#)) or in the *startup scripts* ([on page 214](#)).

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

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

- **Internal preferences** - The document type configuration is stored in the application **Internal preferences** ([on page 90](#)).
- **Additional framework directories** - The document type configuration is loaded from one of the specified **Additional frameworks directories** list.
- **Add-ons** - An *add-on* ([on page 640](#)) can contribute a *framework*. You can manage the add-ons locations in the *Add-ons preferences page* ([on page 86](#)).
- **The frameworks folder** - The main folder containing *framework* configurations.

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

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

Document Type Configuration Dialog Box

The **Document Type Configuration** dialog box allows you to create or edit a *framework* (on page 639) (document type). It is displayed when you use the **New**, **Edit**, **Duplicate**, or **Extend** buttons in the **Document Type Association** preferences page (on page 87) (open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Document Type Association**).

Figure 5. Document Type Configuration Dialog Box

Document type

Name: DITA - Extension Priority: Normal

Description: Darwin Information Typing Architecture

Storage: ☒ Internal ☐ External

Initial edit mode: Author

[Read more about Document Type sharing and how the storage mode affects it](#)

Association rules Schema Classpath Author Templates Catalogs Transformation Validation Extensions

Namespace	Root local name	File name	Public ID	Attribute	Java class
*	*	*	*	*	ro.sync.ecss.e...
*	concept	*	*	*	
*	task	*	*	*	
*	topic	*	*	*	

OK Cancel

The configuration dialog box includes the following fields and sections:

Name

The name of the *framework*. This will be displayed as its name in the **Document Type** column in the **Document Type Association** preferences page (on page 87).

Priority

Depending on the priority level, Oxygen JSON Editor establishes the order that the existing *frameworks* are evaluated to determine the type of a document you are opening. It can be one of the following: Lowest, Low, Normal, High, or Highest. You can set a higher priority for *frameworks* you want to be evaluated first.



Note:

The built-in document types are set to Low priority by default. *Frameworks* that have the same priority are sorted alphabetically.

Description

The document type description displayed as a tooltip in the **Document Type Association** preferences page (on page 87).

Storage

The location where the framework is saved. If you select the **External** storage option, the framework is saved in a specified file with a mandatory extension (located in a subdirectory of your current framework directory. If you select the **Internal** storage option, the framework configuration data is saved in the Oxygen JSON Editor internal options file (if [Global Options \(on page 188\)](#) is selected) or in the current Oxygen JSON Editor project *xpr* file (if [Project Options \(on page 188\)](#) is selected).

Initial edit mode

Sets the default edit mode when you open a document for the first time: **Editor specific**, **Text**, **Author**, **Grid** and **Design**. If the **Editor specific** option is selected, the initial editing mode is determined based upon the document type. You can find the mapping between editors and edit modes in the [Edit modes preferences page. \(on page 117\)](#) You can impose an initial mode for opening files that match the association rules of the document type. For example, if the files are usually edited in the **Author** mode, you can set it in the **Initial edit mode** combo box.



Note:

You can also customize the initial mode for a document type in the **Edit modes** preferences page. [Open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Edit modes**.

Configuration Tabs

The bottom section of the dialog box includes various tabs where you can configure numerous options for the *framework*.

Association Rules Tab

To open the **Association Rules** tab of the **Document type** configuration dialog box, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#), go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button [\(on page 87\)](#), 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.

By combining multiple association rules you can instruct Oxygen JSON Editor to identify the type of a document. Oxygen JSON Editor identifies the type of a document when the document matches at least one of the *association rules*. This tab gives you access to a **Document type rule** dialog box that you can use to create *association rules* that activate on any document matching all the criteria defined in the dialog box.

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

Figure 6. Document Type Rule Dialog Box

The **Document type rule** dialog box includes the following fields and options:

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.

Attribute Local name

Specifies the local name of the attributes for the root element (* (*any*) by default).

Attribute Namespace

Specifies the namespace of the attributes for the root element (* (*any*) by default).

Attribute Value

Specifies the value of the attributes for the root element (* (*any*) by default).

Java class

Presents the name of the Java class that is used to determine if a document matches the rule. This Java class should implement the `ro.sync.ecss.extensions.api.DocumentTypeCustomRuleMatcher` interface.

**Tip:**

You can use wildcards (? and *) or [editor variables \(on page 197\)](#) in the **Document Type Rule** dialog box, and you can enter multiple values by separating them with a comma.

Schema Tab

In the **Schema** tab, you can specify a default schema for Oxygen JSON Editor to use if a 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 \(Options > Preferences\) \(on page 74\)](#), go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button [\(on page 87\)](#), and click on the **Schema** tab.

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

Schema type

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

Schema URI

You can specify the URI of the schema file. You can specify the path by using the text field, its history drop-down, the  **Insert Editor Variables** [\(on page 197\)](#) button, or the browsing actions 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 \(on page 203\)](#) 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* (on page 639) libraries that hold implementations for API extensions, implementations for custom **Author** mode operations, various resources (such as stylesheets), and *framework* (on page 639) translation files. Oxygen JSON Editor loads the resources looking in the folders in the order they appear in the list (from top to bottom).

To open the **Classpath** tab of the **Document type** configuration dialog box, open the **Preferences** dialog box (**Options > Preferences**) (on page 74), go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button (on page 87), and click on the **Classpath** tab.

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

New

Opens a dialog box that allows you to add a resource to the table in the **Classpath** tab. You can specify the path by using the text field, its history drop-down, the  **Insert Editor Variables** (on page 197) button, or the browsing actions in the  **Browse** drop-down list.



Tip:

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

Edit

Opens a dialog box that allows you to edit a resource in the **Classpath** tab. You can specify the path by using the text field, its history drop-down, the  **Insert Editor Variables** (on page 197) button, or the browsing actions 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.

Related information

[Extensions Tab \(on page 113\)](#)

[Author Tab \(on page 94\)](#)

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* (on page 639)-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**) (on page 74), go to **Document Type Association**, use the **New, Edit, Duplicate, or Extend** button (on page 87), 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 JSON Editor uses to render a document in the **Author** mode. In this subtab, you can set *main* and *alternate* CSS files. When you are editing a document in the **Author** mode, you can switch between these CSS files from the **Styles** drop-down menu on the **Author Styles** toolbar.

To open the **CSS** subtab, open the **Preferences** dialog box (**Options > Preferences**) (on page 74), go to **Document Type Association**, use the **New, Edit, Duplicate, or Extend** button (on page 87), click on the **Author** tab, and then the **CSS** subtab.

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

New

Opens a dialog box that allows you to add a CSS file. You can specify the path by using the text field, its history drop-down, the  **Insert Editor Variables** (on page 197) button, or the browsing actions in the  **Browse** drop-down list.

Edit

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

Delete

Deletes the currently selected CSS file.

Move Up

Moves the selected CSS file up in the list.

Move Down

Moves the selected CSS file down in the list.

Enable multiple selection of alternate CSSs

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

If there are CSSs specified in the document then

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

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

Actions Subtab

The **Actions** subtab of the **Document Type Configuration** dialog box contains a sortable table with all the **Author** mode actions that are configured for the specific *framework (on page 639)*. 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) (on page 74)*, go to **Document Type Association**, select your framework, use the **Duplicate** or **Extend** button to create an extension of the framework (or the **Edit** button for an already extended framework), click on the **Author** tab, and then the **Actions** subtab.

The following features are available in this subtab:

Export existing actions ()

It is possible to export existing actions to use them in other frameworks. Each exported action is extracted from the framework configuration file and exported as an individual XML file.

To export actions, the **Storage** option *(on page 90)* in the top part of the **Document Type Configuration** dialog box must be set to **External** and the external location must be a subdirectory of your current framework directory.

The  **Export** action is found by right-clicking an action or a selection of multiple actions (the  **Export** button is also located below the table of actions). If you choose to export a single action, a resulting dialog box will allow you to select the destination path for the new XML file that contains the configuration details of the action. If you export multiple actions, they will automatically be saved as individual XML files inside a newly created folder (it will have **_externalAuthorActions** at the end of the folder name) inside your current framework directory.

Result: Exported actions will display the  icon in the first column in the table.



Important:

The newly created files for the exported actions will not appear on disk until you click **OK** several times to confirm your changes and exit the **Preferences** dialog box.

**Tip:**

If you want to create a new XML file for an action, there is a document template called **Author Actions** in the to help you get started.

**Note:**

You can add or edit the action files outside of Oxygen JSON Editor, but you will need to restart the application each time to reload the changes.

Open in editor

For exported actions, there is a  **Open in editor** action in the contextual menu that will open the file for that action in the main editor.

Create a new action

Use the  **New** button (located underneath the table of actions) to open the **Action dialog box** [\(on page 96\)](#) where you can configure a new action.

Duplicate an existing action

Use the  **Duplicate** action (found in the contextual menu and underneath the table of actions) to duplicate the selected action.

Edit an existing action

Use the  **Edit** button (found in the contextual menu and underneath the table of actions) to open the **Action dialog box** [\(on page 96\)](#) where you can edit the selected action.

Delete an existing action

Use the  **Delete** button (found in the contextual menu and underneath the table of actions) to delete the selected action.

Author Action Dialog Box

To edit an existing document type action or create a new one, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#), go to **Document Type Association**, use the **New, Edit, Duplicate, or Extend** button [\(on page 87\)](#), click on the **Author** tab, and then the **Actions** subtab. At the bottom of this subtab, click  **New** to create a new action, or  **Edit** to modify an existing one.

Figure 7. Action Dialog Box

Action

ID: Description:

Name:

Menu access key:

[How to translate frameworks](#)

Large icon (24x24):

Small icon (16x16):

Shortcut key:

☒ Enable platform-independent shortcut keys

Operations

1

Activation XPath:

This XPath expression applies only to elements as: [More details](#)

Operation:

Arguments:

Name	Description	Type	Value
element	The element to surround with.	Fragment	
schemaAware	This argument applies only on the sun	ConstantList	true

Operation priority:

☐ Evaluate activation XPath expressions even in read-only contexts

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 ([on page 204](#)) to allow for multiple translations of the name.

Menu access key

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

Description

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

**Tip:**

You can use the `${i18n('key')}` editor variable (on page 204) to allow for multiple translations of the description.

How to translate frameworks link

Use this link to see more information about .

Large icon

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

**Tip:**

A good practice is to store the image files inside the *framework* directory and use the `${frameworks}` editor variable (on page 203) to make the image relative to the *framework* location. If the images are bundled in a *jar* archive (for instance, along with some Java operations implementation), it is convenient to reference the images by their relative path location in the *class-path*.

Small icon

Allows you to select an image for the icon that Oxygen JSON Editor 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 selected, the shortcut that you specify in this field is platform-independent and the following modifiers are used:

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

Operations section

In this section of the **Action** dialog box, you configure the functionality of the action that you are editing. An action has one or more operation modes. The evaluation of an XPath expression activates an operation mode. The first selected operation mode is activated when you trigger the action. The scope of the XPath expression must consist only of element nodes and attribute nodes of the edited document. Otherwise, the XPath expression does not return a match

and does not fire the action. For more details see: [Controlling Which Author Operations Gets Executed Through XPath Expressions \(on page 99\)](#).

The following options are available in this section:

Activation XPath

An XPath 2.0 expression that applies to elements and attributes. For more details see: [Controlling Which Author Operations Gets Executed Through XPath Expressions \(on page 99\)](#).

Operation

Specifies the invoked operation that can be a or a .

Arguments

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

Operation priority

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

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

Evaluate activation XPath expressions even in read-only contexts

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

Controlling Which Author Operations Gets Executed Through XPath Expressions

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

For each operation mode of an action, the application will check if the XPath expression is fulfilled (when it returns a non-empty node set or a **true** result). Only the first operation whose XPath operation is fulfilled will be executed.

The following special XPath extension functions are provided:

- [oxy:allows-child-element\(\) \(on page 100\)](#) - Use this function to check whether or not an element is valid child element in the current context, according to the associated schema.
- [oxy:allows-global-element\(\) \(on page 102\)](#) - Use this function to check whether or not an element is a valid global element for the current [framework \(on page 639\)](#), according to the associated schema.

- [oxy:current-selected-element\(\)](#) (on page 103) - Use this function to get the currently selected element.
- [oxy:selected-elements\(\)](#) (on page 103) - Use this function to get the selected elements.
- [oxy:is-required-element\(\)](#) (on page 104) - Use this function to check if the element returned by the given XPath expression is required (based on the rules declared in the schema).
- [oxy:platform\(\)](#) (on page 104) - Use this function to get the current platform in cases where you want to enable or disable an action depending on the platform. Possible values include: *eclipse*, *standalone* and *webapp*.

oxy:allows-child-element() Function

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

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

The following parameters are supported:

childName

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

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

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

The above example verifies if the `<para>` element (of the default namespace) is allowed in the current context.

- The child element with the local name specified by any namespace.

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

The above example verifies if the `<para>` element (of any namespace) is allowed in the current context.

- A prefix-qualified name of an element.

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

The prefix is resolved in the context of the element where the cursor is located. The function matches on the element with the `para` local name from the previously 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 to it.

`contains`

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

oxy:allows-global-element() Function

The **oxy:allows-global-element()** function allows you to check whether or not an element that matches the arguments of the function is valid for the current *framework* (on page 639), according to the associated schema. It has the following signature:

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

The following parameters are supported:

elementName

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

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

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

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

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

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

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

- A prefix-qualified name of an element.

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

The prefix is resolved in the context of the *framework*. The function matches on the element with the `para` local name from the previously resolved namespace. If the prefix is not resolved to a namespace, the function returns a value of `false`.

- A specified namespace-URI-qualified name of an element.

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

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

- Any element.

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

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

attributeName

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

- The attribute with the specified name from no namespace.

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

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

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

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

- A qualified name of an attribute.

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

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

defaultAttributeValue

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

contains

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

oxy:current-selected-element() Function

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

Example: oxy:current-selected-element Function

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

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

oxy:selected-elements() Function

This function returns the selected elements from **Author** mode.

Example: oxy:selected-elements Function

```
oxy:selected-elements()[self::para][@audience="novice"]
```

This example would activate an action when at least one of the selected elements is a `<para>` element with the `@novice` attribute defined.

oxy:is-required-element() Function

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

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

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

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

oxy:is-editable-element() Function

This function checks if the element returned by the given XPath expression is editable (content can be inserted in it), meaning both that the entire XML file is editable and that the current context where the element is placed is editable. For example, if the element is inside an *xi:included* section, it is not editable.

It only has one argument, an XPath expression, and the XPath expression must be written in such a way that it returns a single element.

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

```
oxy:is-editable-element(ancestor-or-self::table)
```

This example would return *true* if the cursor is placed inside a table and it is editable or *false* if it is not editable.

oxy:platform() Function

This function returns the current platform. You can use this if you want to enable or disable an action depending on the platform. The possible values are: **standalone**, **eclipse**, or **webapp**.

Example: *oxy:platform* Function

```
oxy:platform()="standalone"
```

This example would keep the action activated for the *standalone* distribution of Oxygen JSON Editor, but disable it for the *Eclipse* and *Web Author* distributions.

Menu Subtab

In the **Menu** subtab, you can configure which actions will appear in the *framework* (on page 639)-specific menu. The subtab is divided into two sections: **Available actions** and **Current actions**.

To open the **Menu** subtab, open the **Preferences** dialog box (**Options > Preferences**) (on page 74), go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button (on page 87), 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 JSON Editor menu. To add an action in this section as a sibling of the currently selected action, use the

 **Add as sibling** button. To add an image in this section as a child of the currently selected action, use the  **Add as child** button.

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



Edit

Edits an item.



Remove

Removes an item.



Move Up

Moves an item up.



Move Down

Moves an item down.

Contextual Menu Subtab

In the **Contextual menu** subtab you configure what *framework* (on page 639)-specific action the *Content Completion Assistant* (on page 638) proposes. The subtab is divided into two sections: **Available actions** and **Current actions**.

To open the **Contextual Menu** subtab, open the **Preferences** dialog box (**Options > Preferences**) (on page 74), go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button (on page 87), click on the **Author** tab, and then the **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

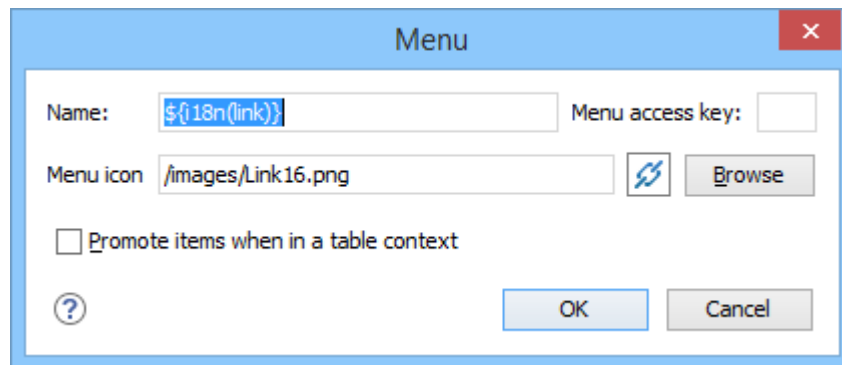
Adds the selected action or submenu from the **Available actions** section to the **Current actions** section as a child of the selected action.



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 8. Menu Action Configuration Dialog Box



The following options are available in this dialog box:

Name

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



Tip:

You can use the `${i18n('key')}` editor variable ([on page 204](#)) to allow for multiple translations of the name.

Menu access key

In Windows, you can access menus by holding down **Alt** and pressing the keyboard key that corresponds to the *letter* that is underlined in the name of the menu.

Then, while still holding down **Alt**, you can select submenus and menu action the same way by pressing subsequent corresponding keys. You can use this option to specify the *letter* in the name of the action that can be used to access the action.

Menu icon

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

Promote items when in a table context

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

✕ Remove

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

 Move Up

Moves the selected item up in the list.

 Move Down

Moves the selected item down in the list.

Toolbar Subtab

In the **Toolbar** subtab you configure what *framework* (on page 639)-specific action the Oxygen JSON Editor toolbar holds. The subtab is divided into two sections: **Available actions** and **Current actions**.

To open the **Toolbar** subtab, open the **Preferences** dialog box (**Options > Preferences**) (on page 74), go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button (on page 87), 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 JSON Editor toolbar when you work with a document belonging to the edited *framework*. To add an action in this section as a sibling of the currently selected action, use the  **Add as sibling** button. To add an action in this section as a child of the currently selected action, use the  **Add as child** button.

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

 Edit

Edits an item.

 Remove

Removes an item.

 Move Up

Moves an item up.

 Move Down

Moves an item down.

Content Completion Subtab

In the **Content Completion** subtab you configure what *framework* (on page 639)-specific the *Content Completion Assistant* (on page 638) proposes. The subtab is divided into two sections: **Available actions** and **Current actions**.

To open the **Content Completion** subtab, open the **Preferences** dialog box (**Options > Preferences**) (on page 74), go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button (on page 87), click on the **Author** tab, and then the **Content Completion** subtab.

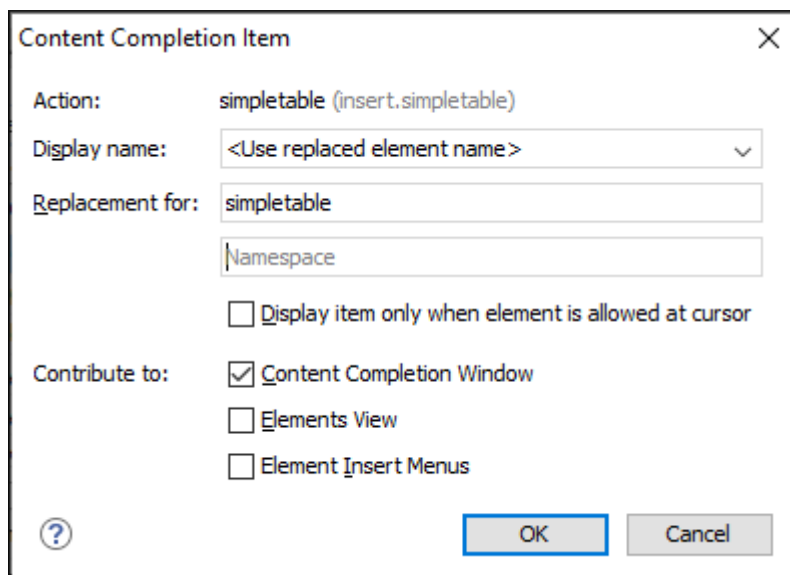
Available and Current Actions

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 that belongs to the edited *framework*.

To add the selected available action as a sibling of the currently selected action in the **Current actions** section, use the  **Add as sibling** button. To add it as a child of the currently selected action, use the  **Add as child** button. To edit an existing action, select it and use the  **Edit** button. To remove an existing action, use the  **Remove** button. You can also move items up and down the list using the  **Move Up** or  **Move Down** buttons.

Adding an action (or editing an existing one) opens the **Content Completion Item** dialog box.

Figure 9. Content Completion Item Dialog Box



The dialog box titled "Content Completion Item" contains the following fields and controls:

- Action:** A text field containing "simpletable (insert.simpletable)".
- Display name:** A drop-down menu with the selected option "<Use replaced element name>".
- Replacement for:** A text field containing "simpletable".
- Namespace:** A text field containing "Namespace".
- Display item only when element is allowed at cursor:** An unchecked checkbox.
- Contribute to:** A group of three checkboxes:
 - ☒ Content Completion Window
 - ☐ Elements View
 - ☐ Element Insert Menus
- Buttons:** A question mark icon, an "OK" button, and a "Cancel" button.

Use this dialog box to configure the action:

Action

Displays the name of the selected action.

Display name

You can use the drop-down menu to choose between displaying the action name or the replaced element name, or you can enter another name to be displayed.

Replacement for

Use this section to replace an existing element with the configured action:

- **Element name** and **Namespace** - The name (and namespace, if needed) of the replaced element. The original element no longer needs to be excluded using the [Filter - Remove content completion items table \(on page 109\)](#).
- **Display item only when element is allowed at cursor** - The configured action is contributed in the UI components selected in the **Contribute to** section only if the associated schema allows the original element at the current location in the document. This is equivalent to defining activation XPaths in the [Author Action dialog box \(on page 96\)](#) using the `oxy:allows-child-element()` XPath extension function. Activation XPaths for the action are still checked when the action is invoked.

Contribute to

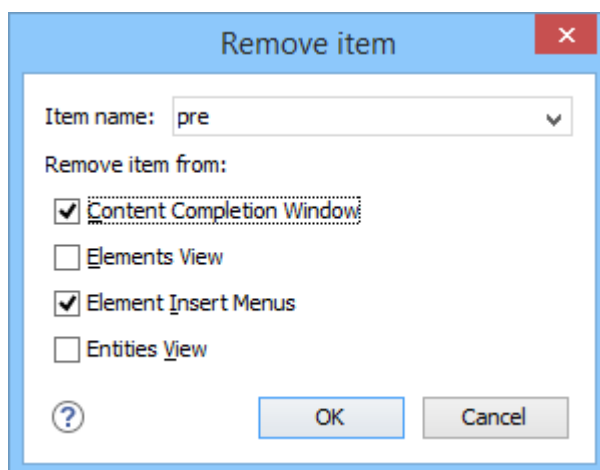
Use this section to specify where to display the configured item in the interface:

- **Content Completion Window** - The configured item will appear in the [Content Completion Assistant \(on page 638\)](#).
- **Elements View** - The configured item will appear in the **Elements** view.
- **Element Insert Menus** - The configured item will 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).

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 10. 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> [elementName]** - Filters split entries for elements that have the form **Split elementName** or **New elementName**.
- **<SPLIT>** - Filters split entries for all elements.
- **<ENTER>** - Filters **Insert New Line** entries that appear in elements where whitespace is significant.

The filter list can be used to remove only content completion items contributed by the schema associated to the document and it does not remove actions added to the content completion list via the framework configuration. The element name specified in the filter is not namespace aware, it matches the name of the element defined in the associated schema exactly as it would appear rendered in the content completion window.



Note:

If you try to insert an element in an invalid position (for example, using the content completion assistant), the editor will attempt to make the insertion valid. This may mean finding an alternate position for the insertion or splitting the element at the current position. If a **<SPLIT>** entry is added in the filter list for an element, the editor will never split that element.

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* (on page 638).
- **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).

Templates Tab

The **Templates** tab specifies a list of directories where new document templates are located for this particular framework. These directories, along with the document templates that are saved inside them, will appear in the **New Document** wizard (on page 225) inside the **Framework templates** category according to your framework and the directory path you specify in this tab.

To open the **Templates** tab of the **Document type** configuration dialog box, open the **Preferences** dialog box (**Options > Preferences**) (on page 74), go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button (on page 87), 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 a directory that contains document templates for this framework. You can specify the path by using the text field, its history drop-down, the  **Insert Editor Variables** (on page 197) button, or the browsing actions 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 a directory that contains document templates for this framework. You can specify the path by using the text field, its history drop-down, the  **Insert Editor Variables** (on page 197) button, or the browsing actions 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 directory from the list.

Move Up

Moves the selected template directory up one spot in the list.

Move Down

Moves the selected template directory down one spot in the list.

Related information

[Creating New Document Templates \(on page 229\)](#)

[Customizing Document Templates \(on page 230\)](#)

[Sharing Custom Document Templates \(on page 234\)](#)

Catalogs Tab

The **Catalogs** tab specifies a list of *XML Catalogs*, specifically for the edited *framework* (on page 639), that are added to list of catalogs that Oxygen JSON Editor uses to resolve resources.

To open the **Catalogs** tab of the **Document type** configuration dialog box, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#), go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button [\(on page 87\)](#), and click on the **Catalogs** tab.

You can perform the following actions:

Add

Opens a dialog box that allows you to add a catalog to the list. You can specify the path by using the text field, its history drop-down, the  **Insert Editor Variables** [\(on page 197\)](#) button, or the browsing actions 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.

Validation Tab

In the **Validation** tab, you can configure the validation scenarios associated with the particular *framework* [\(on page 639\)](#) 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 *main file* is associated with the current file, the validation scenarios defined in the *main file*, along with any Schematron schema defined in the default scenarios for that particular *framework*, are used for the validation. These take precedence over other types of validation units defined in the default scenarios for the particular *framework*.

To open the **Validation** tab of the **Document type** configuration dialog box, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#), go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button [\(on page 87\)](#), and click on the **Validation** tab.

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

Default checkbox

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

New

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

Duplicate

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

Edit

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

Delete

Deletes the currently selected validation scenario.

Import scenarios

Imports validation scenarios.

Export selected scenarios

Export validation scenarios.

Move Up

Moves the selected scenario up one spot in the list.

Move Down

Moves the selected scenario down one spot in the list.

Extensions Tab

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

To open the **Extensions** tab of the **Document type** configuration dialog box, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#), go to **Document Type Association**, use the **New, Edit, Duplicate, or Extend** button [\(on page 87\)](#), and click on the **Extensions** tab.

Libraries containing the implementations must be present in the [classpath \(on page 93\)](#) of your document type. The Javadoc available at <https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/> contains details about how each API implementation functions.

Document Templates Preferences

Oxygen JSON Editor provides a variety of built-in document templates that make it easier to create new documents in various formats. The list of available templates is presented in the **New Document** wizard (on page 225) when you create a new document ( **New** toolbar button or **File > New**).

You can also [create your own templates](#) (on page 229) and share them with others. You can store your custom document templates in the existing **templates** folder in the Oxygen JSON Editor installation directory or store them in a custom directory. If you store them in a custom directory, you need to use this **Document Templates** preferences page to add that directory to the list of template directories that Oxygen JSON Editor makes available in the **New Document** wizard.

To add a template directory, follow these steps:

1. open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Document Templates**.
2. Use the **New** button to select a location of the new document template folder.
3. You can also use the **Edit** or **Delete** buttons to manage folders in the list, and you can alter the order that Oxygen JSON Editor looks in these directories by using the **Up** and **Down** buttons.

Result: This will add the folder to the list in this preferences page and it will now appear in the **New Document** wizard (on page 225) in a category based upon the folder path you specified.

Encoding Preferences

Oxygen JSON Editor lets you configure how character encodings are recognized when opening files and which encodings are used when saving files. To configure encoding options, [open the Preferences dialog box](#) (**Options > Preferences**) (on page 74) 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 JSON Editor tries to collect the character encoding from the document. If no such encoding is found, then *UTF-8* is used.
- For JavaScript, JSON, SQL, XQuery, and RNC, the *UTF-8* encoding is used.

UTF-8 BOM handling

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

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



Note:

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

Encoding errors handling

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

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



Attention:

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

Encode non-ASCII characters in URL paths

If selected (default), Oxygen JSON Editor will escape non-ASCII characters (encode them with their hexadecimal equivalent) within URL paths. If you are using a non-Latin alphabet (such as Arab, Japanese, Chinese), it may be beneficial to deselect this option so that non-ASCII characters in URL paths will not be escaped and will remain more readable.

Editor Preferences

Oxygen JSON Editor offers the possibility to configure the appearance of various components and features of the main editor. To access these options, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) 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 (on page 638)*.

Completion proposal foreground

Allows you to set the color of the text in the *Content Completion Assistant (on page 638)*.

Documentation window background

Allows you to set the background color of the documentation of elements suggested by the *Content Completion Assistant (on page 638)*.

Documentation window foreground

Allows you to set the color of the text for the documentation of elements suggested by the *Content Completion Assistant (on page 638)*.

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 JSON Editor displays.

Show TAB/NBSP/EOL/EOF marks

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

Show SPACE marks

Makes the space character visible in the editor.

Can edit read-only files

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

Undo history size

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

Enable mouse-wheel zooming

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

Edit Modes Preferences

Oxygen JSON Editor lets you configure which edit mode a file is opened in the first time it is opened. This setting only applies the first time a file is opened. The current editing mode of each file is saved when the file is closed and restored the next time it is opened. To configure the options for editing modes, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Edit Modes**.

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

If selected, the initial edit mode setting set in the [Document Type configuration dialog box \(on page 89\)](#) overrides the general edit mode setting from the table below.

Select the initial edit mode (page) for each editor

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

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

Text Preferences

Oxygen JSON Editor allows you to configure how the **Text** mode editor appears. To configure these options, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) 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 \(on page 579\)](#), [Compare Files \(on page 313\)](#), [Compare Directories \(on page 332\)](#)).

Editor cursor color

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

Highlight current line

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

Show line numbers

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

Show print margin

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

Print margin column

Allows you to specify a limit for the print width, measured in the number of characters.

Line wrap

If selected, long lines are automatically wrapped in edited documents. The line wrap does not alter the document content since the application does not use *new-line* characters to break long lines.

Cut / Copy whole line when nothing is selected

If selected, **Cut** and **Copy** actions operate on the entire current line when nothing is selected in the editor.

Enable folding

If selected (default value), the vertical stripe that holds the folding markers is displayed in **Text** mode.

XML section**Highlight matching tag**

If selected, when you place the cursor on a start or end tag, Oxygen JSON Editor highlights the corresponding member of the pair. You can also customize the highlight color.

Lock the XML tags

If selected, XML are locked and cannot be edited in **Text** mode.

JSON section**Show property name after ending bracket**

If enabled (default), property names are displayed after the ending bracket when editing JSON documents in **Text** mode.

YAML section

Show SPACE marks for YAML only

If enabled (default), space characters are visible in the YAML editor.

Grid Preferences

To configure the **Grid** mode options, [open the Preferences dialog box \(Options > Preferences\)](#) ([on page 74](#)) and go to **Editor > Edit modes > Grid**.

The following options are available:

Compact representation

If selected, the *compact representation* of the grid is used: a child element is displayed beside the parent element. In the *non-compact representation*, a child element is nested below the parent.

Format and indent when passing from grid to text or on save

If selected, the content of the document is formatted and indented each time you switch from the **Grid** view to the **Text** view.

Default column width (characters)

Sets the default width (in characters) of a table column of the grid. A column may contain the following:

- Element names
- Element text content
- Attribute names
- Attribute values

If the total width of the grid structure is too large you can resize any column by dragging the column margins with the mouse pointer, but the change is not persistent. To make it persistent, set the new column width with this option.

Active cell color

Allows you to set the background color for the [active cell \(on page 637\)](#) of the grid. The keyboard input always goes to the *active cell* and the selection always contains it.

Selection color

Allows you to set the background color for the selected cells of the grid, except the [active cell \(on page 637\)](#).

Border color

Allows you to set the color used for the lines that separate the grid cells.

Background color

Allows you to set the background color of grid cells that are not selected.

Foreground color

Allows you to set the text color of the information displayed in the grid cells.

Row header colors

Background color

Allows you to set the background color of row headers that are not selected.

Active cell color

Allows you to set the background color of the row header cell that is currently active.

Selection color

Allows you to set the background color of the header cells corresponding to the currently selected rows.

Column header colors

The column headers are painted with two color gradients, one for the upper 1/3 part of the header and the other for the lower 2/3 part. The start and end colors of the first gradient are set with the first two color buttons. The start and end colors of the second gradient are set with the last two color buttons.

Background color

Allows you to set the background color of column headers that are not selected.

Active cell color

Allows you to set the background color of the column header cell that is currently active.

Selection color

Allows you to set the background color of the header cells corresponding to the currently selected columns.

Author Preferences

Oxygen JSON Editor provides an **Author** editing mode that provides a configurable graphical interface for editing documents. To configure the options for the **Author** mode, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Edit modes > Author**.

The following options are available:

Author default background color

Sets the default background color of the **Author** editing mode. The `background-color` property set in the CSS file associated with the currently edited document overwrites this option.

Author default foreground color

Sets the default foreground color of the **Author** editing mode. The `color` property set in the CSS file associated with the currently edited document overwrites this option.

Show XML comments

When this option is selected, XML comments are displayed in **Author** mode. Otherwise, they are hidden.

Show placeholders for empty elements

When this option is selected, placeholders are displayed for elements with no content to make them clearly visible. The placeholder is rendered as a light gray box and displays the element name.

Show processing instructions

When this option is selected, XML processing instructions are displayed in **Author** mode. Otherwise, they are hidden.

Show Author layout messages

When this option is selected, all errors reported while rendering the document in **Author** mode are presented in the **Results** view at the bottom of the editor.

Show doctype

When this option is selected, the **doctype** declaration is displayed in **Author** mode. Otherwise, it is hidden.

Show block range

When this option is selected, a *block range indicator* is displayed in a stripe located in the left side of the editor. It is displayed as a heavy line that spans from the first line to the last line of the block.

Fast text layout

In certain cases, the widths computed in the **Author** visual editing mode for lines of text may be larger than expected, leading to an incorrect visual layout. Deactivating this option will improve the computation quality for character widths in the visual editing mode, but it may hinder overall performance for very large documents.

**Tip:**

For macOS users, some specific examples of this type of situation can be found here: [Text Rendering Issues on macOS \(on page 636\)](#). Because of such problems, when an installation kit with Java 9 or newer is used on macOS, the checkbox is not selected by default.

Show floating contextual toolbar

When this option is selected (default), the *floating contextual toolbar* is displayed in the **Author** mode in certain situations. When not selected, the *floating contextual toolbar* is never displayed.

Images Section

The following options regarding images in **Author** mode are available in this section:

Auto-scale images wider than (pixels)

Sets the maximum width that an image will be displayed. Wider images will be scaled to fit.

Show very large images

When this option is selected, images larger than 6 megapixels are displayed in **Author** mode. Otherwise, they are not displayed.

**Important:**

If you select this option and your document contains many such images, Oxygen JSON Editor may consume all available memory, throwing an **OutOfMemory error**. To resolve this, increase the [available memory limit \(on page 211\)](#) and restart the application.

Tags Section

In this section, you can configure the following options regarding tags that are displayed in **Author** mode:

Tags display mode

Sets the default display mode for element tags presented in **Author** mode. You can choose between the following:

**Full Tags with Attributes**

Displays full tag names with attributes for both *block* [\(on page 637\)](#) and *inline elements* [\(on page 639\)](#). Oxygen JSON Editor

**Full Tags**

Displays full tag names without attributes for both *block elements* and *inline elements*.

**Block Tags**

Displays full tag names for *block elements* and simple tags without names for *inline elements*.

**Block Tags without Element Names**

Displays tags for *block elements* but without element names for a more compact version of **Block Tags** mode. You can still see the element names by hovering over the tags.

**Inline Tags**

Displays full tag names for *inline elements*, while *block elements* are not displayed.

**Partial Tags**

Displays simple tags without names for *inline elements*, while *block elements* are not displayed.

No Tags

No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Sort attributes alphabetically for "Full Tags with Attributes"

When selected, if you choose  **Full Tags with Attributes** for the **Tags Display Mode**, the attributes will be displayed in alphabetical order. Otherwise, they are displayed in the order that they appear in the XML source code.

Tags background color

Sets the **Author** mode tags background color.

Tags foreground color

Sets the **Author** mode tags foreground color.

Tags font

Allows you to change the font used to display tags text in the **Author** visual editing mode. The *default* font is computed based on the setting of the **Author default font** option in the **Fonts** preferences page (*on page 83*).

Compact tag layout

If this option is not selected, the **Author** mode displays the tags in a more decompressed layout, where block tags are displayed on separate lines.

References Section

Display referenced content (external entities, XInclude, DITA conref, etc.)

When selected, the references (such as external entities, XInclude, DITA conrefs) also display the content of the resources they reference. When the option is not selected, the referenced resources are not automatically loaded and displayed, but the referenced content can be expanded on demand by using the small expansion button located next to each element that contains references. If you toggle this option while editing, you need to reload the file for the modification to take effect.

Allow referenced content to be edited

When selected, for a specific XML vocabulary that supports this feature, the content referenced from other files and presented in the **Author** visual editing mode can be edited in-place and saved. For now, if the feature is enabled and you use the  **Open Map in Editor with Resolved Topics** toolbar action in the **DITA Maps Manager** view, the referenced content in the opened document becomes editable in-place. Saving the document will save all other modified topics.

Local files only

When selected (default), the **Allow referenced content to be edited (Experimental)** option only works for local files. For files located in remote locations such as a CMS, additional steps might be necessary to save all modified content because this feature might not function properly with remote resources.

For advanced Author configuration see the Document Type Association settings

Click this link to open the [Document Type Association preferences page \(on page 87\)](#).

Cursor Navigation Preferences

Oxygen JSON Editor allows you to configure the appearance and behavior of the cursor in the **Author** mode editor. To set cursor navigation preferences, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Author > Cursor Navigation**.

The following options are available:

Highlight elements near cursor

When this option is selected, the element that contains the cursor is highlighted. If the cursor is between two elements, both of them are highlighted. You can use the color picker to choose the color of the highlight.

Show cursor position tooltip

Oxygen JSON Editor uses tooltips in **Author** mode to indicate the position of the cursor in the element structure of the underlying document. Depending on context, the tooltips may show the current element name or the names of the elements before and after the current cursor position.

Show location tooltip on mouse move

When this option is selected, Oxygen JSON Editor displays tooltips when you are editing the document in certain tags display modes (**Inline Tags**, **Partial Tags**, **No Tags**) or when the mouse pointer is moved between [block elements \(on page 637\)](#).

Quick up/down navigation

This option is deselected by default and this means that when you navigate using the up and down arrow keys in **Author** mode, the cursor is placed within each of the underlying XML elements between two blocks of text (the cursor changes to a horizontal line when it is between blocks of text). This allows you to easily insert elements and manage the structure of your XML content. However, if this option is selected, the cursor ignores the XML structure and jumps from one line of text to another, similar to how the cursor behaves in a word processor.

Quick navigation in tables

This option is selected by default and this means that when navigating between table cells with the arrow keys, the cursor jumps from one cell to another. If this option is not selected, the cursor navigates between XML nodes when navigating between table cells with the arrow keys.

Avoid positioning the cursor between blocks after a deletion

If selected (default), the cursor will not stay between block element sentinels after a deletion is performed.

Arrow keys move the cursor in the writing direction

This setting determines how the left and right arrow keys behave in **Author** mode for bidirectional (BIDI) text. When this option is selected (default value), the right arrow key advances the cursor in the reading direction and the left arrow moves it in the opposite direction. When this option is not selected, pressing the right arrow will simply move the cursor to the right (and the left arrow moves it to the left), regardless of the text direction.

Schema-Aware Preferences

Oxygen JSON Editor can use the schema of your XML language to improve the way the **Author** mode editor handles your content. To configure the **Schema-Aware** options, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Edit modes > Author > Schema-Aware**.

The following options are available:

Schema-aware normalization, format, and indent

When you open or save a document in **Author** mode, white space is normalized using the `display` property of the current CSS stylesheet and the values of the settings for **Preserve space elements**, **Default space elements**, and **Mixed content elements**. When this option is selected, the schema will also be used to normalize white space, based on the content model (*element-only*, *simple-content*, or *mixed*). Note that the schema information takes precedence.

Indent blocks-only content

To avoid accidentally introducing inappropriate white space around *inline elements* (on page 639), Oxygen JSON Editor does not normally apply indenting to the source of an element with mixed content. If this option is selected, Oxygen JSON Editor will apply indenting to the source of mixed content elements that only contain *block elements* (on page 637).

Schema-Aware Editing

The options in this section determine how Oxygen JSON Editor will use the schema of a document to control the behavior of the **Author** mode.

- **On** - Enables all schema-aware editing options.
- **Off** - Disables all schema-aware editing options.
- **Custom** - Allows you to select custom schema-aware editing options from the following:

Schema-Aware Actions section

Delete element tags with backspace and delete

Controls what happens when you attempt to delete an element tag. The two options are:

- **Smart delete** - If deleting the tag would make the document invalid, Oxygen JSON Editor will attempt to make the document valid by unwrapping the current element or by appending it to an adjacent element where the result would be valid. For instance, if you delete a bold tag, the content can be unwrapped and become part of the surrounding paragraph, but if you delete a list item tag, the list item content cannot become part of the list container. However, the content could be appended to a preceding list item.
- **Reject action when its result is invalid** - A deletion that would leave the document in an invalid state is rejected.

Paste and Drag and Drop

Controls the behavior for paste and drag and drop actions. Available options are:

- **Smart paste and drag and drop** - If the content inserted by a paste or drop action is not valid at the cursor position, according to the schema, Oxygen JSON Editor tries to find an appropriate insert position. The possibilities include:
 - Creating a sibling element that can accept the content (for example, if you tried to paste a paragraph into an existing paragraph).
 - Inserting the content into a parent or child element (for example, if you tried to paste a list item into an existing list item, or into the space above or below an existing list).
 - Inserting the content into an ancestor element where it would be valid.
- **Reject action when its result is invalid** - If selected, Oxygen JSON Editor will not let you paste content into a position where it would be invalid.

Typing

Controls the behavior that takes place when typing. Available options are:

- **Smart typing** - If typed characters are not allowed in the element at the cursor position, but the previous element does allow text, then a similar element will be inserted, along with your content.
- **Reject action when its result is invalid** - If selected, and the result of the typing action is invalid, the action will not be performed.

Content Completion

Controls the behavior that takes place when inserting elements using the *Content Completion Assistant* in **Author** mode. Available options are:

- **Press ENTER to show available content completion proposals** - If selected, pressing **Enter** will open the *Content Completion Assistant*. If deselected, there are three possibilities:
 - The current element will be split (if possible).
 - A new element with the same name will be inserted (if possible).
 - Otherwise, a new paragraph will be inserted.
- **Show all possible elements in the content completion list** - If selected, the content completion list will show all the elements in the schema, even those that cannot be entered validly at the current position. If you select an element that is not valid at the current position, Oxygen JSON Editor will attempt to find a valid location to insert it and may present you with several options.
- **Allow only insertion of valid elements and attributes** - If selected, you can only select elements in the content completion list that are valid (according to the schema) at the current position.
- **Allow only insertion of valid attribute values** - If selected, you cannot enter an attribute value that is not valid (according to the schema). If the attribute has a choice of values, you can select a possible value from a drop-down list in the combo box, but you cannot enter a value manually.

Warn on invalid content when performing action

A warning message will be displayed when performing an action that will result in invalid content. Available options are:

- **Delete Element Tags** - If selected, a warning message will be displayed if the Delete Element Tags action will result in an invalid document. You will be asked to confirm the deletion.
- **Join Elements** - If selected, a warning message will be displayed if the Join Elements action will result in an invalid document. You will be asked to confirm the join.

Automatically apply the best schema-aware insertion operation

If selected, Oxygen JSON Editor automatically uses what it considers to be the best insertion solution, when there is an attempt to insert content that is not valid in a specific context. If not selected, Oxygen JSON Editor will ask the user to choose from a list of proposed solutions.

Convert external content on paste

If selected, the *Smart Paste* feature is enabled when external content is pasted in **Author** mode.

Convert even when pasting inside space-preserve elements

If selected, the *Smart Paste* feature will be used even when external content is pasted inside a *space-preserve* element (such as a `<codeblock>`).

Convert pasted URLs to links

If selected, when a URL is pasted into **Author** mode, a link will be inserted (the type of link depends on the type of document). For example, in DITA documents, an `<xref>` is inserted.

AutoCorrect Preferences

Oxygen JSON Editor includes an option to automatically correct misspelled words as you type in **Author** mode. To enable and configure this [AutoCorrect feature \(on page 300\)](#), open the **Preferences** dialog box (**Options > Preferences**) ([on page 74](#)) and go to **Editor > Edit Modes > Author > AutoCorrect**.

The following options are available:

Enable AutoCorrect

When selected (default state), while editing in **Author** mode, if you type anything that is listed in the **Replace** column of the Replacements table displayed in this preferences page, Oxygen JSON Editor will automatically replace it with the value listed in the **With** column.

Use additional suggestions from the spell checker

If selected, in addition to anything listed in the Replacements table displayed in this preferences page, Oxygen JSON Editor will also use suggestions from the Spell Checker to automatically correct misspelled words. Suggestions from the Spell Checker will only be used if the misspelled word is not found in the Replacements table.



Note:

The *AutoCorrect* feature shares the same options configured in the [Language options \(on page 156\)](#) and [Ignore elements \(on page 158\)](#) sections in the **Spell Check** preferences page.

Include text-to-markup corrections based on the current document type

If selected, in addition to anything listed in the Replacements table displayed in this preferences page, the *AutoCorrect* mechanism will also include XML markup insertion rules specified in a configuration file for each document type. For example, for default DITA, DocBook, and TEI documents, entering a hyphen (-) followed by a space in an empty paragraph will automatically insert a list element with an empty list item element inside. The configuration file is located at: `[OXYGEN_INSTALL_DIR]/frameworks/[DOC_TYPE]/resources/structureAutocorrect.xml`.

Spell Check options link

Use this link to navigate to the [Spell Check Preferences page \(on page 156\)](#).

Replacements Table section

The *AutoCorrect* feature uses the Replacements table to automatically replace anything that is listed in the **Replace** column with the value listed in the **With** column for each language.

Replacements for language drop-down menu

You can specify the language for the Replacements table, and for each language, you can configure the items listed in the table. The language selected in this page is not the language that will be used by the *AutoCorrect* feature. It is simply the language for the Replacements table.

Replacements Table

You can double-click on cells in either column to edit the listed items. Use the **Add** button to insert new items and the **Remove** button to delete rows from the table.



Note:

Any changes, additions, or deletions you make to this table are saved to a path that is specified in the [AutoCorrect Dictionaries preferences page \(on page 131\)](#).

Smart quotes section

You can also choose to automatically convert double and single quotes to a quotation character of your choice by using the following options in the **Smart quotes** section:

- **Replace "Single quotes"** - Replaces single quotes with the quotation symbols you select with the **Start quote** and **End quote** buttons.
- **Replace "Double quotes"** - Replaces double quotes with the quotation symbols you select with the **Start quote** and **End quote** buttons.



Note:

These **Smart quotes** options are ignored for content inside any element listed in the [Ignore elements](#) section of the [Spell Check preferences page \(on page 158\)](#).

Global Options [\(on page 639\)](#)

If this option is selected, the options are stored on your local computer, in a folder that is not accessible to other users.

Project Options [\(on page 640\)](#)

If this option is selected, the options are stored in the project file and can be shared with other users. Selecting [Project Options \(on page 640\)](#) will only save your selections in [Enable AutoCorrect \(on page 128\)](#), [Use additional suggestions from the spell checker \(on page 128\)](#), and the options in the [Smart quotes section \(on page 129\)](#). Changes to the Replacements table are not saved in this page. To save changes to the Replacements table at

project level you need to specify a custom location in the **User-defined replacements** section of the **AutoCorrect Dictionaries** preferences page (on page 131) and select **Project Options** from that preferences page instead.

Restore Defaults

Restores the options in this preferences page to their default values and **also deletes any changes you have made to the Replacements table** (on page 129).

AutoCorrect Dictionaries Preferences

To set the Dictionaries preferences for the *AutoCorrect* feature (on page 300), open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Editor > Edit Modes > Author > AutoCorrect > Dictionaries**. This page allows you to specify the location of the dictionaries that Oxygen JSON Editor uses for the *AutoCorrect* feature and the location for saving user-defined replacements.

The following options are available in this preferences page:

Dictionaries default folder

Displays the default location where the dictionaries that Oxygen JSON Editor uses for the *AutoCorrect* feature are stored.

Include dictionaries from

Selecting this option allows you to specify a location where you have stored *AutoCorrect* dictionaries that you want to include, along with the default ones.



Important:

Consider the following notes regarding this option:

- The *AutoCorrect* mechanism takes into account *AutoCorrect* dictionaries collected both from the default and custom locations and multiple dictionaries from the same language are merged (for example, **en_UK.dat** from the default location is merged with **en_US.dat** from a custom location).
- If you have a generic *AutoCorrect* dictionary file (one that just has a two-letter language code for its file name, such as **en.dat**) saved in either the default or custom location, the other more specific dictionaries (for example, **en_UK.dat** and **en_US.dat**) will not be merged and the existing generic dictionary will simply be used instead.
- You can use a custom suffix in the dictionary file name after the language code. For example, **en_US_synopsys.dat** or **en_synopsys.dat**.
- If the additional location contains a file with the same name as one from the default location, the file in the additional location takes precedence over the file from the default location.

How to add more dictionaries link

Use this link to open a topic in the Oxygen JSON Editor User Guide that explains how to [add dictionaries for the *AutoCorrect* feature \(on page 302\)](#).

Save user-defined replacements in the following location

Specifies the target where added, edited, or deleted replacements are saved. By default, the target is the application preferences folder, but you can also choose a custom location.



Tip:

To save changes to [the Replacement table \(in the **AutoCorrect** preferences page\) \(on page 129\)](#) at [project level \(on page 640\)](#), select a custom location for the **User-defined replacements** and select [Project Options \(on page 640\)](#) at the bottom of the page.

Related information

[Add Dictionaries for the AutoCorrect Feature \(on page 302\)](#)

Schema Design Preferences

Oxygen JSON Editor provides a [graphical schema design editor \(on page 396\)](#) to make editing XML Schema easier. To configure the **Schema Design** options, open the **Preferences** dialog box (**Options > Preferences**) [\(on page 74\)](#) 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 JSON Editor displays the content of documentation 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 JSON Editor opens the imported schema file so that you can edit it.
- **Never go to its definition** - The imported schema file is not opened and the component cannot be edited in place.
- **Always ask** - Oxygen JSON Editor asks if you want to open the imported schema file.

Zoom

Allows you to set the default zoom level of the schema diagram.

JSON Schema Properties Preferences

Oxygen JSON Editor lets you control which properties to display for JSON Schema components in the JSON Schema **Design** mode. To configure the JSON properties that are displayed, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Edit modes > Schema Design > JSON Schema 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 JSON 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 JSON 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.

Open Preferences

Oxygen JSON Editor lets you control how files are opened. To configure the options for opening documents, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Open**.

The following options are available:

Open each document in a tab next to the current one

When selected (default), each new document is opened in a tab next to the currently open tab. If not selected, each new document is opened in a tab at the end of the current tab stack.

Restore cursor position

Selected by default, it ensures that the last position of the cursor will be remembered when a document is re-opened. If this option is not selected, the cursor will always be positioned at the beginning of the document.

Lock local resources

When this option is selected and you open a file from the local file system or a shared network drive, Oxygen JSON Editor locks the file for the current user and the file becomes read-only for other users while the lock exists. Locked and read-only files have a lock icon (🔒) displayed on their editor tabs. Newly created files are *locked* when you first save them. If you select this option with files already opened in Oxygen JSON Editor, it will *lock* all the currently open files. If you deselect this option with files already opened, it will unlock them by deleting the

corresponding **.lock** files. When you try to save locked (read-only) files, a **Save As** dialog box will be displayed to avoid overwriting the initial resource.

Support for Special Characters section



Note:

The options in this section only affect the **Text** editing mode.

When bidirectional text, Asian languages, or other special characters are detected

You can choose how you want Oxygen JSON Editor to handle bidirectional text, Asian languages, or other special characters when they are detected in **Text** mode. 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** (default setting) - You will be prompted to choose whether or not to enable the support for special characters whenever they are detected in a newly opened document.

Disable special characters support for documents larger than (characters)

Enabling bidirectional text editing support can affect performance on large files. When this option is selected, bidirectional editing is disabled for files exceeding the specified size (even if the **Enable support for special characters** option ([on page 133](#)) is selected). The default limit is 300 MB. You can change it to 500 MB or 800 MB, but it is recommended that you always leave this option selected regardless of the limit that is set.

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 \(on page 308\)](#) for memory-intensive operations.

Show warning when loading large documents

Oxygen JSON Editor will warn you if you open a file that is bigger than the specified size.

Optimize loading for documents with lines longer than (Characters)

Line wrap is automatically performed for documents that contain lines that exceed the length specified in this text field. For a list of the restrictions applied to a document with long lines, see [Editing Documents with Long Lines \(on page 310\)](#).

Show warning when loading documents with long lines

When selected, Oxygen JSON Editor 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 [Documents with Long Lines \(on page 310\)](#).

Save Preferences

Oxygen JSON Editor lets you control how files are saved. To configure the options for saving documents, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Save**.

The following options are available:

Show "Save as" option to save newly created documents in the "New" document wizard

It is selected by default, but if you deselect this option, the [Save as option \(on page 226\)](#) will not be available in the [New Document wizard \(on page 225\)](#), so you will not have the ability to change the default name and path of the new file.

Safe save (only for local files)

In the unlikely event of a failure when attempting a **Save** action, this option provides increased protection from corruption of the original file. When this option is selected, it saves the content to a temporary file and if the save is unsuccessful, the editor preserves its unsaved state status.

Automatically save the document every

If selected, your documents are automatically saved after a preset time interval that is specified in the drop-down list.

On Save, make backup copy with extension (only for local files)

If selected, a backup copy is made when saving the edited document. This option is available only for local files (files that are stored on the local file system). The default backup file extension is **.bak**, but that can be changed in the text field.

Save auto-recover information every

If selected, your documents are automatically saved to a backup file after the time interval specified in the drop-down list.

Auto-recover file location

Specifies the location of the backup file for auto-recovery.

Validate document before saving

If selected, Oxygen JSON Editor runs a validation that checks your document for errors before saving it.

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.

Save all files before calling external tools

If selected, all files are saved before executing an external tool.

Automatically compile LESS to CSS when saving

If selected, when you save a LESS file it will automatically be compiled to CSS (deselected by default).



Important:

If this option is selected, when you save a LESS file, the CSS file that has the same name as the LESS file is overwritten without warning. Make sure all your changes are made in the LESS file. Do not edit the CSS file directly, as your changes might be lost.

Performance section

Clear undo buffer on save

If selected, Oxygen JSON Editor clears its undo buffer when you save a document. Thus, modifications made prior to saving the document cannot be undone. Select this option if you frequently encounter **out of memory** errors when editing large documents.

Format Preferences

This preferences page contains various formatting options that influence editing and formatting.

To configure the **Format** options, [open the Preferences dialog box \(Options > Preferences\)](#) (on page 74) and go to **Editor > Format**.

The following options are available:

Detect indent on open

If selected, Oxygen JSON Editor detects how a document is indented when it is opened. Oxygen JSON Editor uses a heuristic method of detection by computing a weighted average indent value from the initial document content. You can deselect this setting if the detected value does not work for your particular case and you want to use a fixed-size indent for all the edited documents. If this option is selected, Oxygen JSON Editor detects the following:

- When TAB characters are used to indent content, the size of the TAB characters is used for the indent size.
- Otherwise, the detected size of SPACE characters is used for the indent size.

**Tip:**

If you want to minimize the formatting differences created by the **Format and Indent** operation in a document edited in the **Text** edited mode, make sure that both the **Detect indent on open** and **Detect line width on open** ([on page 137](#)) options are selected.

Use zero-indent, if detected

By default, if no indent was detected in the document, the fixed-size indent is used. Select this option if all of your documents 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** ([on page 135](#)) option.

Indent size

The meaning of this setting depends on the following:

- If the **Detect indent on open option** ([on page 135](#)) is selected and TAB characters are detected at the beginning of the line, the *indent size* is the width of a TAB character. Otherwise, the *indent size* value is ignored and Oxygen JSON Editor uses the number of detected SPACE characters.
- If the **Indent with tabs option** ([on page 136](#)) is selected, the *indent size* is the width of a TAB character.
- If neither of these options are selected, the *indent size* is the number of SPACE characters used for indenting the text lines.

For information about when this setting is used, see [Where Indent Size and Line Width Settings are Used in Oxygen JSON Editor](#) ([on page 138](#)).

Indent on enter

If selected, when you press **Enter** to insert a line break in the **Text** editing mode, an indentation will be added to the new line.

Enable smart enter

If selected, when you press the **Enter** key between a start and an end XML tag in the **Text** editing mode, the cursor is placed in an indented position on the empty line formed between the start and end tag.

Format and indent the document on open

If selected, an XML document is formatted and indented before opening it in Oxygen JSON Editor.

**Note:**

Some specialized types of XML documents do not benefit from this feature, including Relax NG, XSD, XSL, and Ant. However, the feature is available for some non-XML types of documents, such as CSS and JSON.

Detect line width on open

If selected, Oxygen JSON Editor automatically detects the line width when the document is opened.

Hard line wrap (Limit to "Line width - Format and Indent")

If selected, when typing content in the **Text** editing mode and the maximum line width is reached, a line break is automatically inserted.

Line width - Format and Indent

Defines the number of characters after which the **Format and Indent** (pretty-print) action performs hard line-wrapping. For example, if set to 100, after a **Format and Indent** action, the longest line will have a maximum of 100 characters. This setting is also used when saving XML content edited in the **Author** editing mode.

**Note:**

To avoid having an indent that is longer than the line width setting and without having sufficient space available for the text content, the indent limit is actually set at half the value of the **Line width - Format and Indent** setting. The remaining space is reserved for text.

For information about when this setting is used, see [Where Indent Size and Line Width Settings are Used in Oxygen XML Editor \(on page 138\)](#).

Clear undo buffer before Format and Indent

The **Format and Indent** operation can be *undone*, but if used intensively, a considerable amount of the memory allocated for Oxygen JSON Editor will be used for storing the undo states. If this option is selected, Oxygen JSON Editor empties the undo buffer before doing a **Format and Indent** operation. This means you will not be able to undo any changes you made before the format and indent operation. Select this option if you encounter out of memory problems (**OutOfMemoryError**) when performing the **Format and Indent** operation.

Where Indent Size and Line Width Settings are Used in Oxygen JSON Editor

The values set in the **Indent Size** and **Line Width - Format and Indent** options are used in various places in the application, including the following:

- When the **Format and Indent** action is used in the **Text** editing mode.
- When you press **Enter** to break a line in the **Text** editing mode.
- When the **Hard line wrap (Limit to "Line width - Format and Indent")** option is selected and the maximum line width is reached while editing in the **Text** mode.
- When the XML is serialized by saving content in the **Author** editing mode.

Resources

For more information about the formatting options offered by Oxygen JSON Editor, watch our video demonstration:

<https://www.youtube.com/embed/1plmdN0Cfso>

CSS Preferences

Oxygen JSON Editor can format and indent your CSS files. To configure the **CSS** formatting options, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Format > CSS**.

The following options control how your CSS files are formatted and indented:

Class body on new line

If selected, the *class* body (including the curly brackets) is placed on a new line. This option is not selected by default.

Indent class content

When selected (default state), the *class* content is indented.

Add space before the value of a CSS property

When selected (default state), whitespaces are added between the `:` (colon) and the value of a style property.

Add new line between classes

If selected, an empty line is added between two classes. This option is not selected by default.

Preserve empty lines

When selected (default state), the empty lines from the CSS content are preserved.

Allow formatting embedded CSS

When selected (default state), CSS content that is embedded in XML is also formatted when the XML content is formatted.

JavaScript Preferences

To configure the **JavaScript** format options, open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Editor > Format > JavaScript**.

The following options control the behavior of the **Format and Indent** action:

- **Start curly brace on new line** - Opening curly braces start on a new line.
- **Preserve empty lines** - Empty lines in the JavaScript code are preserved. This option is selected by default.
- **Allow formatting embedded JavaScript** - Applied only to XHTML documents, this option allows Oxygen JSON Editor to format embedded JavaScript code. This option is selected by default.

Content Completion Preferences

Oxygen JSON Editor provides a *Content Completion Assistant* (on page 638) 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**) (on page 74) and go to **Editor > Content Completion**. These options control how the *Content Completion Assistant* works.

The following options are available:

Auto close the last opened tag

When selected, Oxygen JSON Editor automatically closes the last open tag when you type `</`.

Automatically rename/delete/comment matching tags

If you rename, delete, or comment out a start tag, Oxygen JSON Editor automatically renames, deletes, or comments out the matching end tag.



Note:

If you select **Toggle comment** for multiple starting tags and the matching end tags are on the same line as other start tags, the end tags are not commented.

Enable content completion

Toggles the content completion feature on or off.

Consider subsequent sibling elements

When this option is selected (default), the subsequent sibling elements of the current element are taken into account when using the *Content Completion Assistant*. For example, in DITA, if you invoke the content completion before an already inserted required element (e.g. a `<title>` element), the content completion mechanism will not offer a proposal to insert a title (since it was already inserted).

Close the inserted element

When you choose an entry from the *Content Completion Assistant* list of proposals, Oxygen JSON Editor inserts both start and end tags. The following additional options are available with regard 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 JSON Editor inserts the required elements specified in the DTD, XML Schema, or RELAX NG schema that is associated with the edited XML document.
 - **Add optional content** - If selected, Oxygen JSON Editor inserts the optional elements specified in the DTD, XML Schema, or RELAX NG schema.
 - **Add first Choice particle** - If selected, Oxygen JSON Editor inserts the first **choice** particle specified in the DTD, XML Schema, or RELAX NG schema.

Case sensitive search

When selected, the search in the *Content Completion Assistant* is case-sensitive when you type a character ('a' and 'A' are different characters).



Note:

This option is ignored when the current language itself is not case-sensitive. For example, the case is ignored in the CSS language.

Position cursor between tags

When selected, Oxygen JSON Editor automatically moves the cursor between the start and end tag after inserting the element. This only applies to:

- Elements with only optional attributes or no attributes at all.
- Elements with required attributes, but only when the **Insert the required attributes** option (on page 140) is not selected.

Show all entities

Oxygen JSON Editor displays a list with all the internal and external entities declared in the current document when you type the start character of an entity reference (for example, &).

Insert the required attributes

Oxygen JSON Editor inserts automatically the required attributes taken from the DTD or XML Schema.

Insert the fixed attributes

If selected, Oxygen JSON Editor automatically inserts any **FIXED** attributes from the DTD or XML Schema for an element inserted with the help of the *Content Completion Assistant*.

Show recently used items

When selected, Oxygen JSON Editor remembers the last inserted items from the *Content Completion Assistant* window. The number of items to be remembered is limited by the **Maximum number of recent items shown** list box. These most frequently used items are displayed on the top of the content completion window and are separated from the rest of the suggestions by a thin gray line .

Maximum number of recent items shown

Specifies the limit for the number of recently used items presented at the top of the *Content Completion Assistant* window.

Learn attributes values

When selected, Oxygen JSON Editor learns the attribute values used in a document.

Learn on open document

Oxygen JSON Editor automatically learns the document structure when the document is opened.

Learn words (Dynamic Abbreviations, available on Ctrl+Space)

When selected, Oxygen JSON Editor learns the typed words and makes them available in a content completion fashion by pressing **Ctrl + Space** on your keyboard;



Note:

For the words to be learned, they need to be separated by space characters.

Activation delay of the proposals window (ms)

Delay in milliseconds from the last key press until the *Content Completion Assistant* is displayed.

XPath Preferences

Oxygen JSON Editor provides content-completion support for XPath expressions. To configure the options for the content completion in XPath expressions, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Content Completion > XPath**.

The following options are available:

- **Enable content completion for XPath expressions** - Enables the *Content Completion Assistant (on page 638)* in XPath expressions that you enter in the `@match`, `@select`, and `@test` XSL attributes.
 - **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.

- **Function signature window background** - Specifies the background color of the tooltip window.
- **Function signature window foreground** - Specifies the foreground color of the tooltip window.

JavaScript Preferences

Oxygen JSON Editor can provide content completion suggestions when you are writing JavaScript files. To configure content completion support for JavaScript, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) 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 JSON Editor to include components (object names, properties, functions, and variables) collected from the built-in JavaScript library files when making suggestions.

Use defined libraries

Oxygen JSON Editor can also use JavaScript libraries to when making suggestions. List the paths (URLs) of any JavaScript files you want Oxygen JSON Editor to use when making suggestions.



Note:

The paths can contain [editor variables \(on page 197\)](#) such as `${pdu}`, or `${oxygenHome}`. You can make these paths relative to the project directory or installation directory.

JSON Preferences

Oxygen JSON Editor can provide content completion suggestions when you are editing JSON files. To configure content completion support for JSON, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Content Completion > JSON**. You can configure the following options:

Generate required content

When invoking content completion over JSON files, all contextual required content is automatically generated according to the specifications from the associated JSON Schema.

Generate optional properties

If selected, optional properties that are defined in the associated JSON Schema will be added when using content completion in JSON files.

Generate additional content

If selected, additional properties (or additional items for arrays) that are defined in the associated JSON Schema will be added when using content completion in JSON files.

YAML Preferences

Oxygen JSON Editor can provide content completion suggestions when you are editing YAML files. To configure content completion support for YAML, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Content Completion > YAML**. You can configure the following options:

Generate required content

When invoking content completion over YAML files, all contextual required content is automatically generated according to the specifications from the associated JSON schema.

Property value

You can specify the way the values of the properties are generated. The following options are available:

- *None* - Assigns empty values for properties (a template file will be generated). This is the default value.
- *Default* - Assigns the name of the property as the value (for strings) or assigns the specified minimum value (for numbers).
- *Random* - Assigns random values according to schema restrictions.

Generate optional properties

If selected, optional properties that are defined in the associated JSON schema will be added when using content completion in YAML files.

Generate additional content

If selected, additional properties (or additional items for arrays) that are defined in the associated JSON schema will be added when using content completion in YAML files.

Annotations Preferences

Certain types of schemas can include annotations that document the various elements and attributes that they define. Oxygen JSON Editor can display these annotations when offering content completion suggestions. To configure the **Annotations** preferences, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Content Completion > Annotations**.

The following options are available:

Show annotations in Content Completion Assistant

If selected, Oxygen JSON Editor displays the schema annotations of an element, attribute, or attribute value currently selected in the [Content Completion Assistant \(on page 638\)](#) proposals list.

Show annotations in tooltip

If selected, Oxygen JSON Editor displays the annotation of elements and attributes as a tooltip when you hover over them with the cursor in the editing area or in the **Elements** view. If not selected, tooltips are disabled in all modes.

Show annotation in HTML format, if possible

This option allows you to view the annotations associated with an element or attribute in HTML format. If this option is not selected, the annotations are converted and displayed as plain text.

Prefer DTD comments that start with "doc:" as annotations

To address the lack of dedicated annotation support in DTD documents, Oxygen JSON Editor recommends prefixing with the `doc:` particle all comments intended to be shown to the developer who writes an XML validated against a DTD schema.

If this option is selected, Oxygen JSON Editor uses the following mechanism to collect annotations:

- If at least one `doc:` comment is found in the entire DTD, only comments of this type are displayed as annotations.
- If no `doc:` comment is found in the entire DTD, all comments are considered annotations and displayed as such.

If not selected, all comments, regardless of their type, are considered annotations and displayed as such.

Use all Relax NG annotations as documentation

If 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 (on page 638)* window. 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.

Code Templates Preferences

Code templates are code fragments that can be inserted at the current editing position. Oxygen JSON Editor includes a set of built-in templates for CSS, LESS, Schematron, XSL, XQuery, JSON, HTML, and XML Schema document types.

To configure **Code Templates**, open the **Preferences** dialog box (**Options > Preferences**) (on page 74) 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** - [Optional] 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 (on page 638)*. 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** - [Optional] If you want to assign a shortcut key that can be used to insert the code template, place the cursor in the **Shortcut key** field and press the desired key combination on your keyboard. Use the **Clear** button if you make a mistake. If the **Enable platform-independent shortcut keys** checkbox is selected, the shortcut is platform-independent and the following modifiers are used:
 - **M1** represents the **Command** key on macOS, and the **Ctrl** key on other platforms.
 - **M2** represents the **Shift** key.
 - **M3** represents the **Option** key on macOS, and the **Alt** key on other platforms.
 - **M4** represents the **Ctrl** key on macOS, and is undefined on other platforms.
- **Content** - Text box where you define the content that is used when the code template is inserted. An *editor variable (on page 197)* can be inserted in the text box using the  **Insert Editor Variables** button.

Edit

Opens the **Code template** dialog box and allows you to edit an existing code template. You can edit the following fields:

- **Description** - [Optional] 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 (on page 638)*. HTML markup can be used for better rendering.
- **Shortcut key** - [Optional] If you want to assign a shortcut key that can be used to insert the code template, place the cursor in the **Shortcut key** field and press the desired key combination on your keyboard. Use the **Clear** button if you make a mistake. If the **Enable platform-independent shortcut keys** checkbox is selected, the shortcut is platform-independent and the following modifiers are used:
 - **M1** represents the **Command** key on macOS, and the **Ctrl** key on other platforms.
 - **M2** represents the **Shift** key.
 - **M3** represents the **Option** key on macOS, and the **Alt** key on other platforms.
 - **M4** represents the **Ctrl** key on macOS, and is undefined on other platforms.
- **Content** - Text box where you define the content that is used when the code template is inserted. An *editor variable (on page 197)* can be inserted in the text box using the  **Insert Editor Variables** button.

Duplicate

Creates a duplicate of the currently selected code template.

Delete

Deletes the currently selected code template. This action is not available for the built-in code templates.

Export

Exports a file with code templates.

Import

Imports a file with code templates that was created by the **Export** action.

You can use the following [editor variables \(on page 197\)](#) 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 currently selected text content in the currently 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', @id)}`** - 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.
- **'default-value'** - Optional parameter. Provides a default value.
- **@id** - Optional parameter. Used for identifying the variable to reuse the answer using the **`\${answer(@id)}`** editor variable.
- **type** - Optional parameter (defaults to **generic**), with one of the following values:



Note:

The title of the dialog box will be determined by the type of parameter and as follows:

- For *url* and *relative_url* parameters, the title will be the name of the parameter and the value of the *'message'*.
- For the other parameters listed below, the title will be the name of that respective parameter.
- If no parameter is used, the title will be "Input".



Notice:

Editor variables that are used within a parameter of another editor variable must be escaped within single quotes for them to be properly expanded. For example:

```
${ask( 'Provide a date',generic, '${date(yyyy-MM-dd'T'HH:MM)'}' )}
```

Parameter	
generic (default)	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 <i>Hello world!</i> message displayed. ▪ <code>\${ask('Hello world!', generic, 'Hello again!')}</code> - The dialog box has a <i>Hello world!</i> message displayed and the value displayed in the input box is <i>Hello again!</i>.
url	Format: <code>\${ask('message', url, 'default_value')}</code>
	Description: Input is considered a URL. Oxygen JSON Editor checks that the provided URL is valid.
	Example: ▪ <code>\${ask('Input URL', url)}</code> - The displayed dialog box has the name <i>Input URL</i>. The expected input type is URL. ▪ <code>\${ask('Input URL', url, 'http://www.example.com')}</code> - The displayed dialog box has the name <i>Input URL</i>. The expected input type is URL. The input field displays the default value <i>http://www.example.com</i>.
relative_url	Format: <code>\${ask('message', relative_url, 'default')}</code>
	Description: Input is considered a URL. This parameter provides a file chooser, along with a text field. Oxygen JSON Editor 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 JSON Editor will transform it into an absolute URL.
	Example: <code>\${ask('File location', relative_url, 'C:/example.txt')}</code> - The dialog box has the name <i>File location</i> . The URL inserted in the input box is made relative to the currently edited document location.
password	Format: <code>\${ask('message', password, 'default')}</code>

Parameter	
	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 abcd value.
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 <i>rendered_value</i> values. Choosing such a value will return its associated value (<i>real_value</i>).  Note: The list of '<i>real_value</i>':'<i>rendered_value</i>' pairs can be computed using <code>\${xpath_eval()}</code>.  Note: The '<i>default</i>' parameter specifies the default-selected value and can match either a key or a value. Example: ▪ <code>\${ask('Operating System', combobox, ('win':'Microsoft Windows'; 'macos':'macOS'; 'lnx':'Linux/UNIX'), 'macos')}</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 <i>osx</i> key. However, the same result could be obtained if the default value is indicated by <i>macOS</i>, as in the following example: <code>\${ask('Operating System', combobox, ('win':'Microsoft Windows'; 'macos':'macOS'; 'lnx':'Linux/UNIX'), 'macOS')}</code>

Parameter	
	▪ <code>\${ask('Mobile OS', combobox, ('ios':'iOS';'and':'Android'), 'Android')}</code> ▪ <code>\${ask('Mobile OS', combobox, (\$xpath_eval(for \$pair in (['ios', 'iOS'], ['and', 'Android']) return "" \$pair?1 ":" \$pair?2 ";")), 'ios')}</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 <i>rendered_value</i> values. Choosing such a value will return its associated real value (<i>real_value</i>) or the value inserted when you edit a list entry.  Note: The list of 'real_value':'rendered_value' pairs can be computed using <code>\$xpath_eval()</code>.  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';'macos':'macOS';'lnx':'Linux/UNIX'), 'macos')}</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. ▪ <code>\${ask('Operating System', editable_combobox, (\$xpath_eval(for \$pair in (['win', 'Microsoft Windows'], ['macos', 'macOS'], ['lnx', 'Linux/UNIX']) return "" \$pair?1 ":" \$pair?2 ";")), 'ios')}</code>
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 <i>rendered_value</i> and will return an associated <i>real_value</i>.

Parameter	
	 Note: The list of <i>'real_value': 'rendered_value'</i> pairs can be computed using <code>\${xpath_eval()}</code>.  Note: The <i>'default'</i> parameter specifies the default-selected value and can match either a key or a value.
	Example: ▪ <code>\${ask('Operating System', radio, ('win': 'Microsoft Windows'; 'macos': 'macOS'; 'lnx': 'Linux/UNIX'), 'macos')}</code> - The dialog box has the name <i>'Operating System'</i>. The radio button group allows you to choose between the three operating systems.  Note: In this example, <i>macOS</i> is the default-selected value and if selected, it would return <i>macos</i> for the output. ▪ <code>\${ask('Operating System', radio, (\$xpath_eval(for \$pair in (['win', 'Microsoft Windows'], ['macos', 'macOS'], ['lnx', 'Linux/UNIX']) return "" \$pair?1 ":" \$pair?2 ";" })), 'ios')}</code>

- **`${timeStamp}`** - The timestamp, which is the current time in Unix format. For example, it can be used to save transformation results in multiple output files on each transformation.
- **`${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 the extension and parent folder. The current file is the one currently open and selected.
- **`${cfne}`** - Current file name with extension. The current file is the one currently open and selected.
- **`${cf}`** - Current file as file path, that is the absolute file path of the currently edited document.
- **`${cfd}`** - Current file folder as file path, that is the path of the currently edited document up to the name of the parent folder.
- **`${frameworksDir}`** - The path (as file path) of the **frameworks** directory. When used to define references inside a framework configuration, it expands to the parent folder of that specific framework folder. Otherwise, it expands to the main **frameworks** folder defined in the **Document Type Association > Locations** preferences page.

- **`${pd}`** - The file path to the folder that contains the current project file (`.xpr`).
- **`${oxygenInstallDir}`** - Oxygen JSON Editor installation folder as file path.
- **`${homeDir}`** - The path (as file path) of the user home folder.
- **`${pn}`** - Current project name.
- **`${env(VAR_NAME)}`** - Value of the `VAR_NAME` environment variable. The environment variables are managed by the operating system. If you are looking for Java System Properties, use the **`${system(var.name)}`** editor variable.
- **`${system(var.name)}`** - Value of the `var.name` Java System Property. The Java system properties can be specified in the command-line arguments of the Java runtime as `-Dvar.name=var.value`. If you are looking for operating system environment variables, use the **`${env(VAR_NAME)}`** editor variable instead.
- **`${date(pattern)}`** - Current date. The allowed patterns are equivalent to the ones in the [Java SimpleDateFormat class](#). **Example:** `yyyy-MM-dd`.

**Note:**

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

Syntax Highlight Preferences

Oxygen JSON Editor supports syntax highlighting in the **Text** mode editors for numerous types of documents, including XML, XHTML, JavaScript, XQuery, XPath, PHP, PowerShell, CSS, LESS, Markdown, Text, DTD, RNC, Java, JSON, and more.

To configure syntax highlighting, open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Editor > Syntax Highlight**.

To set syntax colors for a language, expand the listing for that language in the top panel to show the list of syntax items for that type of document. Use the color and style selectors to change how each syntax item is displayed. The results of your changes are displayed in the **Preview** panel. If you do not know the name of the syntax token that you want to configure, click that token in the **Preview** area to select it.

**Note:**

All default color sets come with a high-contrast variant that is automatically used when you switch to a black-background or white-background high-contrast theme in your Windows operating system settings. The high-contrast theme will not overwrite any default color you set in **Editor > Syntax Highlight** preferences page.

The settings for XML documents are also used in XSD, XSL, RNG documents and the **Preview** area has a separate tab for each of them when **XML** is selected in the top pane.

The **Enable nested syntax highlight** option controls whether or not content types that are nested in the same file (such as PHP, JS, or CSS scripts inside an HTML file) are highlighted according to the color schemes defined for each content type.

Elements/Attributes by Prefix Preferences

Oxygen JSON Editor allows you to specify syntax highlighting colors for elements and attributes with specific namespace prefixes. To configure the **Elements/Attributes by Prefix** preferences, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Syntax Highlight > Elements/Attributes by Prefix**.

To change the syntax coloring for a specific namespace prefix, choose the prefix from the list, or add a new one using the **New** button, and use the color and style selectors to set the syntax highlighting style for that namespace prefix.



Note:

Syntax highlighting is based on the literal namespace prefix, not the namespace that the prefix is bound to in the document.

If you only want the prefix (and not the whole element or attribute name) to be styled with a particular color, select the **Draw only the prefix with a separate color** option.

Mark Occurrences Preferences

This preferences page specifies which types of files will have the Highlight IDs Occurrences feature activated. To configure these options, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Mark Occurrences**:

The following options are available in this preferences page:

Highlight component occurrences in the current file for:

- **XML files** - Activates the Highlight IDs Occurrences feature in XML files.
- **XSLT files** - Activates the Highlight IDs Occurrences feature in XSLT files.
- **XML Schema files** - Activates the Highlight IDs Occurrences feature in XSD files.
- **WSDL files** - Activates the Highlight IDs Occurrences feature in WSDL files.
- **RNG files** - Activates the highlight component occurrences feature in RNG files.
- **Schematron files** - Activates the Highlight IDs Occurrences feature in Schematron files.
- **Ant files** - Activates the Highlight IDs Occurrences feature in Ant files.

Declaration highlight color

Allows you to choose the color to be used for highlighting component declarations.

Reference highlight color

Allows you to choose the color to be used for highlighting component references.

Document Validation Preferences

To configure document validation options, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Document Validation**. This page contains preferences for configuring how a document is checked for both well-formedness and validation errors.

The following options are available:

Maximum number of validation highlights

If a validation generates more errors than the number specified in this option, only the errors up to this number are highlighted in the editor panel and on the stripe that is displayed at the right side of the editor panel. This option applies to both automatic validation and manual validation.

Validation fatal error highlight color

The color used to highlight fatal validation errors in the document.

Validation error highlight color

The color used to highlight validation errors in the document.

Validation warning highlight color

The color used to highlight validation warnings in the document.

Validation info highlight color

The color used to highlight validation info messages in the document.

Validation success color

The color used to highlight the success indicator of the validation operation in the vertical ruler bar.

Always show validation status

If this option is selected, the current validation error or warning is always visible in the message line at the bottom of the editor panel. This is useful when the **Enable automatic validation** option is selected and the vertical scroll bar changes position due to an error message being displayed.

Enable automatic validation

This causes the validation to be automatically executed in the background as the document is modified in Oxygen JSON Editor.

Delay after the last key event (s)

The period of keyboard inactivity before starting a new validation (in seconds).

At the bottom of the preferences page you can [choose whether or not the saved options will be shared with other users by selecting Global or Project storage options \(on page 187\)](#).

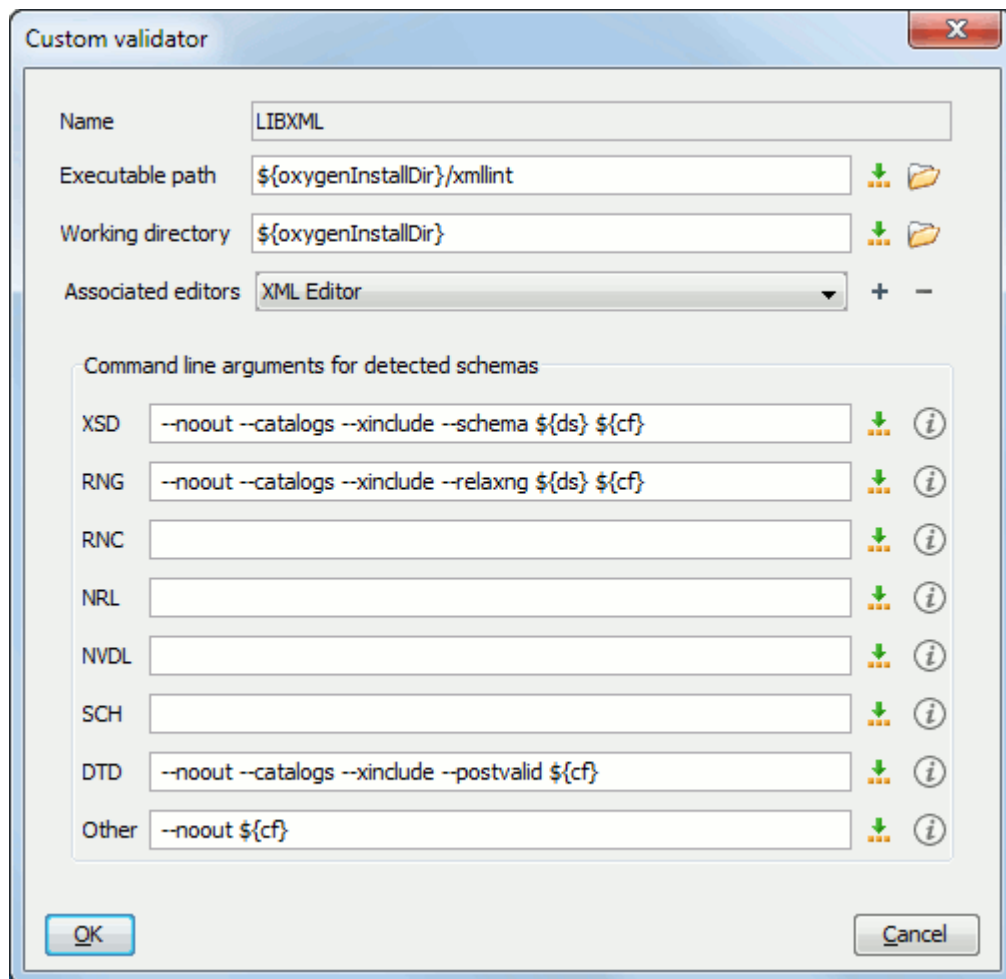
Custom Validation Engines Preferences

As the name implies, the **Custom Validation Engines** preferences page displays the list of custom validation engines that can be associated to a particular editor and used for validating documents. To access this page,

open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Editor > Document Validation > Custom Validation Engines**.

If you want to add a new custom validation tool or edit the properties of an existing one, you can use the **Custom Validator** dialog box displayed by pressing the **New** or **Edit** button.

Figure 11. 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** (on page 197) 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** (on page 197) 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 (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 \(on page 197\)](#):

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

Related information

[Editor Variables \(on page 197\)](#)

Increasing the Stack Size for Validation Engines

To prevent the appearance of a **StackOverflowException** error, use one of the following methods:

- Use the **com.oxygenxml.stack.size.validation.threads** property to increase the size of the stack for validation engines. The value of this property is specified in bytes. For example, to set a value of one megabyte specify $1 \times 1024 \times 1024 = 1048576$. For information about how to setup the system property on the JVM, see [Setting a Java Virtual Machine Parameter when Launching Oxygen JSON Editor \(on page 211\)](#).
- Increase the value of the **-Xss** parameter.



Note:

Increasing the value of the **-Xss** parameter affects all the threads of the application.

Related information

[Setting a Java Virtual Machine Parameter when Launching Oxygen JSON Editor \(on page 211\)](#)

Ignored Validation Problems Preferences

Some validation issues include a Quick Fix proposal that instructs the application to ignore that type of validation problem. The ignored problems are then listed in the **Ignored Validation Problems** preferences

page. To access this page, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Document Validation > Ignored Validation Problems**.

The **Ignored Validation Problems** preferences page includes the following:

Enable support for ignoring validation problems

If this option is selected, the support for ignoring certain validation problems is enabled.

Ignored problems table

The table includes an entry for each validation problem that has been ignored. The columns in the table include information about the **Severity**, **Problem ID**, **Message**, and **System ID**. You can delete an entry by selecting it and clicking the **Delete** button at the bottom of the table. The deleted problem is no longer ignored.



Tip:

Changes made in this preferences page can be [saved at project level \(on page 188\)](#) so that you can easily share your ignored problems configuration with others.

Spell Check Preferences

Oxygen JSON Editor provides support for spell checking in the **Text** mode and **Author** mode. To configure the **Spell Check** options, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Spell Check**.

The following options are available:

Automatic spell check

This option is not selected by default. When selected, Oxygen JSON Editor automatically checks the spelling as you type and highlights misspelled words in the document.

Select editors

You can select which editors (and therefore which file types) will automatically be spell checked. File types such as CSS and DTD are excluded by default since automatic spell checking is not usually helpful in these types of files.

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 \(on page 288\)](#).

Use "lang" and "xml:lang" attributes

When this option is selected, the contents of an element with one of the `@lang` or `@xml:lang` attributes is checked in that language. Choose between the following two options for instances when these attributes are missing:

- **Use the default language** - If the `@lang` and `@xml:lang` attributes are missing, the selection in the **Default language list (on page 156)** is used.
- **Do not check** - If the `@lang` and `@xml:lang` attributes are missing, the element is not checked.

XML spell checking in section

You can choose to check the spelling inside the following XML items:

- **Comments**
- **Processing instructions**
- **Attribute values**
- **Text**
- **CDATA**

Options section

This section includes the following other options:

Check capitalization

When selected, the spell checker reports detected capitalization errors.



Note:

When such problems are reported, they cannot be learned and ignored by the application as words stored in dictionaries, term lists, and the list of learned words are not handled as case-sensitive.

Check punctuation

When selected, the spell checker checks punctuation. Misplaced white space and unusual sequences, such as a period following a comma, are highlighted as errors.

Ignore mixed case words

When selected, the spell checker does not check words containing mixed case characters (for example, *SpellChecker*).

Ignore acronyms

Available only for the **Hunspell Spell Checker**. When selected, acronyms are not reported as errors.

Ignore words with digits

When selected, the spell checker does not check words containing digits (for example, *b2b*).

Ignore duplicates

When selected, the spell checker does not signal two successive identical words as an error.

Ignore URL

When selected, the spell checker ignores words recognized as URLs or file names (for example, *www.oxygenxml.com* or *c:\boot.ini*).

Allow compounds words

When selected, all words formed by concatenating multiple legal words with hyphens or underscores are accepted.

Allow file extensions

When selected, the spell checker accepts any word ending with recognized file extensions (for example, *myfile.txt* or *index.html*).

Ignore elements section

You can use the **Add** and **Remove** buttons to configure a list of element names or XPath expressions to be ignored by the spell checker. The following restricted set of XPath expressions are supported:

- '/' and '/' separators
- '*' wildcard

An example of an allowed XPath expression is: */a*/b*.

AutoCorrect options link

Use this link to navigate to the [AutoCorrect preferences page \(on page 128\)](#).

Spell Check Dictionaries Preferences

To set the Dictionaries preferences, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Spell Check > Dictionaries**. This page allows you to configure the dictionaries (*.dic* files) and term lists (*.tdi* files) that Oxygen JSON Editor uses and to choose where to save new learned words.

The following options are valid when Oxygen JSON Editor uses the Hunspell spell checking engine:

Dictionaries and term lists default folder

Displays the default location where the dictionaries and term lists that Oxygen JSON Editor uses are stored.

Include dictionaries and term list from

Selecting this option allows you to specify a location where you have stored dictionaries and term lists that you want to include, along with the default ones.

**Important:**

Consider the following notes regarding this option:

- The spell checker takes into account dictionaries and term lists collected both from the default and custom locations and multiple dictionaries and term lists from the same language are merged (for example, **en_UK.dic** from the default location is merged with **en_US.dic** from a custom location).
- If you have a generic dictionary file (one that just has a two-letter language code for its file name, such as **en.dic**) saved in either the default or custom location, the other more specific dictionaries (for example, **en_UK.dic** and **en_US.dic**) will not be merged and the existing generic dictionary will simply be used instead.
- If the additional location contains a file with the same name as one from the default location, the file in the additional location takes precedence over the file from the default location.

How to add more dictionaries and term lists link

Use this link to open a topic in the Oxygen JSON Editor User Guide that explains how to [add more dictionaries and term lists \(on page 292\)](#).

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.

**Note:**

Words stored in dictionaries, term lists, and the list of learned words are not handled as case-sensitive. Therefore, you do not need to include both uppercase and lowercase versions of the words.

Related information

[Adding Custom Spell Check Dictionaries \(on page 292\)](#)

[Adding Custom Spell Check Term Lists \(on page 294\)](#)

Print Preferences

Oxygen JSON Editor lets you configure how files are printed out of the editor. Note that these settings cover how files are printed directly from Oxygen JSON Editor itself, not how they are printed after the XML source has been transformed by a publishing stylesheet. To configure the **Print** options, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Print**.

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 currently edited document represented as URL.
- **\${cfne}** - Current file name with extension. The current file is the one currently open 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**.

CSS Validator Preferences

To configure the **CSS Validator** preferences, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **CSS Validator**.

You can configure the following options for the built-in **CSS Validator** of Oxygen JSON Editor:

- **Profile** - Selects one of the available validation profiles: **CSS 1**, **CSS 2**, **CSS 2.1**, **CSS 3**, **CSS 3 + SVG**, **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 \(on page 352\)](#) 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. The following vendor extensions are specified as ignored by default: **-ro-*** (*PDFreactor*), **-ah-*** (*Antenna House*), **prince-*** (*Prince*). As wildcards you can use:
 - ***** to match any string.
 - **?** to match any character.
- **Recognize browser CSS extensions (also applies to content completion)** - If selected, Oxygen JSON Editor recognizes browser-specific CSS properties (no validation is performed). The [Content Completion Assistant \(on page 638\)](#) lists these properties at the end of its list, prefixed with the following particles:
 - **-moz-** for *Mozilla*.
 - **-ms-** for *Edge*.
 - **-o-** for *Opera*.
 - **-webkit-** for *Safari/Webkit*.

SVN Preferences

To configure the options for the SVN client tool, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) 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 12. SVN Preferences Panel

SVN

☒ Allow unversioned obstructions

☐ Use unsafe copy operations

Results Console

Maximum number of lines: 1000

Annotations View

Annotation highlight color: [Light Green]

Revision Graph

☒ Enable log caching

Clear cache (0.0 MB)

The following SVN options can be configured in this preferences page:

Enable symbolic link support (*available only on macOS and Linux*)

Apache Subversion™ can put a symbolic link under version control, via the usual SVN *add* command. The Subversion repository has no internal concept of a symbolic link. It stores a versioned symbolic link as an ordinary file with a `svn:special` property attached. On Unix/Linux, the SVN client sees the property and translates the file into a symbolic link in the working copy. If the symbolic link support is disabled, the versioned symbolic links appear as a text file instead of symbolic link.



Note:

Windows file systems have no symbolic links, so a Windows client will not do any such translation and the object appears as a normal file.



Important:

It is recommended to disable symbolic links support if you do not have versioned symbolic links in your repository, since the SVN operations will work faster. However, you should not disable this option when you do have versioned symbolic links in repository. In that case a workaround would be to reference the working copy by its real path, instead of a path that includes a symbolic link.

Allow unversioned obstructions

Controls how to handle a situation where working copy resources are ignored / unversioned when performing an update operation and incoming files (from the repository) with the same name and location intersect with those being ignored / unversioned. If the option is selected, the incoming items will become BASE revisions of the ones already present in the working copy, and those present will be made versioned resources and will be marked as modified (exactly as if the user first made the update operation and then modified the files). If the option is not selected,

the update operation will fail when encountering files in this situation, possibly leaving other files not updated. By default, this option is selected.

Use unsafe copy operations

Sometimes when the working copy is accessed through Samba and the SVN client cannot make a safe copy of the committed file due to a delay in getting a write permission, the result is that the committed file will be saved with zero length (the content is removed) and an error will be reported. In this case, this option should be selected so that the SVN client does not try to make the safe copy.

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 \(Options > Preferences\)](#) ([on page 74](#)) 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 (whether or not to upgrade the format of the current working copy).
- **Never upgrade** - Older format working copies are left untouched. No attempt to upgrade the format is made.

**Note:**

SVN 1.6 and older working copies still need to be upgraded before loading them.

Enable working copy caching

If selected, the content of the working copies is cached for refresh operations.

Automatically refresh the working copy

If selected, the working copy is refreshed from cache. Only the new changes (modifications with a date/time that follows the last refresh operation) are refreshed from disk. This option is not selected by default.

Allow moving/renaming mixed revision directories

If selected, Oxygen JSON Editor will allow you to move or rename a directory even if its child items have a different revision. Otherwise, an error message is displayed when there are multiple revisions to avoid unnecessary conflicts. It is recommended to leave this option deselected and to **Update** the subtree to a single revision before moving or renaming it.

When synchronizing with repository

The action that will be executed automatically after the **Synchronize** action. The possible actions are:

- **Always switch to 'Modified' mode** - The **Synchronize** action is followed automatically by a switch to **Modified** mode of **Working Copy** view, if **All Files** mode is currently selected.
- **Never switch to 'Modified' mode** - Keeps the currently selected view mode unchanged.
- **Always ask** - The user is always asked if they want to switch to **Modified** mode.

Application global ignores

Allows you to set file patterns that may include the `*` and `?` wildcards for unversioned files and folders that must be ignored when displaying the working copy resources in the **Working Copy** view. These patterns are case-sensitive. For example, `*.txt` matches `file.txt`, but does not match `file.TXT`.

Diff Preferences

To configure the SVN Diff options, open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Diff**.

The following option is available:

Compare With External Application

Specifies an external application to be launched for compare operations in the following cases:

- When two history revisions are compared.
- When the working copy file is compared with a history revision.
- When a conflict is edited.

The parameters **\${firstFile}** and **\${secondFile}** specify the positions of the two compared files in the command line for the external diff application. The parameter **\${ancestorFile}** specifies the common ancestor (that is, the BASE revision of a file) in a three-way comparison. The working copy version of a file is compared with the repository version, with the BASE revision (the latest revision read from the repository by an Update or Synchronize operation) being the common ancestor of these two compared versions.



Important:

If the path to the external compare application includes spaces (or any of the subsequent options or arguments), then each of these paths or *tokens* must be double-quoted for the Oxygen JSON Editor to correctly parse and identify them. For example,

`C:\Program Files\compareDir\app name.exe` must be written as `"C:\Program Files\compareDir\app name.exe"`.

Messages Preferences

The **Messages** preferences page allows you to disable certain warning messages that may appear in the application. To configure these options, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **SVN > Messages**.

This preferences page allows you to disable the following warning messages:

Show confirmation dialog when using the "Update All" action

Allows you to avoid performing accidental update operations by requesting you to confirm them before execution.

Show confirmation dialog for drag and drop actions in Working Copy

This option avoids doing a drag and drop when you just want to select multiple files in the **Working Copy** view.

Show warning dialog when editing conflicts

When the **Edit Conflicts** action is executed, a warning dialog box notifies you that the action overwrites the conflicted version of the file created by an update operation. The conflicted file is overwritten with the version of the same file that existed in the working copy before the update operation and then proceeds with the visual editing of the conflicting file.

Show warning dialog when "svn:externals" definitions are ignored

A warning dialog box is displayed when "svn:externals" definitions are ignored before performing any operation that updates resources of the working copy (such as *Update* and *Override and Update*).

Diff Preferences

The Diff Preferences Page has sub-pages for configuring File Comparisons and Directory Comparisons.

Files Comparison Preferences

To configure the **Files Comparison** options, [open the Preferences dialog box \(Options > Preferences\)](#) (on [page 74](#)) and go to **Diff > Files Comparison**.

This preferences page allows you to configure the following options:

Enable file comparison in Author mode

If selected, a visual **Author** mode is available in the file comparison tool. It displays the files in a visual mode similar to the **Author** editing mode in *Oxygen XML Editor/Author*. This visual mode is available when both compared files are detected as being XML and from a recognized document type.

Ignore Whitespaces (Not applicable for the visual Author comparison mode)

If selected, before performing the comparison, the application normalizes the content (collapses any sequence of whitespace characters into a single space) and trims its leading and trailing whitespaces.



Note:

If the **Ignore Whitespaces** checkbox is selected, comparing the `a b` sequence with `a b`, Oxygen JSON Editor finds no differences, because after normalization, all whitespaces from the first sequence are collapsed into a single space character. However, when comparing `a b` with `ab` (no whitespace between `a` and `b`), Oxygen JSON Editor signals a difference.

Two-Way Diff section

Default algorithm

The default algorithm used for comparing two files. The following options are available:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Characters** - Computes the differences at character level, meaning that it compares two files or fragments looking for identical characters. This algorithm is not available when the file comparison is in **Author** comparison mode.

- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **Syntax Aware** - Computes differences for the file types or fragments known by Oxygen JSON Editor, taking the syntax (the specific types of tokens) into consideration. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **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. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **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 (Not applicable for the visual Author comparison mode)

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. You can choose to ignore *Processing Instructions*, *Comments*, *CDATA*, *DOCTYPE*, *Text*, *Namespaces*, *Prefixes*, *Namespace declarations*, and *Attribute order*.

Ignore nodes by XPath (Not applicable for the visual Author comparison mode)

If selected, you can enter an [XPath expression \(on page 474\)](#) to ignore certain nodes from the comparison. It will be processed as XPath version 2.0. The XPath expression specified in this option is used as the default ignore instructions **only** when the application is started. If you enter an XPath expression in the similar option on the **Diff Files** toolbar, that expression will be used instead.

Merge adjacent differences (Not applicable for the visual Author comparison mode)

If selected, the application considers two adjacent differences as one when the differences are painted in the side-by-side editors. If not selected, every difference is represented separately.

Mark end tags as different for modified elements (Not applicable for the visual Author comparison mode)

If selected, end tags of modified elements are also presented as differences. Otherwise, only the start tags are presented as differences.

Ignore expansion state for empty elements (Not applicable for the visual Author comparison mode)

If selected, empty elements in both expansion states are considered matched (that is `<element/>` and `<element></element>` are considered equal).

Appearance Preferences

To configure the appearance options for the Files Comparison tool, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Diff > Files Comparison > Appearance**. This preferences page offers the following options:

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.

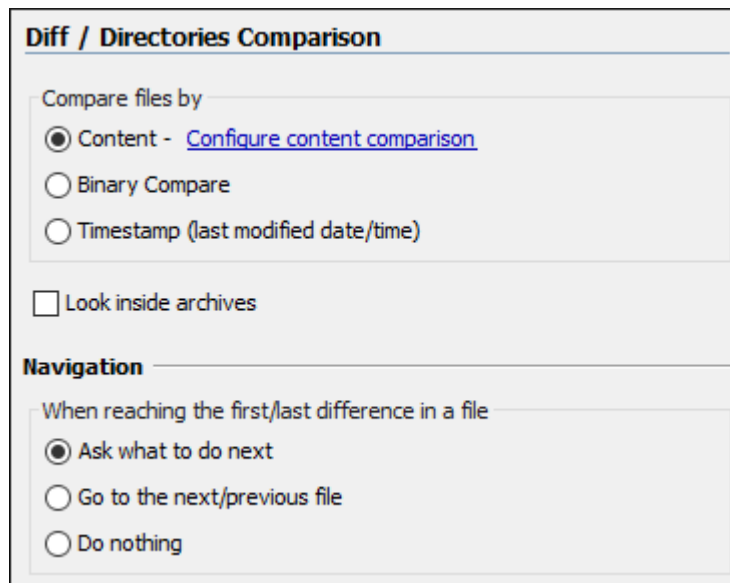
Conflict color

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**) (*on page 74*) and go to **Diff > Directories Comparison**.

Figure 13. Diff Preferences Page



You can specify the following options for the directories comparison tool:

Compare files by

Controls the method used for comparing two files:

- **Content** - The file content is compared using the current [diff algorithm](#) (*on page 166*). 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** (*on page 166*) where you can configure options for comparing files. However, the **Ignore nodes by XPath** option is ignored when using the **Compare Directories** tool.

- **Binary Compare** - The files are compared at byte level.
- **Timestamp (last modified date / time)** - The files are compared only by their last modified timestamp.

Look in archives

If selected, [known archive types \(on page 171\)](#) are considered directories and their content is compared just like regular files.

Navigation

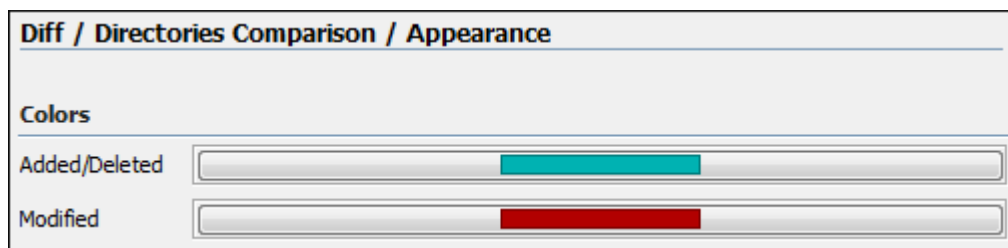
This options control the behavior of the differences traversal actions (**Go to previous modification**, **Go to next modification**) when the first or last difference in a file is reached:

- **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 \(Options > Preferences\) \(on page 74\)](#) and go to **Diff > Directories Comparison > Appearance**.

Figure 14. Diff Appearance Preferences Panel



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

Archive Preferences

To configure *Archive* options, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) 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.

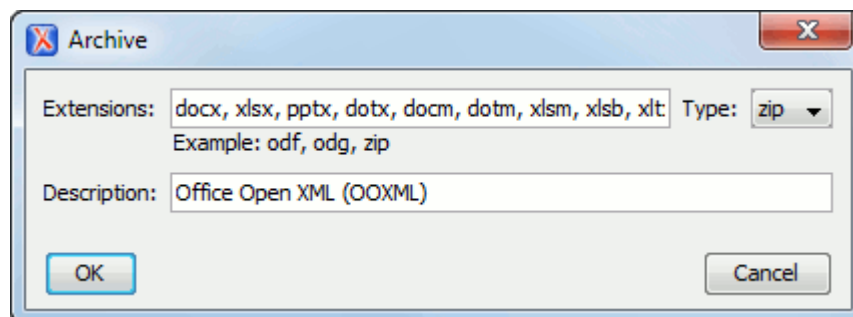
**Note:**

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 JSON Editor. You can use the **Edit** button at the bottom of the table to edit an existing mapping or the **New** button to create a new one and associate your own list of extensions to an archive format.

Figure 15. Edit Archive Extension Mappings

**Important:**

You have to restart Oxygen JSON Editor after removing an extension from the table for that extension to not be recognized as an archive extension.

Store Unicode file names in Zip archives

Use this option when you archive files that contain international (non-English) characters in file names or file comments. If this option is selected and an archive is modified in any way, UTF-8 characters are used in the names of all files in the archive.

Plugins Preferences

You can add *plugins* (on page 640) that extend the functionality of Oxygen JSON Editor. The *plugins* are shipped as separate packages. To check for new *plugins*, go to http://www.oxygenxml.com/oxygen_sdk.html.

A *plugin* consists of a separate sub-folder in the **Plugins** folder of the Oxygen JSON Editor installation folder (for example, `[OXYGEN_INSTALL_DIR]/plugins/myPlugin`). This sub-folder must contain a valid `plugin.xml` file in accordance with the `plugin.dtd` file located in the **Plugins** folder.

Oxygen JSON Editor automatically detects and loads *plugins* installed correctly in the **Plugins** folder and displays them in the **Plugins** preferences page. To configure *plugins*, open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Plugins**.

You can use the checkboxes in front of each *plugin* to enable or disable them. To display the properties of a *plugin* in the second section of the **Plugins** preferences page, click the name of the *plugin*.

External Tools Preferences

A command-line tool can be started in the Oxygen JSON Editor user interface as if from the command line of the operating system shell. The **External Tools** preferences page allows you to add and configure these external tools that could be used while working with Oxygen JSON Editor. To access this preferences page, open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **External Tools** (or select **Configure** from the **Tools > External Tools** menu).

This preferences page presents a list of the external tools that have been configured. You can use the buttons at the bottom of the page to configure the items in the list. Once a tool has been configured, you can open it by selecting it from the **Tools > External Tools** menu or from the  **External Tools** drop-down menu on the toolbar (the **Tools** toolbar needs to be selected in the **Configure Toolbars** dialog box (on page 222)).

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 16. 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** (on page 197) button, or the  **Browse** button. You can use one of the following editor variables: `${cfd}` (on page 202), `${pd}` (on page 204), `${oxygenInstallDir}` (on page 204), `${homeDir}` (on page 204), `${system(var.name)}` (on page 204), `${date(pattern)}` (on page 203), `${xpath_eval(expression)}` (on page 205).

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** (on page 197) button, or the  **Browse** button. You can use one of the following editor variables: `${homeDir}` (on page 204), `${home}` (on page 204), `${cfn}` (on page 203), `${cfne}` (on page 203), `${cf}` (on page 202), `${currentFileURL}` (on page 203), `${cfd}` (on page 202), `${cfdu}` (on page 203), `${tsf}` (on page 205), `${pd}` (on page 204), `${pdu}` (on page 204), `${oxygenInstallDir}` (on page 204), `${oxygenHome}` (on page 204), `${frameworksDir}` (on page 203), `${frameworks}` (on page 203), `${ps}` (on

page 204), `${timeStamp}` (on page 205), `${uuid}` (on page 205), `${id}` (on page 204), `${afn}` (on page 198), `${afne}` (on page 198), `${af}` (on page 198), `${afu}` (on page 198), `${afd}` (on page 198), `${afdu}` (on page 198), `${ask('message', type, 'default_value')}` (on page 198), `${dbgXML}` (on page 203), `${dbgXSL}` (on page 203), `${env(VAR_NAME)}` (on page 203), `${system(var.name)}` (on page 204), `${date(pattern)}` (on page 203), and `${xpath_eval(expression)}` (on page 205).

Show output messages

When this option is selected, all the messages emitted by the external tool are displayed in the **Results** view. When this option is not selected, only the error messages are displayed. You can also choose the output encoding and content type:

- **Output encoding** - The encoding of the output stream of the external tool that will be used by Oxygen JSON Editor to read the output of the tool.
- **Output content type** - A list of predefined content type formats that instruct Oxygen JSON Editor how to display the generated output. For example, setting the **Output content type** to `text/xml` enables the syntax coloring of XML output.

Error Encoding

The encoding of the error stream of the external tool that will be used by Oxygen JSON Editor to read the error stream.

Shortcut key

You can choose a keyboard shortcut that can be used to launch the external tool.

Menu Shortcut Keys Preferences

You can use the **Menu Shortcut Keys** preferences page to configure shortcut keys for the actions available in Oxygen JSON Editor. The shortcuts assigned to actions are displayed in a table in this preference page. To access the full list of shortcut keys, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Menu Shortcut Keys** (or simply go to **Options > Menu Shortcut Keys**).

For a list of the most commonly used shortcuts, see [Frequently Used Shortcut Keys \(on page 20\)](#).

Figure 17. Menu Shortcut Keys Preferences Page

Menu Shortcut Keys		
Name ▲	Category	Shortcut key
Remove all	Bookmarks	Ctrl+F7
Remove all	Breakpoints	Ctrl+Shift+F7
Remove all	Highlights	
Remove all	Results	
Remove comment(s)	Edit	
Remove from Disk	Project	Shift+Delete
Remove from Project	Project	Delete
Remove from version control	Working copy	
Remove highlight(s)	Edit	
Remove selected	Results	Delete
Rename	Archive Browser	F2
Rename	File	F2
Rename	Markup	
Rename	Project	F2
Rename	SharePoint	F2
Repositories	Perspective	

The **Menu Shortcut Keys** preferences page also contains the shortcuts that you define at *document type* level (on page 98).

**Note:**

A shortcut defined at *document type* level overwrites a default shortcut.

Furthermore, the shortcuts table also contains entries for actions that show side-views contributed by plug-ins.

To find a specific action, you can use the filter text field to search through any of the columns in the table. You can also press shortcut key combinations on your keyboard to filter the list and click on a column header to sort that column.

The table includes the following columns or options:

- **Description** - A short description of the action.
- **Category** - A classification of the actions in categories for easier management and more flexibility in assigning multiple keys for the same action.

- **Shortcut key** - The combination of keyboard keys that can be used to launch the action. To add or change a shortcut key, you can either double-click a row or select the row and click the **Edit** button.
- **'Home' and 'End' keys are applied at line level** (available on macOS only) - Controls the way the HOME and END keys are interpreted. If selected, the default behavior of these keys is overridden and the cursor only moves on the current line.

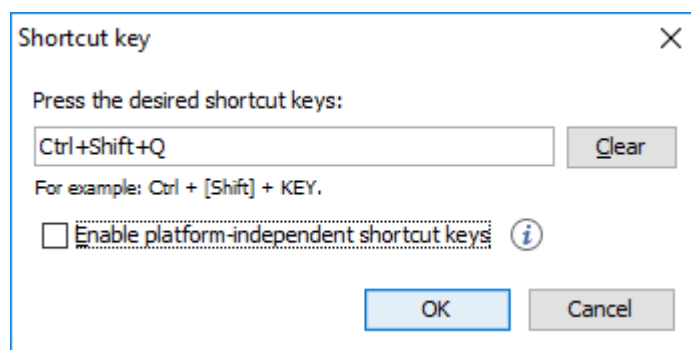
How to Assign a Shortcut Key or Edit an Existing Shortcut

To assign a shortcut key to an action or edit an existing shortcut configuration, follow these steps:

1. Select the action in the table.
2. Click the **Edit** button.

Step Result: The **Shortcut key** configuration dialog box is displayed.

Figure 18. Shortcut Key Configuration Dialog Box



3. Press the desired shortcut keys on your keyboard.
4. If you need the shortcut to work on multiple platforms, select the **Enable platform-independent shortcut keys** option. In this case, the following modifiers are used:
 - **M1** represents the **Command** key on macOS, and the **Ctrl** key on other platforms.
 - **M2** represents the **Shift** key.
 - **M3** represents the **Option** key on macOS, and the **Alt** key on other platforms.
 - **M4** represents the **Ctrl** key on macOS, and is undefined on other platforms.
5. Click **OK** to save your configuration.



Troubleshooting:

If you encounter problems with keyboard shortcuts not working as expected, see [Keyboard Shortcuts Result in Unexpected Behavior \(on page 632\)](#) or [Keyboard Shortcuts Do Not Work At All \(on page 632\)](#).

Related information

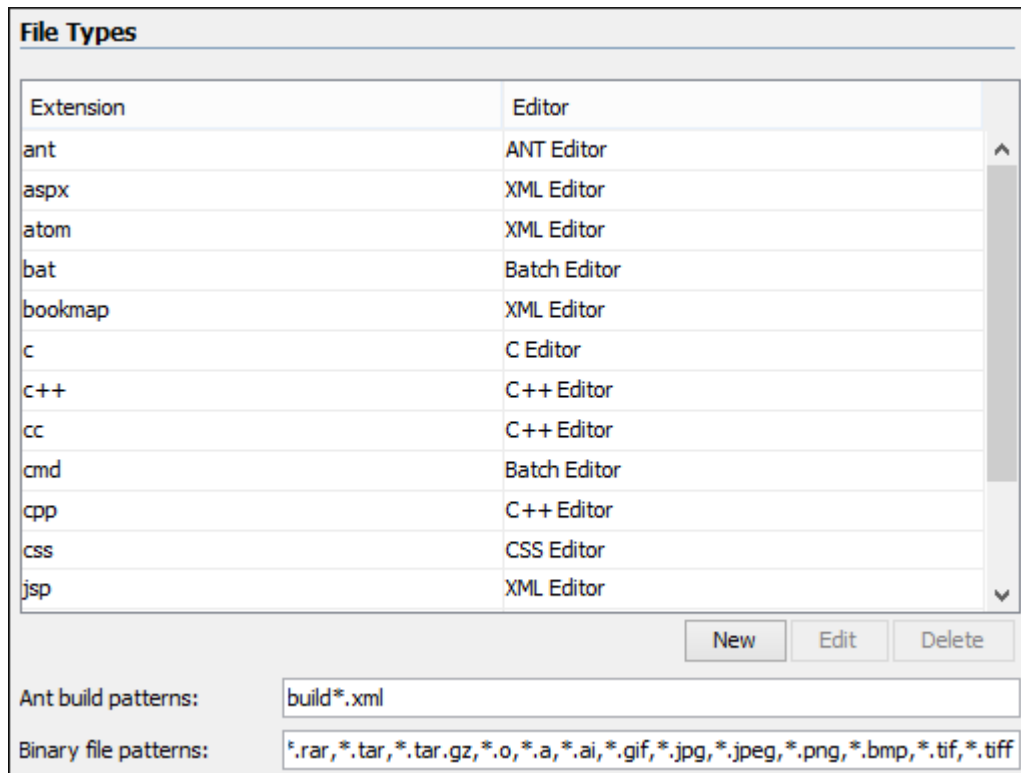
[Frequently Used Shortcut Keys \(on page 20\)](#)

File Types Preferences

Oxygen JSON Editor offers built-in editing support for a wide variety of file types, but you can also add new file extensions and associate them with whatever editor type fits your needs. The associations set here between a file extension and the type of editor will determine which editor will be opened for editing purposes when that type of file is created or opened.

To configure the **File Types** options, open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **File Types**.

Figure 19. File Types Preferences Page



The table contains the following columns:

- **Extension** - The extensions of the files that will be associated with an editor type.
- **Editor** - The type of editor which the extensions will be associated with. Some editors provide easy access to frequent operations via toolbars (XML editor, XSL editor, DTD editor) while others provide just a syntax highlight scheme (Java editor, SQL editor, Shell editor).

If the editor set here is not one of the XML editors (XML editor, XSL editor, XSD editor, RNG editor, WSDL editor) then the encoding set in the **Encoding for non-XML files** option (on page 114) 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** (on page 251): **File/Replace in Files**, **Check Spelling in Files**, **Validate**.

**Note:**

If you associate an empty extension to a content type, all files without extensions will be opened with that specific content type.

Open/Find Resource Preferences Page

You can configure various options that pertain to the **Open/Find Resource** dialog box (on page 267) and **Open/Find Resource** view (on page 264). To access these options, open the **Preferences** dialog box (**Options > Preferences**) (on page 74) 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 (on page 267).

Enable searching in content

This option is selected by default and it allows you to use the **Open/Find Resource** dialog box/view (on page 267) to search in content or reviews, as well as in file paths. If this option is not selected, you can only use the **Open/Find Resource** feature to search in file paths.

Content search scope section

Ignore content of these files

Allows you to select specific directories, files, or file types that are ignored when you perform a search. For example, `*.txt` ignores all the `.txt` files, `*/topics/` ignores all the files from the `topics` directory, regardless of their depth, and `file:/C:/tmp/*` ignores everything from the `tmp` directory.

Include the contents of these binary files

Results from a search in the **Open/Find Resource** dialog box/view (on page 267) will be returned as a PDF if the PDF contains the searched text. If you want to disable this feature, delete the contents of the text field.

Index the content of remote resources

Controls the indexing of resources that are not local. For example, the resources referenced in a *DITA map* (on page 638) opened from a remote server (from a CMS or from a WebDAV location) are not indexed by default. To index the content of these resources, select this option.

**Note:**

Selecting this option may lead to delays when the indexing is computed.

Content search options section**Content language**

Use this option to specify a language for the search engine to use for the current document. This is helpful if you have multiple languages within the content of a document. The search engine will use a set of *stop words* and analyzers tuned specifically for that specific language. By default, it is mapped to the [UI language specified in the Global preferences page \(on page 76\)](#). Therefore, you need to change this option only if the language of the text you want to perform the search in differs from the UI language.

**Tip:**

If you select **<Generic language (no stemming)>** from the drop-down list, no word stemming is performed when creating the index. This might be useful if your content has many technical terms that should be indexed as they are.

Stop words

A list of *stop words* that will be filtered out of the search processing. The list is automatically populated based upon the specified **Content language**, but you can add or remove words from the list.

When searching in content, return

This option specifies how matches are returned when doing searches in content. You can choose between two options:

- **Exact matches** - The search results match the exact whole words that you enter in the search field of the **Open/Find Resource** dialog box/view.
- **Prefix matches** (default) - The search results match documents that contain words starting with the search terms. For instance, searching for "pref page" will also find documents containing "preference page".

Automatically join search terms using:

Allows you to select the default boolean operator that Oxygen JSON Editor applies when you perform a search. For example, if the AND operator is selected and you search for "car assembly", the matches must contain both of the words. If you choose OR, the matches must contain one of the selected search terms and results that contain both words are promoted to the top of the list.

Stop Words

A list of comma-separated *stop words*, meaning that the words added in this list are filtered out prior to processing a search query.

Related information

[Open/Find Resource View \(on page 264\)](#)

[Open/Find Resource Dialog Box \(on page 267\)](#)

Custom Editor Variables Preferences

An [editor variable \(on page 197\)](#) 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 \(on page 197\)](#).

Custom editor variables are created and configured in the **Custom Editor Variables** preferences page. To access this page, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) 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.

Figure 20. Custom Editor Variables Table

Custom Editor Variables		
Name	Value	Description
<code>\${startDir}</code>	<code>../../bin</code>	Start directory of command line validator
<code>\${standardParams}</code>	<code>-c config.xml -v -level 5 -list</code>	List of command line standard parameters
New Edit Delete		

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**) (on page 74) and go to **Network Connection Settings > Proxy**. The following options are available:

Proxy section

Specifies how HTTP(S) connections go through the proxy server. You can choose between the following three options:

- **Direct connection** - HTTP(S) connections will go directly to the target host without going through a proxy server.
- **Use system settings** (default setting) - HTTP(S) connections will go through the proxy server set in the operating system.



Attention:

The system settings for the proxy cannot be read correctly from the operating system on some Linux systems. The system settings option should work properly on Gnome-based Linux systems, but it does not work on KDE-based ones as the Java virtual machine does not offer the necessary support yet.

- **Manual proxy configuration** - HTTP(S) connections will go through the proxy server specified in the **Web Proxy (HTTP/HTTPS)** section.

Web Proxy (HTTP/HTTPS) section

Address

The address of the proxy server used for manual configurations.

Port

The port of the proxy server used for manual configurations.

No proxy for

Specifies the hosts that the connections must not go through a proxy server.

A host needs to be written as a fully qualified domain name (for example, `myhost.example.com`) or as a domain name (for example, `example.com`). Use a comma to separate multiple hosts.

User

The user name for authentication with the proxy server.

Password

The password for authentication with the proxy server.

SOCKS Proxy section

Address

The address of a SOCKS proxy that all connections will pass through. If this field is empty, the connections do not use a SOCKS proxy.

Port

The port of a SOCKS proxy that all connections will pass through.

HTTP(S) Preferences

To set the **HTTP(S)/WebDAV** preferences, open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Network Connection Settings > HTTP(S)/WebDAV**. The following options are available:

Maximum number of simultaneous connections per host

Defines the maximum number of simultaneous connections established by the application with a distinct host. Servers might consider multiple connections opened from the same source to be a **Denial of Service** attack. You can avoid that by lowering the value of this option.



Note:

The minimum value that can be set in this option is 5.

Read Timeout (seconds)

The period (in seconds) after which the application considers that an HTTP server is unreachable if it does not receive any response from that server.

Enable HTTP 'Expect: 100-continue' handshake (for HTTP/1.1 protocol)

Activates *Expect: 100-Continue* handshake. The purpose of the *Expect: 100-Continue* handshake is to allow a client that is sending a request message with a request body to determine if the origin server is willing to accept the request (based on the request headers) before the client sends the request body. The use of the *Expect: 100-continue* handshake can result in noticeable performance improvement when working with databases. The *Expect: 100-continue* handshake should be used with caution, as it may cause problems with HTTP servers and proxies that do not support the HTTP/1.1 protocol.

Use the 'Content-Type' header field to determine the content type

When selected, Oxygen JSON Editor tries to determine a resource type using the **Content-Type** header field. This header indicates the *Internet media type* of the message content, consisting of a type and subtype. For example:

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

When unchecked, the resource type is determined by analyzing its extension. For example, a file ending in `.xml` is considered to be an XML file.

Automatic retry on recoverable error

When selected, if an HTTP error occurs when Oxygen JSON Editor communicates with a server via HTTP (for example, sending or receiving a SOAP request to or from a Web services server) and the error is recoverable, Oxygen JSON Editor tries to re-send the request to the server.

Cache content for similar HTTP requests when opening and validating documents

When opening XML documents that contain lots of `xi:include` elements over HTTP (for example), depending on how content is reused, there may be lots of HTTP requests to the same target files during the validation or opening of the XML document. When this setting is selected, HTTP calls to the same target file are cached and the opening and validation of such XML documents may take less time.

Automatically accept a security certificate, even if invalid

When selected, the HTTPS connections that Oxygen JSON Editor attempts to establish with will accept all security certificates, even if they are invalid.

**Important:**

By accepting an invalid certificate, you accept (at your own risk) a potential security threat, since you cannot verify the integrity of the certificate's issuer.

Lock WebDAV files on open

If selected, the files opened through WebDAV are locked on the server so that they cannot be edited by other users while the lock placed by the current user still exists on the server.

(S)FTP Preferences

To configure the (S)FTP options, open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Network Connection Settings > (S)FTP**. You can customize the following options:

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 JSON Editor will use it for communication. Otherwise, it will use ISO-8859-1. This section also includes a **Show hidden files** toggle option.

Public known hosts file

Specifies the file that contains 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 that contains the private key used for the private key method of authentication of the secure FTP (SFTP) protocol. Only RSA private keys in PEM (Base64) and PPK (PuTTY) formats are supported. Other keys (such as OpenSSH) are not supported.

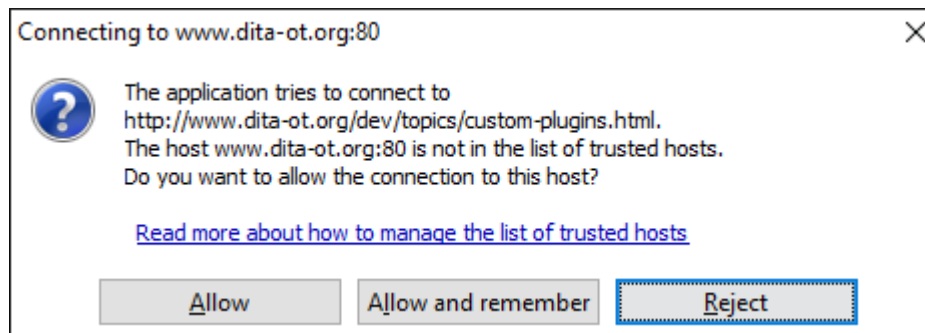
Passphrase

The passphrase used for the private key method of authentication of the secure FTP (SFTP) protocol.

Trusted Hosts Preferences

Oxygen JSON Editor comes with a built-in firewall that controls the access to external resources. Anytime the application detects a request to connect to a remote resource, it checks to see if the URL belongs to a domain that has been identified as trusted. If not, a confirmation dialog box will be displayed where you can choose whether to allow or reject access to the remote connection.

Figure 21. Trusted Hosts Confirmation Dialog Box



You can configure the list of trusted hosts using the **Trusted Hosts** preferences page. It contains a list of domains that have been identified as trusted. You can add or remove domains from the list and Oxygen JSON Editor will allow connections to the listed hosts without requesting user confirmation.

To add or remove domains, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) 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 \(Options > Preferences\) \(on page 74\)](#) 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
```

Views Preferences

The **Views** preferences page allows you to configure some options regarding certain views. To edit these options, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) 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** [\(on page 251\)](#). It should be disabled only if there is a possibility of accidentally changing the structure of the project by drag and drop actions.

Information view section

Maximum number of lines

Specifies the maximum number of lines that can be written in the **Information view** [\(on page 348\)](#).

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 \(Options > Preferences\) \(on page 74\)](#) and go to **Messages**.

The following warning messages can be enabled or disabled:

Show Java vendor warning at startup

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

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** [\(on page 251\)](#), and **Archive Browser** [\(on page 480\)](#). 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 open 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** (*on page 474*) 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

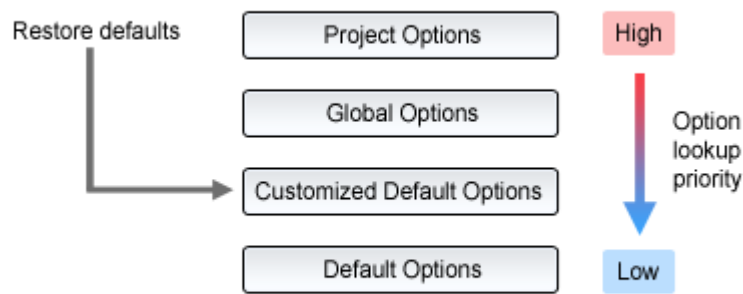
Specifies whether or not to display an error message if you try to connect to a Microsoft SharePoint server without having the proper license.

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. 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 JSON Editor, allowing you to configure most of the features. To offer you the highest degree of flexibility in customizing the application to fit the needs of your organization, Oxygen JSON Editor includes several distinct layers of option values.

Figure 22. Option Lookup Priority

The option layers are as follows (sorted from high priority to low):

- **Project Options** ([on page 188](#))

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** ([on page 188](#))

Allows individual users to personalize Oxygen JSON Editor according to their specific needs.

- **Default Options**

The predefined default values, tuned so that Oxygen JSON Editor behaves optimally in most working environments.



Important:

If you set a specific option in one of the layers, but it is not applied in the application, make sure that one of the higher priority layers does not overwrite it.

Storing Global and Project Level Options

When you configure the Oxygen JSON Editor options, you can store them globally or bind them to a specific project by choosing the appropriate setting in the preferences pages. They can then be [shared with others by exporting the global options \(on page 189\)](#) or by [sharing the stored project-level files \(on page 189\)](#). The same is true with transformation and validation scenarios.

For each preferences page, you can choose between **Global Options** ([on page 188](#)) and **Project Options** ([on page 188](#)) depending upon how you want to store the options in that particular preferences page.

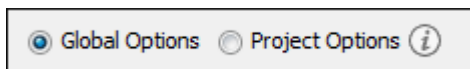
**Notice:**

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

If changes have been made to the options in a preferences page and you switch between **Project Options** and **Global Options**, a dialog box will be displayed that allows you to select one of the following:

- **Overwrite** - The existing options from the current preferences page will be overwritten.
- **Preserve** - The existing options from the current preferences page will be preserved.

Figure 23. Controlling the Storage Options for the Preferences



Global Options

By default, **Global Options** is selected in the preferences pages, meaning that the options are stored locally on your computer and are not accessible to other users (unless you [export them into an XML options file that can then be shared \(on page 188\)](#)).

Global options are stored locally in option files (for example, `oxyOptionsSa19.1.xml` for a standalone distribution of Oxygen JSON Editor version 19.1) located in the following directories:

- **Windows (7, 8, 10)** - `[user_home_directory]\AppData\Roaming\com.oxygenxml.jsoneditor`
- **macOS** - `[user_home_directory]/Library/Preferences/com.oxygenxml.jsoneditor`
- **Linux/Unix** - `[user_home_directory]/.com.oxygenxml.jsoneditor`

Project Options

If you select **Project Options**, the preferences are stored in the project file (`.xpr`), which can easily be [shared with other users \(on page 188\)](#).

**Notice:**

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

Sharing Application Settings

There are a variety of ways that you can share the settings in Oxygen JSON Editor with other members of your team so that you all use a common set of options. This topic describes various possibilities.

Share Settings Through a Project File

Most of the preference pages in Oxygen JSON Editor include a **Project Options** (on page 640) button that allows you to pass changes to the settings to the current project file that is opened in the **Project view** (on page 251). That project file can then be shared with other users. For instance, if your project file is saved on a version control system (such as SVN, CVS, or Source Safe) or in a shared folder, your team will have access to the same option configuration that you stored in the project file.

For more information about sharing projects, see [Sharing a Project - Team Collaboration](#) (on page 261).

Share Settings by Exporting/Importing Global Options

Oxygen JSON Editor includes actions in the **Options** menu that allow you to export and import the **global settings** (on page 639). The **Export Global Options** action will save the global settings as an XML properties file. You can then share those settings with others by using the **Import Global Options** action to import that properties file on their computer.

For more information about global options, see [Importing/Exporting/Resetting Global Options](#) (on page 189).

Share Settings with a Custom Options File During Installation

When Oxygen JSON Editor is installed, all the settings are set to default values. You can customize the set of default values by creating an XML *options file* that you will use when installing Oxygen JSON Editor on each computer. You can then copy the XML options file to the installation directory or specify its path in a startup parameter.

Share Settings by Imposing Fixed Options with an API

The Maven-based [Oxygen XML SDK](#) includes a sample *plugin* called **ImposeOptions** that imposes a fixed set of options when the application starts. This can be achieved by using the `PluginWorkspaceProvider.getPluginWorkspace().setGlobalObjectProperty(key, value)` API method.

For more information about this API, see [PluginWorkspaceProvider Class](#).

Importing/Exporting/Resetting Global Options

Actions for importing, exporting, and resetting global options are available in the **Options** menu. The export operation allows you to save **global preferences** (on page 639) as an XML properties file and the import operation allows you to load the property file. You can use this file to reload saved options on your computer or to [share with others](#) (on page 187).

The following actions are available in the **Options** menu:

Reset Global Options

Restores the preferences to the factory defaults.

Import Global Options

Allows you to import a set of *Global Options* from an exported XML properties file. You can also select a [project-level options file \(on page 261\)](#) (.xpr) to import all the *Global Options* that are set in that project file. After you select a file, the **Import Global Options** dialog box is displayed, and it informs you that the operation will only override the options that are included in the imported file. You can select the **Reset all other options to their default values** option to reset all options to the default values before the file is imported.

Export Global Options

Allows you to export *Global Options* to an XML properties file. Some user-specific options that are private are not included. For example, passwords and the name of the *Review Author* is not included in the export operation.

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

- [user-home-folder]\AppData\Roaming\com.oxygenxml.jsoneditor for Windows
- [user-home-folder]/Library/Preferences/com.oxygenxml.jsoneditor for macOS
- [user-home-folder]/.com.oxygenxml.jsoneditor for Linux

The name of the **options** file of Oxygen JSON Editor 26.0 is oxyJSONEditorOptionsSa26.0.xml.

Configuring the Layout of the Views and Editors

All of the side-views available in Oxygen JSON Editor are [dockable \(on page 638\)](#) and there are various ways to configure and arrange the layout of the views and editing panes. You can also [configure the layout of the toolbars \(on page 222\)](#).

To open a view, select it from the **Window > Show View** menu. You can hide a view by closing it with the **×** button at the top-right corner of the view, or with the **Window > Hide current view** action.

Arranging the Layout

You can drag any view to any margin of another view or editor inside the Oxygen JSON Editor window. Once you create a layout that suits your needs, you can save it from **Window > Export Layout**. Oxygen JSON Editor creates a layout file containing the preferences of the saved layout. To load a layout, go to **Window > Load Layout**. To reset it, select **Window > Reset Layout**.



Note:

The **Load Layout** menu lets you select between the default layout, a predefined layout, or a custom layout. The changes you make using the **Load Layout** menu are also reflected in the **Application Layout** preferences page.

The changes you make to any layout are preserved between working sessions. The predefined layout files are saved in the [preferences directory \(on page 75\)](#) of Oxygen JSON Editor.

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 the following example, the `personal.xml` file was dragged over the `personal-schema.xml` file:

Hide or Float Views

Hide

To gain more editing space in the Oxygen JSON Editor window, click  **Toggle auto-hide** in any view. This button sets the view in the *auto-hide* state, making it visible only as a vertical tab, at the margins of the Oxygen JSON Editor window. To display a view in the *auto-hide* state, hover its side-tab with your cursor, or click it to keep the view visible until you click elsewhere.

Float

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 JSON Editor window.

Maximize the Editing Environment

You can configure the interface to maximize the editing area, leaving more vertical screen space available for the main editing pane. This is, for example, useful for presentations on low-resolution screens or for laptops with small screen space. You can use the following two actions that are available in the **Window** menu to create a near full-screen editing environment:

Maximize Editor Area

If toggled on, all side views are minimized to give you more horizontal space in the editing pane.

Hide All Toolbars

If toggled on, all toolbar buttons are hidden to give you more vertical space in the interface.

Tile/Stack Editor Actions

You can also tile or stack all open editors using the following actions from the toolbar or **Window** menu:

Tile Editors Horizontally

Splits the editing area into horizontal tiles, one for each open file.

Tile Editors Vertically

Splits the editing area into vertical tiles, one for each open file.

Stack Editors

The reverse of the **Tile Editors Horizontally/Vertically** actions. Stacks all open editors.

Synchronous Scrolling

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



Note:

When tiled, you can still drag and drop the editors, but note that they are docked in the same way as a window/view (instead of just tabs). You are actually rearranging the editor windows, so drag the editor tab and drop it to one of the sides of an editor (left/right/top/bottom). While dragging, you will see the dark gray rectangle aligned to one of the sides of the editor, or around the entire editor window. If you drop it to one of the sides, it will dock to that side of the editor. If you drop it when the rectangle is around the entire window of the editor, it will get stacked on top of that editor. You can also grab one of the stacked editors and tile it to one of the sides.

Split Editor Actions

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

-  **Split Editor Horizontally** - Splits the editor horizontally so that two editor panes are displayed with one on top of the other. This is useful for comparing and merging content between two documents.
-  **Split Editor Vertically** - Splits the editor vertically so that two editor panes are displayed side by side. This is useful for comparing and merging content between two documents.
-  **Unsplit Editor** - Removes a split action on the editing area.

To maximize or restore the editors, go to **Window > Maximize Editing Area**.

Switch, Move, or Hide Editor Tabs

Each file that has been opened has a tab at the top of the editing pane and there are several ways to switch between tabs or move them, and you can even hide the tabs to only show the currently open file.



Note:

If multiple file tabs are left open when you close the application, upon startup, Oxygen JSON Editor will not load the file content until you switch to the corresponding file tab. The tabs remain visible as a placeholders until the focus is switched to them. This helps to improve the application's startup time. If you want to disable this feature (meaning that the previously open files will all be re-loaded at startup), deselect the **Load file content only when switching to its corresponding editor tab** option in the **Global** preferences page (on page 77).

Switching Editor Tabs

You can switch between editor tabs by using any of the following methods:

Mouse and Scroll Wheel

Of course, you can switch to a different editor tab by left-clicking the tab with your mouse, but when there are too many open tabs to fit on the screen, you can hover over the tab stripe and use the scroll wheel on your mouse to scroll to the left or right (same as using the two arrows on the far-right of the tab stripe).

Buttons on the Far-Right of the Tab Stripe (◀▶☰)

You can use the arrow buttons (◀▶) on the right side of the tab stripe to scroll to the left or right and the ☰ **Show List** button opens a pop-up window that displays all the open file tabs and allows you to select and switch to a specific open file.

Ctrl + Tab (Command + Tab on macOS) [NOTE: Ctrl + Page Down (Ctrl + Option + Right Arrow on macOS) does the same]

Switches to the next open tab in the order specified in the [Order of switching between editor tabs option \(on page 78\)](#).

Ctrl + Shift + Tab (Command + Shift + Tab on macOS) [NOTE: Ctrl + Page Up (Ctrl + Option + Left Arrow on macOS) does the same]

Switches to the previous open tab in the order specified in the [Order of switching between editor tabs option \(on page 78\)](#).

Window > Switch editor tab (Ctrl + F9 (Command + F9 on macOS))

This action opens a dialog box that allows you to switch to a particular editor tab by selecting it from a filterable list. This is especially helpful when you have a large amount of open file tabs and you want to switch to a certain tab this is not shown on the screen. It includes a search filter field and several options to help you find specific open file tabs.

The **Switch Editor Tab** dialog box contains the following options and features:

Search Filter

You can enter text in the filter field at the top of the dialog box to filter the list and search for specific open files. You can enter any number of terms, separated by space, and wildcards are allowed (for example, * to match any sequence of characters, or ? to match a single character). This field also has a history drop-down that allows you to select previously used search terms.

Match all terms

If this option is selected, only the files that match all of your search terms will be displayed. If you use a wildcard in the search filter, this option is automatically disabled.

Include file paths

If this option is selected, the search is expanded to include file paths, and also the paths are displayed in this dialog box.

Case sensitive

If this option is selected, the search operation will be case-sensitive.

List of Open File Tabs

All files that are currently open are displayed in the upper part of the main pane of the dialog box, followed by recently closed files. Files that have been modified but not yet saved are prefixed by an asterisk. To switch to a particular file tab, double-click the file or select it and click **OK**.

Moving Editor Tabs

You can move editor tabs by using any of the following methods:

Mouse Drag

You can use your mouse to drag editor tabs to a new location on the tab stripe.

Ctrl + Alt + Comma

Moves the current file tab one position to the left.

Ctrl + Alt + Period

Moves the current file tab one position to the right.

Hiding Editor Tabs

If you want to hide all the file tabs and only show the currently open file, select **Hide editor tabs** from the **Window** menu. This does not close the other tabs, just hides them. You can still navigate between tabs using keyboard shortcuts (**Ctrl + Tab**, **Ctrl + Shift + Tab**, **Ctrl + F6**, **Ctrl + Shift + F6**) or by selecting **Next editor** or **Previous editor** from the **Window** menu.

Resources

For more information about configuring the interface of Oxygen JSON Editor, watch our video demonstration:

<https://www.youtube.com/embed/anwjepfAdEk>



Tip:

To get more ideas for more advanced customization possibilities, watch our Webinar: [Working with DITA in Oxygen - Customizing the Editing Experience](#). It offers a visual demonstration of how to customize actions, document validation, content completion, new document templates, **Author** mode rendering, and more.

Related information

[Configuring Toolbars \(on page 222\)](#)

Configuring Toolbars

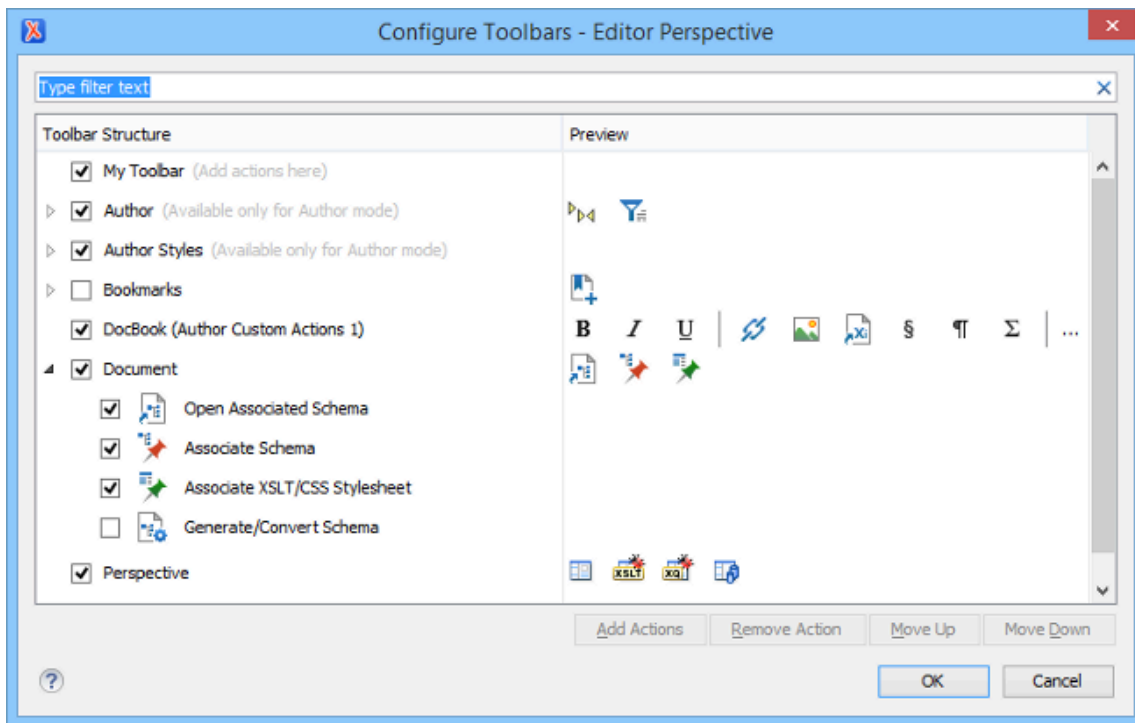
You can configure the toolbars in Oxygen JSON Editor to personalize the interface for your specific needs. You can choose which toolbars to show or hide in the current editor mode (**Text**, **Author**, **Design**, or **Grid**). You can

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 24. 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 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* (on page 639) or 3rd party *plugins* (on page 640).

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.

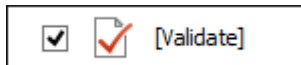
The **Configure Toolbars** dialog box also provides a variety of other ways to customize the layout in Oxygen JSON Editor.

Customize My Toolbar

You can customize the **My Toolbar** to include your most commonly used actions. By default, this toolbar is listed first. Also, it is hidden until you add actions to it and you can easily hide it with the **Hide "My Toolbar" Toolbar** action that is available when you right-click anywhere in the toolbar area.

Drop-down Menu Actions

Composite actions that are usually displayed as a drop-down menu can only be selected in one toolbar at a time. These actions are displayed in the **Configure Toolbars** dialog box with the name in brackets.



Configure External Tools Action

There is a  **Configure external tools** composite action that appears in the toolbar called **Tools**. It is a drop-down menu that contains any external tools that are configured in the **External Tools** preferences page.



Note:

If no external tools are configured, this drop-down menu is not shown in the toolbar.

Additional actions are available from the **Window** menu or contextual menu when invoked from a toolbar that allows you to further customize your layout. These actions include:

Reset Toolbars

To reset the layout of toolbars to the default setting, select the **Reset Toolbars** action from the contextual menu or **Window** menu.

Reset Layout

To reset the entire layout (including toolbars, editing modes, views, etc.) to the default setting, select **Reset Layout** from the contextual menu or **Window** menu.

Export Layout

You can use the **Export Layout** action that is available in the **Window** menu to export the entire layout of the application to share it with other users.

Hide Toolbars

You can use the **Hide Toolbar** action from the contextual menu to easily hide a displayed toolbar. When you right-click a toolbar it will be highlighted to show you which actions are included in that toolbar.

Related information

[Configuring the Layout of the Views and Editors \(on page 217\)](#)

Import/Export Validation Scenarios

You can export validation scenarios into specialized *scenarios* files. You can import validation scenarios from various sources (such as project files, [framework \(on page 639\)](#) option files, or exported scenario files). The import and export scenario actions are available in the **Options** menu. The following actions are available:

Import Validation Scenarios

Loads a set of validation scenarios from a project file, *framework* options file, or exported scenarios file.

Export Global Validation Scenarios

Stores a set of global validation scenarios in a specialized *scenarios* file.

The **Export Global Validation Scenarios** option is 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 Validation Scenarios** action. 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 JSON Editor includes a variety of built-in editor variables. You can also create your own custom editor variables by using the [Custom Editor Variables preferences page \(on page 180\)](#).

Editor variables are evaluated and automatically expanded in many places in the application, when:

- Creating new documents from file templates (*on page 229*).
- Inserting code templates (*on page 229*) in the **Text** or **Author** editor modes.
- Using specific Java API `UtilAccess.expandEditorVariables(String, URL)` from plugins and framework extensions.

You can use the following editor variables in Oxygen JSON Editor commands of external engines or other external tools, and in various places in the application:

- **\${af}** - The local file path of the ZIP archive that includes the currently edited document.
- **\${afd}** - The local directory path of the ZIP archive that includes the currently edited document.
- **\${afdu}** - The URL path of the directory of the ZIP archive that includes the currently edited document.
- **\${afn}** - The file name (without parent directory and without file extension) of the zip archive that includes the currently 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 currently edited file.
- **\${afu}** - The URL path of the ZIP archive that includes the currently edited document.
- **\${answer(@id)}** - Used in conjunction with the **\${ask}** editor variable. The **@id** parameter is required and identifies the answer from the **\${ask}** editor variable with the same ID.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topic PUBLIC "-//OASIS//DTD DITA Topic//EN" "topic.dtd">
<topic id="topic_lcf_1c4_tdb">
  <title></title>
  <body>
    <data name="${ask('Set a data name', String, 'name', @name)}"></data>
    <p>The name is: ${answer(@name)}</p>
  </body>
</topic>
```

- **\${ask('message', type, ('real_value1': 'rendered_value1'; 'real_value2': 'rendered_value2'; ...), 'default_value', @id)}** - 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.
- **'default-value'** - Optional parameter. Provides a default value.
- **@id** - Optional parameter. Used for identifying the variable to reuse the answer using the **\${answer(@id)}** editor variable.
- **type** - Optional parameter (defaults to **generic**), with one of the following values:



Note:

The title of the dialog box will be determined by the type of parameter and as follows:



- For *url* and *relative_url* parameters, the title will be the name of the parameter and the value of the 'message'.
- For the other parameters listed below, the title will be the name of that respective parameter.
- If no parameter is used, the title will be "Input".

**Notice:**

Editor variables that are used within a parameter of another editor variable must be escaped within single quotes for them to be properly expanded. For example:

```
${ask( 'Provide a date',generic, '${date(yyyy-MM-dd'T'HH:MM)}' )}
```

Parameter	
generic (default)	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 <i>Hello world!</i> message displayed. ▪ <code>\${ask('Hello world!', generic, 'Hello again!')}</code> - The dialog box has a <i>Hello world!</i> message displayed and the value displayed in the input box is <i>Hello again!</i>.
url	Format: <code>\${ask('message', url, 'default_value')}</code>
	Description: Input is considered a URL. Oxygen JSON Editor checks that the provided URL is valid.
	Example: ▪ <code>\${ask('Input URL', url)}</code> - The displayed dialog box has the name <i>Input URL</i>. The expected input type is URL. ▪ <code>\${ask('Input URL', url, 'http://www.example.com')}</code> - The displayed dialog box has the name <i>Input URL</i>. The expected input type is URL. The input field displays the default value <i>http://www.example.com</i>.
relative_url	Format: <code>\${ask('message', relative_url, 'default')}</code>
	Description: Input is considered a URL. This parameter provides a file chooser, along with a text field. Oxygen JSON Editor tries to make the URL relative to that of the document you are editing.

Parameter	
	 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 JSON Editor 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 currently edited document location.
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 abcd value.
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 <i>rendered_value</i> values. Choosing such a value will return its associated value (<i>real_value</i>). Note: The list of <i>'real_value':'rendered_value'</i> pairs can be computed using <code>\${xpath_eval()}</code>.

Note:
 The *'default'* parameter specifies the default-selected value and can match either a key or a value.

Example:

Parameter	
	▪ <code>\${ask('Operating System', combobox, ('win':'Microsoft Windows';'macos':'macOS';'lnx':'Linux/UNIX'), 'macos')}</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>macOS</code>, as in the following example: <code>\${ask('Operating System', combobox, ('win':'Microsoft Windows';'macos':'macOS';'lnx':'Linux/UNIX'), 'macOS')}</code> ▪ <code>\${ask('Mobile OS', combobox, ('ios':'iOS';'and':'Android'), 'Android')}</code> ▪ <code>\${ask('Mobile OS', combobox, (\$xpath_eval(for \$pair in (['ios', 'iOS'], ['and', 'Android']) return "" \$pair?1 "":" \$pair?2 "",")), 'ios')}</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 <i>rendered_value</i> values. Choosing such a value will return its associated real value (<i>real_value</i>) or the value inserted when you edit a list entry.  Note: The list of 'real_value':'rendered_value' pairs can be computed using <code>\$xpath_eval()</code>.  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';'macos':'macOS';'lnx':'Linux/UNIX'), 'macos')}</code> - The dialog box has the name 'Operating System'. The drop-down menu displays the three given operating systems and also allows you to ed-

Parameter	
	it the entry. The associated value will be returned based upon your selection or the text you input. ▪ <code>\${ask('Operating System', editable_combobox, (\$xpath_eval(for \$pair in ([win', 'Microsoft Windows'], [macos', 'macOS'], [lnx', 'Linux/UNIX']) return "" \$pair?1 "':" \$pair?2 "';"))}, 'ios')}</code>
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 <i>rendered_value</i> and will return an associated <i>real_value</i>.  Note: The list of <i>real_value</i>:'rendered_value' pairs can be computed using <code>\$xpath_eval()</code>.  Note: The <i>default</i> parameter specifies the default-selected value and can match either a key or a value. Example: ▪ <code>\${ask('Operating System', radio, ('win':'Microsoft Windows';'macos':'macOS';'lnx':'Linux/UNIX'), 'macos')}</code> - The dialog box has the name <i>Operating System</i>. The radio button group allows you to choose between the three operating systems.  Note: In this example, macOS is the default-selected value and if selected, it would return <i>macos</i> for the output. ▪ <code>\${ask('Operating System', radio, (\$xpath_eval(for \$pair in ([win', 'Microsoft Windows'], [macos', 'macOS'], [lnx', 'Linux/UNIX']) return "" \$pair?1 "':" \$pair?2 "';"))}, 'ios')}</code>

- **`${caret}`** - The position where the cursor is located. This variable can be used in a code template, in **Author** mode operations, or in a **selection plugin**.
- **`${cf}`** - Current file as file path, that is the absolute file path of the currently edited document.
- **`${cfd}`** - Current file folder as file path, that is the path of the currently edited document up to the name of the parent folder.

- **`\${cfdu}`** - Current file folder as URL, that is the path of the currently edited document up to the name of the parent folder, represented as a URL.
- **`\${cfn}`** - Current file name without the extension and parent folder. The current file is the one currently open and selected.
- **`\${cfne}`** - Current file name with extension. The current file is the one currently open and selected.
- **`\${comma}`** - Used to display a comma when the actual comma symbol would be considered part of some sort of instruction or delimiter.
- **`\${cp}`** - Current page number. Used to display the current page number on each printed page in the **Editor / Print** Preferences page.
- **`\${currentFileURL}`** - Current file as URL, that is the absolute file path of the currently edited document represented as URL.
- **`\${date(pattern)}`** - Current date. The allowed patterns are equivalent to the ones in the [Java SimpleDateFormat](#) class. **Example:** `yyyy-MM-dd`.

**Note:**

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

- **`\${dbgXML}`** - The local file path to the XML document that is currently selected.
- **`\${dbgXSL}`** - The local file path to the XSL/XQuery document that is currently selected.
- **`\${dita.dir.url}`** - A special local contextual editor variable that gets expanded only in the **Libraries** dialog box that is accessible from the **Advanced** tab of DITA transformation scenarios. The **Libraries** dialog box allows you to specify additional libraries (*JAR (on page 639)* files or additional class paths) to be used by the transformer. This ``${dita.dir.url}`` editor variable gets expanded to the value of the *dita.dir* parameter from the **Parameters** tab of the DITA transformation scenario.
- **`\${ds}`** - The path of the detected schema as a local file path for the current validated XML document.
- **`\${dsu}`** - The path of the detected schema as a URL for the current validated XML document.
- **`\${env(VAR\_NAME)}`** - Value of the *VAR_NAME* environment variable. The environment variables are managed by the operating system. If you are looking for Java System Properties, use the **`\${system(var.name)}`** editor variable.
- **`\${framework(fr\_name)}`** - The path (as URL) of the *fr_name* framework.
- **`\${framework}`** - The path (as URL) of the current **framework** directory.
- **`\${frameworkDir}`** - The path (as file path) of the current **framework** directory.
- **`\${frameworks}`** - The path (as URL) of the **frameworks** directory. When used to define references inside a framework configuration, it expands to the parent folder of that specific framework folder. Otherwise, it expands to the main frameworks folder defined in the **Document Type Association > Locations** preferences page.
- **`\${frameworksDir}`** - The path (as file path) of the **frameworks** directory. When used to define references inside a framework configuration, it expands to the parent folder of that specific framework folder. Otherwise, it expands to the main **frameworks** folder defined in the **Document Type Association > Locations** preferences page.

- **\${home}** - The path (as URL) of the user home folder.
- **\${homeDir}** - The path (as file path) of the user home folder.
- **\$(18n(key))** - Editor variable used only at *framework*-level to allow translating names and descriptions of **Author** mode actions in multiple actions.
- **\${id}** - Application-level unique identifier. It is a short sequence of 10-12 letters and digits that is not guaranteed to be universally unique.
- **\$(makeRelative(base,location))** - Takes two URL-like paths as parameters and tries to return a relative path. A use-case would be to insert content references to a certain reusable component when defining code templates.

Example:

```
${makeRelative(${currentFileURL}, ${dictionaryURL}#gogu)}
```

- **\$(oxygenHome)** - Oxygen JSON Editor installation folder as URL.c
- **\$(oxygenInstallDir)** - Oxygen JSON Editor installation folder as file path.
- **\$(pd)** - The file path to the folder that contains the current project file (**.xpr**).
- **\$(pdu)** - The URL path to the folder that contains the current project file (**.xpr**).
- **\$(pluginDir(pluginID))** - Each plugin has an ID specified in its **plugin.xml** file. This editor variable expands to the file path of the folder that contains the **plugin.xml** file where that specific plugin ID is located.
- **\$(pluginDirURL(pluginID))** - Each plugin has an ID specified in its **plugin.xml** file. This editor variable expands to the URL path of the folder that contains the **plugin.xml** file where that specific plugin ID is located.
- **\$(pn)** - Current project name.
- **\$(ps)** - Path separator, which is the separator that can be used on the current platform (Windows, macOS, Linux) between library files specified in the class path.
- **\$(rootMapDir)** - Will be expanded to the current root map parent directory file path.
- **\$(rootMapDirURL)** - Will be expanded to the current root map parent directory URL.
- **\$(rootMapFile)** - Will be expanded to the current root map file path.
- **\$(rootMapURL)** - Will be expanded to the current root map URL. For example, if in the main DITA Map you define a key with a certain value:

```
<keydef keys="test">
  <topicmeta><keywords><keyword>ABC</keyword></keywords></topicmeta>
</keydef>
```

you can modify a DITA-OT publishing parameter to have the value: `$(xpath_eval(doc('${rootMapURL}')/keydef[@keys='test']/keywords/keyword/text()))`. It will be expanded to the value of that specified key name.

- **\$(selection)** - The currently selected text content in the currently edited document. This variable can be used in a code template, in **Author** mode operations, or in a **selection plugin**.
- **\$(system(var.name))** - Value of the *var.name* Java System Property. The Java system properties can be specified in the command-line arguments of the Java runtime as *-Dvar.name=var.value*. If you are looking for operating system environment variables, use the **\$(env(VAR_NAME))** editor variable instead.

- **\${timeStamp}** - The timestamp, which is the current time in Unix format. For example, it can be used to save transformation results in multiple output files on each transformation.
- **\${tp}** - Total number of pages in the document. Used to display the total number of pages on each printed page in the **Editor / Print** Preferences page.
- **\${tsf}** - The transformation result file path. If the current opened file has an associated scenario that specifies a transformation output file, this variable expands to it.
- **\${uuid}** - Universally unique identifier, a unique sequence of 32 hexadecimal digits generated by the Java **UUID** class.
- **\${xmlCatalogFilesList}** - A list of file paths that point to all known XML catalog files, separated by semi-colons (;).
- **\${xpath_eval(expression)}** - Evaluates an XPath 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)}')}
```

Custom System Properties

A variety of Java system properties can be set in the application to influence its behavior. For information about how to do this, see [Setting a System Property \(on page 213\)](#).

com.oxygenxml.disable.http.protocol.handlers

- **Allowed Values:** `true` or `false`
- **Default Value:** `false`
- **Purpose:** By default, Oxygen JSON Editor uses the open source Apache HTTP Client software for HTTP(S) connections. If set to `True`, the default Java Sun HTTP(S) will be used instead. You will also lose **WebDAV** support and possibly other related features.

com.oxygenxml.present.license.reminders

- **Allowed Values:** `true` or `false`
- **Default Value:** `true`
- **Purpose:** When set to `false`, Oxygen JSON Editor will not display the messages that remind you to renew your Support and Maintenance Pack that covers your current license.

com.oxygenxml.enable.content.reference.caching

- **Allowed Values:** true or false
- **Default Value:** true
- **Purpose:** Enables content reference caching.

com.oxygenxml.eclipse.remove.grid.editing.mode

- **Allowed Values:** true or false
- **Default Value:** false
- **Purpose:** When set to false, Oxygen JSON Editor does not show the Grid editing mode when opening an XML document.

com.oxygenxml.default.java.accessibility

- **Allowed Values:** true or false
- **Default Value:** false
- **Purpose:** System property that can be set to true to force the default detection of java accessibility. If *com.sun.java.accessibility.AccessBridge* cannot be loaded, Oxygen JSON Editor forces the Java accessibility to be disabled.

com.oxygenxml.floating.license.timeout

- **Allowed Values:** An integer (minutes)
- **Default Value:** 120
- **Purpose:** Stores the time interval (in minutes) before floating licenses are released in case of application's inactivity.

com.oxygenxml.language

- **Allowed Values:** Language code (for example, en-us)
- **Default Value:** N/A
- **Purpose:** Property that holds the language code set during installation.

com.oxygenxml.default.options

- **Allowed Values:** A URL-type relative or absolute path.
- **Default Value:** N/A
- **Purpose:** Provides the path to an XML file containing default application options.

com.oxygenxml.customOptionsDir

- **Allowed Values:** A file system absolute path pointing to a folder.
- **Default Value:** N/A
- **Purpose:** 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. For more details, see [Importing/Exporting/Resetting Global Options \(on page 189\)](#).

com.oxygenxml.ApplicationDataFolder (Windows only)

- **Allowed Values:** A file system absolute path pointing to a folder.
- **Default Value:** %APPDATA%
- **Purpose:** When the application runs on Windows, you can set this property to change the location where the application considers that the *APPDATA* folder is located.

com.oxygenxml.editor.frameworks.url

- **Allowed Values:** A URL-type absolute path.
- **Default Value:** OXYGEN_DIR \frameworks
- **Purpose:** Changes the folder where the application considers that the main *frameworks* are installed. It has the same effect as changing the custom *frameworks* directory value in the [Location preferences page \(on page 88\)](#).

com.oxygenxml.editor.plugins.dir

- **Allowed Values:** The path can be specified with any of the following:
 - A URL or file path that is relative to the application's installation folder (for example: `-Dcom.oxygenxml.editor.plugins.dir=my-plugins`).
 - A system variable that specifies the file path (for example: `-Dcom.oxygenxml.editor.plugins.dir=${system(CONFIG)}/plugins`).
 - An environmental variable that specifies the file path (for example: `-Dcom.oxygenxml.editor.plugins.dir=${env(CONFIG)}/plugins`).
- **Default Value:** N/A
- **Purpose:** Specifies the directory where the application finds *plugins* to load.

com.oxygenxml.MultipleInstances

- **Allowed Values:** true or false
- **Default Value:** false
- **Purpose:** If set to true, multiple instances of the application are allowed to be started.

com.oxygenxml.xep.location

- **Allowed Values:** A file system absolute path pointing to a folder.
- **Default Value:** N/A
- **Purpose:** Points to a folder where RenderX XEP is installed.

com.oxygenxml.additional.classpath

- **Allowed Values:** A list of *JAR* (on page 639)-type resources separated by a classpath separator.
- **Default Value:** N/A
- **Purpose:** An additional list of libraries to be used in the application's internal class loader in addition to the libraries specified in the **lib** folder.

com.oxygenxml.user.home (Windows only)

- **Allowed Values:** A file system absolute path pointing to a folder.
- **Default Value:** `USERPROFILE` folder
- **Purpose:** Overwrites the user home directory that was implicitly detected for the application.

com.oxygenxml.use.late.delegation.for.author.extensions

- **Allowed Values:** `true` or `false`
- **Default Value:** `true`
- **Purpose:** All Java extensions in a *framework* configuration are instantiated in a separate class loader. When **true**, the *JAR* libraries used in a certain document type will have priority to resolve classes before delegating to the parent class loader. When **false**, the parent class loader will take precedence.

com.oxygenxml.stack.size.validation.threads

- **Allowed Values:** The number of bytes used for validation threads.
- **Default Value:** `5*1024*1024`
- **Purpose:** Some parts of the application (validation, content completion) that use the Relax NG parser sometimes require a larger *Thread* stack size to parse complex schemas. The default value should be more than enough.

com.oxygenxml.jing.skip.validation.xhtml.data.attrs

- **Allowed Values:** `true` or `false`
- **Default Value:** `true`
- **Purpose:** 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.

com.oxygenxml.report.problems.url

- **Allowed Values:** User-defined URL
- **Default Value:** N/A
- **Purpose:** The URL where a problem reported through the **Report Problem** dialog box is sent. The report is sent in XML format using the **report** parameter with the POST HTTP method.

com.oxygenxml.parallel.title.computing.threads

- **Allowed Values:** Integers
- **Default Value:** 4
- **Purpose:** The number of parallel threads that will be used to compute referenced topic titles. Increasing this value reduces the amount of time it takes to compute topic titles in the **DITA Maps Manager** view.

com.oxygenxml.hidpi.scaling

- **Allowed Values:** Numerical values between 1 and 2 (1, 1.5, and 2 have been tested, and for example, 1.5 is for 150% scaling)
- **Default Value:** N/A
- **Purpose:** Used to override the HiDPI scaling detection to force a specific scaling setting. This is helpful if you encounter scaling detection issues in Windows or Linux.

com.oxygenxml.prefer.plugin.classloader.context.loader

- **Allowed Values:** true or false
- **Default Value:** true
- **Purpose:** Used to instruct the application to use the plugin class loader when there is code that loads content (usually Xerces code) using the thread's class loader. For instance, if you have a plugin that specifies a certain Xerces version and you want to load that version instead of the one from **Oxygen's lib** directory.

com.oxygenxml.classic.file.output.stream.save

- **Allowed Values:** true or false
- **Default Value:** false
- **Purpose:** When set to true, the files are saved using a Java classic file output stream, which destroys the NTFS alternate data streams set on the file. However, this might prevent data loss in the rare occasions when Oxygen JSON Editor saves empty file content over shared network drives.

com.oxygenxml.format.indent.files.parallel

- **Allowed Values:** true or false
- **Default Value:** false

- **Purpose:** By default, when using the **Format and Indent Files** action from the **Project** view, only one thread will be used for the formatter. If the system property is enabled, up to four parallel threads are used by the operation, speeding up the processing when formatting very large or a large amount of documents.

Localizing the User Interface

Oxygen JSON Editor comes with the following built-in languages: English, French, German, Japanese, Dutch, and Chinese. 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.

How to Create an Interface Localization File

You can change the language of the Oxygen JSON Editor user interface by creating an interface localization file:

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 JSON Editor support team and ask them to send you the `translation.xml` sample file.
3. Open the `translation.xml` file in Oxygen JSON Editor. The file contains all the interface messages that can be translated and is updated at every new release with the latest additions. Here is a small 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. For example, set the `@description` attribute to `Spanish` and the `@lang` attribute to `es_ES`.

- For each `<key>` element, translate the `<comment>` (optional) and `<val>` elements. For example, 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>
```

- Open the **Preferences** dialog box (**Options > Preferences**) (on page 74), go to **Global**, and select the **Other language** option (on page 76). Browse for the `translation.xml` file.
- Restart the application.

Adding New Languages to the Interface

Oxygen JSON Editor provides a *plugin* extension is available in the **Oxygen SDK** that provides the ability to contribute a new translation language to the interface. By using this *plugin* extension, you can bundle the new language translation and that new language will be available in the **Languages** drop-down menu in the **Options > Preferences > Global** preferences page (on page 76).

Setting a Java Virtual Machine Parameter when Launching Oxygen JSON Editor

You can set Java Virtual Machine parameters (for example, if you want to increase the maximum amount of memory available) for the Oxygen JSON Editor **application launchers** (on page 211) or **command-line scripts** (on page 214). You can also create a **custom startup parameters file** (on page 214).

Setting Parameters for the Application Launchers

Increasing the Amount of Memory that Oxygen JSON Editor Uses on Windows and Linux

For Windows and Linux installations of Oxygen JSON Editor, the startup launchers for the application 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 in the installation directory that contains some startup parameters (such the `-Xmx` parameter, which is used for allocating memory for that particular application). If your installation contains these `.vmoptions` files, you can edit the parameters in them 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 increase the memory available to the Oxygen JSON Editor application on Windows:

1. Browse the installation directory of Oxygen JSON Editor.
2. Locate the `-Xmx` parameter in the `oxygenJSONEditor26.0.vmoptions` file. If it is located in a directory where you do not have write access, copy the file to another folder (where you do have write access), modify it there, and then copy it back to the original location.



Note:

The parameters from the `.vmoptions` file are used when you start Oxygen JSON Editor with the `oxygen` launcher (or with the desktop shortcut). If you use the command-line script (`<oxygen.bat>` or `<oxygen.sh>`), make sure you use the following procedure instead: [Setting Parameters in the Command-Line Scripts \(on page 214\)](#).



Tip:

By default, the maximum memory available to the application is about a quarter of the internal memory available on the machine. It is recommended to not use more than half of your existing physical RAM.

3. Restart Oxygen JSON Editor. Go to **Help > About** and verify the amount of memory that is actually available (see the **JVM Memory Used** in the last row in the **Copyright** tab). If Oxygen JSON Editor does not start and you receive an error message saying that it could not start the JVM, decrease the `-Xmx` parameter and try again.

Increasing the Amount of Memory that Oxygen JSON Editor Uses on macOS

To increase the memory available to Oxygen JSON Editor on macOS:

1. Create a file named `vmoptions.txt`.
2. Add the `-Xmx` argument (or other Java VM arguments), one per line, and do not add extra new lines at the beginning or end of the file. For example:

```
-Xmx4g
-Dcom.oxygenxml.editor.plugins.dir="$OXYGEN_HOME/plugins"
```


3. Make sure you save the file as plain text (in the **TextEdit**, go to **Menu > Format > Make plain text**) and copy the file to the **Contents** folder for the main application launcher. To show the **Contents** folder for the application launcher, right-click (or **Command+Single-Click**) the Oxygen JSON Editor icon in **Finder**, and choose **Show Package Contents**.

Setting a System Property

Depending on the operating system and type of installer, you can set a Java system property in multiple ways:

- **[Windows/Linux Installer]** When installing the application on Windows or Linux using the provided installation kit, you can create your own [custom startup parameters file \(on page 214\)](#) in the installation folder.
- **[macOS Installer]** Create a file named **vmoptions.txt** in the **Contents** folder within the application installation folder, similarly to [this procedure \(on page 212\)](#). Add each system property (or Java VM argument) on a separate line. For example:

```
-DpropertyName1=value1
-DpropertyName2=value2
```

- **[Windows Linux/Mac Startup Scripts]** The application also contains startup scripts in the installation folder. If you are using such scripts to start the application, you can follow this procedure to set system properties for them: [Setting Parameters in the Command-Line Scripts \(on page 214\)](#).



Note:

You can also set a system property through a parameter prefixed with **-Doxy** in the command line used to start the application:

```
oxygen25.1.exe "-Doxyproperty.name=value"
```

but this system property will be set immediately after the application starts and might not be available if it is needed sooner.

To check the value for a system property, you can select **Help > About** from the main menu and look in the **System properties** tab.

To view the list of Oxygen JSON Editor system properties, go to [Custom System Properties \(on page 205\)](#).

Disabling DPI Scaling

Some users may prefer the look of smaller icons in an HiDPI display. To achieve this, display scaling needs to be disabled for high DPI settings. To disable the DPI scaling, [set the following property \(on page 213\)](#):

```
sun.java2d.dpiaware=false
```

Setting Environment Variables

When started, the application inherits and can access all environment variables set in the operating system. All processes started by the application (for example, publishing using the DITA Open Toolkit engine or

starting external tools) also inherit the environment variables provided to the application. Depending on the operating system, environment variables can be set in various ways:

- **[Windows]** (Note: You will need Administrator permissions or to work with a system administrator):

1. Go to **Start > Edit the system environment variables > Environment Variables**.
2. Click **New** in the **System variables** section.
3. Specify the variable name and value in the **Name** and **Value** fields.
4. Click **OK**.
5. Restart Windows.

- **[Linux]:**

1. Append the following line to the `/etc/environment` file:

```
ENV_VAR_NAME=VALUE
```

2. Reboot the computer.

- **[macOS]:** There is no standard way to set an environment variable so that it is inherited by the applications regardless of the way they start.

To check the value for an environmental variable, you can select **Help > About** from the main menu and look in the **System properties** tab.

Setting Parameters in the Command-Line Scripts

If you start Oxygen JSON Editor with a command-line script (`oxygenJSONEditor.bat/oxygenJSONEditor.sh`), you have to add or modify parameters in the java command at the end of the script.

For example, to set the maximum amount of Java memory to 2 GB in **Windows**, add the **-Xmx** parameter to the last line of the `.bat` file like this:

```
%OXYGEN_JAVA% -Xmx2g -Dsun.java2d.noddraw=true ...
```

On **macOS/Linux**, add the **-Xmx** parameter (followed by a `\`) to a new line just above the `ro.sync.xml.Oxygen\` line (at approximately line 100) in the `.bat` file like this:

```
-cp "$CP"\  
-Xmx2g\  
ro.sync.xml.Oxygen\  
..
```

Creating Custom Startup Parameters File

You can create your own custom `.vmoptions` file 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.

For example: To specify a different language for all launchers you can use the custom **vmoptions** file called **custom_commons.vmoptions** and the content would look like this:

```
-Dcom.oxygenxml.language=French
```

For example: To increase the memory available for a specific tool, such as the **Compare Files** tool (**diffFiles.exe**), you can use a custom **vmoptions** file called **custom_diffFiles.vmoptions** and the content would look like this:

```
-Xmx2g
```

To be recognized and included, these custom startup parameter files must be saved in the installation directory of Oxygen JSON Editor.

How to Increase the Amount of Available Memory

Determining how to increase the amount of memory that is allocated to Oxygen JSON Editor depends on how you launch the application.

- **Windows/Linux Application Launcher** - If you start Oxygen JSON Editor using the default startup launcher that was created during a Windows or Linux installation, see [Increasing the Amount of Memory that Oxygen JSON Editor Uses on Windows and Linux \(on page 211\)](#).
- **macOS Application Launcher** - If you start Oxygen JSON Editor using the default startup launcher that was created during a macOS installation, see [Increasing the Amount of Memory that Oxygen JSON Editor Uses on macOS \(on page 212\)](#).
- **Command-Line Script** - If you start Oxygen JSON Editor using a command-line script, see [Setting Parameters in the Command-Line Scripts \(on page 214\)](#).
- **Custom Startup Parameters File** - You can also create your own custom startup parameters file and increase the memory using this file. For more information, see [Creating Custom Startup Parameters File \(on page 214\)](#).

5.

Working With Documents

Oxygen JSON Editor includes a variety of general features that can be used when working with most types of documents. More specialized features are available when working with specific types of documents, such as the various types of XML documents, [CSS \(on page 352\)](#), [JavaScript \(on page 439\)](#), [Markdown \(on page 449\)](#), and more. For details about those specialized features for specific types of documents, see [Editing Supported Document Types \(on page 352\)](#).

This chapter includes information about how to create and work with documents, working with projects, and various general editing features. This chapter also includes information about some of the special tools that are included in Oxygen JSON Editor, such as the [file and directory comparison tools \(on page 312\)](#).

Getting Familiar with the Interface

Oxygen JSON Editor includes several editing modes (**Text**, **Grid**, **Author**, and **Design**) and a large variety of helper views, menu actions, toolbars, and contextual menu functions.

There are various ways that you can [configure the layout of the views or editors \(on page 217\)](#), and you can [customize the toolbars \(on page 222\)](#).

Regardless of the editing mode that you are working with, the default layout consists of the following areas:

Menus

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

Toolbars

Easy access to common and frequently used functions. Each icon is a button that acts as a shortcut to a related function. Some of the toolbars are common for all *perspectives*, editing modes, and types of documents, while others are specific to the particular *perspective* or mode. Some toolbars are also *framework*-specific, depending on the type of document that is being edited. All the [toolbars can be configured \(on page 222\)](#) to suit your specific needs.

Helper Views

Oxygen JSON Editor includes a large variety of [dockable \(on page 638\)](#) 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 \(on page 217\)](#) 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 various contextual menu actions and other features to help streamline your editing tasks. Each file that has been opened has a tab at the top of the editing pane and there are [several ways to switch between tabs or move them \(on page 243\)](#).

Status Bar

The status bar at the bottom of the application contains some useful information when you are 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 \(on page 303\)](#) 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 \(on page 76\)](#) is selected, this section will show you when new messages have been received. The types of messages include the addition of new videos on the Oxygen JSON Editor website, the announcement of upcoming webinars and conferences where the Oxygen JSON Editor team will participate, and more.
- The memory consumption, including the memory used by the application and the maximum amount that is allocated to the application.
- If the [Show memory status option \(on page 78\)](#) is selected, a  **Free unused memory** icon is displayed in the bottom-right corner and you can use this icon to free up unused memory.

Configuring the Layout of the Views and Editors

All of the side-views available in Oxygen JSON Editor are [dockable \(on page 638\)](#) and there are various ways to configure and arrange the layout of the views and editing panes. You can also [configure the layout of the toolbars \(on page 222\)](#).

To open a view, select it from the **Window > Show View** menu. You can hide a view by closing it with the ✕ button at the top-right corner of the view, or with the **Window > Hide current view** action.

Arranging the Layout

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

creates a layout file containing the preferences of the saved layout. To load a layout, go to **Window > Load Layout**. To reset it, select **Window > Reset Layout**.

**Note:**

The **Load Layout** menu lets you select between the default layout, a predefined layout, or a custom layout. The changes you make using the **Load Layout** menu are also reflected in the **Application Layout** preferences page.

The changes you make to any layout are preserved between working sessions. The predefined layout files are saved in the [preferences directory \(on page 75\)](#) of Oxygen JSON Editor.

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 the following example, the `personal.xml` file was dragged over the `personal-schema.xml` file:

Hide or Float Views

Hide

To gain more editing space in the Oxygen JSON Editor window, click  **Toggle auto-hide** in any view. This button sets the view in the *auto-hide* state, making it visible only as a vertical tab, at the margins of the Oxygen JSON Editor window. To display a view in the *auto-hide* state, hover its side-tab with your cursor, or click it to keep the view visible until you click elsewhere.

Float

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 JSON Editor window.

Maximize the Editing Environment

You can configure the interface to maximize the editing area, leaving more vertical screen space available for the main editing pane. This is, for example, useful for presentations on low-resolution screens or for laptops with small screen space. You can use the following two actions that are available in the **Window** menu to create a near full-screen editing environment:

Maximize Editor Area

If toggled on, all side views are minimized to give you more horizontal space in the editing pane.

Hide All Toolbars

If toggled on, all toolbar buttons are hidden to give you more vertical space in the interface.

Tile/Stack Editor Actions

You can also tile or stack all open editors using the following actions from the toolbar or **Window** menu:

 Tile Editors Horizontally

Splits the editing area into horizontal tiles, one for each open file.

 Tile Editors Vertically

Splits the editing area into vertical tiles, one for each open file.

 Stack Editors

The reverse of the **Tile Editors Horizontally/Vertically** actions. Stacks all open editors.

 Synchronous Scrolling

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

**Note:**

When tiled, you can still drag and drop the editors, but note that they are docked in the same way as a window/view (instead of just tabs). You are actually rearranging the editor windows, so drag the editor tab and drop it to one of the sides of an editor (left/right/top/bottom). While dragging, you will see the dark gray rectangle aligned to one of the sides of the editor, or around the entire editor window. If you drop it to one of the sides, it will dock to that side of the editor. If you drop it when the rectangle is around the entire window of the editor, it will get stacked on top of that editor. You can also grab one of the stacked editors and tile it to one of the sides.

Split Editor Actions

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

-  **Split Editor Horizontally** - Splits the editor horizontally so that two editor panes are displayed with one on top of the other. This is useful for comparing and merging content between two documents.
-  **Split Editor Vertically** - Splits the editor vertically so that two editor panes are displayed side by side. This is useful for comparing and merging content between two documents.
-  **Unsplit Editor** - Removes a split action on the editing area.

To maximize or restore the editors, go to **Window > Maximize Editing Area**.

Switch, Move, or Hide Editor Tabs

Each file that has been opened has a tab at the top of the editing pane and there are several ways to switch between tabs or move them, and you can even hide the tabs to only show the currently open file.

**Note:**

If multiple file tabs are left open when you close the application, upon startup, Oxygen JSON Editor will not load the file content until you switch to the corresponding file tab. The tabs remain visible as a placeholders until the focus is switched to them. This helps to improve the application's startup time. If you want to disable this feature (meaning that the previously open files will all be re-loaded at



startup), deselect the **Load file content only when switching to its corresponding editor tab** option in the **Global** preferences page (*on page 77*).

Switching Editor Tabs

You can switch between editor tabs by using any of the following methods:

Mouse and Scroll Wheel

Of course, you can switch to a different editor tab by left-clicking the tab with your mouse, but when there are too many open tabs to fit on the screen, you can hover over the tab stripe and use the scroll wheel on your mouse to scroll to the left or right (same as using the two arrows on the far-right of the tab stripe).

Buttons on the Far-Right of the Tab Stripe (◀▶☰)

You can use the arrow buttons (◀▶) on the right side of the tab stripe to scroll to the left or right and the ☰ **Show List** button opens a pop-up window that displays all the open file tabs and allows you to select and switch to a specific open file.

Ctrl + Tab (Command + Tab on macOS) [NOTE: Ctrl + Page Down (Ctrl + Option + Right Arrow on macOS) does the same]

Switches to the next open tab in the order specified in the **Order of switching between editor tabs** option (*on page 78*).

Ctrl + Shift + Tab (Command + Shift + Tab on macOS) [NOTE: Ctrl + Page Up (Ctrl + Option + Left Arrow on macOS) does the same]

Switches to the previous open tab in the order specified in the **Order of switching between editor tabs** option (*on page 78*).

Window > Switch editor tab (Ctrl + F9 (Command + F9 on macOS))

This action opens a dialog box that allows you to switch to a particular editor tab by selecting it from a filterable list. This is especially helpful when you have a large amount of open file tabs and you want to switch to a certain tab this is not shown on the screen. It includes a search filter field and several options to help you find specific open file tabs.

The **Switch Editor Tab** dialog box contains the following options and features:

Search Filter

You can enter text in the filter field at the top of the dialog box to filter the list and search for specific open files. You can enter any number of terms, separated by space, and wildcards are allowed (for example, * to match any sequence of characters, or ? to match a single character). This field also has a history drop-down that allows you to select previously used search terms.

Match all terms

If this option is selected, only the files that match all of your search terms will be displayed. If you use a wildcard in the search filter, this option is automatically disabled.

Include file paths

If this option is selected, the search is expanded to include file paths, and also the paths are displayed in this dialog box.

Case sensitive

If this option is selected, the search operation will be case-sensitive.

List of Open File Tabs

All files that are currently open are displayed in the upper part of the main pane of the dialog box, followed by recently closed files. Files that have been modified but not yet saved are prefixed by an asterisk. To switch to a particular file tab, double-click the file or select it and click **OK**.

Moving Editor Tabs

You can move editor tabs by using any of the following methods:

Mouse Drag

You can use your mouse to drag editor tabs to a new location on the tab stripe.

Ctrl + Alt + Comma

Moves the current file tab one position to the left.

Ctrl + Alt + Period

Moves the current file tab one position to the right.

Hiding Editor Tabs

If you want to hide all the file tabs and only show the currently open file, select **Hide editor tabs** from the **Window** menu. This does not close the other tabs, just hides them. You can still navigate between tabs using keyboard shortcuts (**Ctrl + Tab**, **Ctrl + Shift + Tab**, **Ctrl + F6**, **Ctrl + Shift + F6**) or by selecting **Next editor** or **Previous editor** from the **Window** menu.

Resources

For more information about configuring the interface of Oxygen JSON Editor, watch our video demonstration:

<https://www.youtube.com/embed/anwjepfAdEk>



Tip:

To get more ideas for more advanced customization possibilities, watch our Webinar: **Working with DITA in Oxygen - Customizing the Editing Experience**. It offers a visual demonstration of how to



customize actions, document validation, content completion, new document templates, **Author** mode rendering, and more.

Related information

[Configuring Toolbars \(on page 222\)](#)

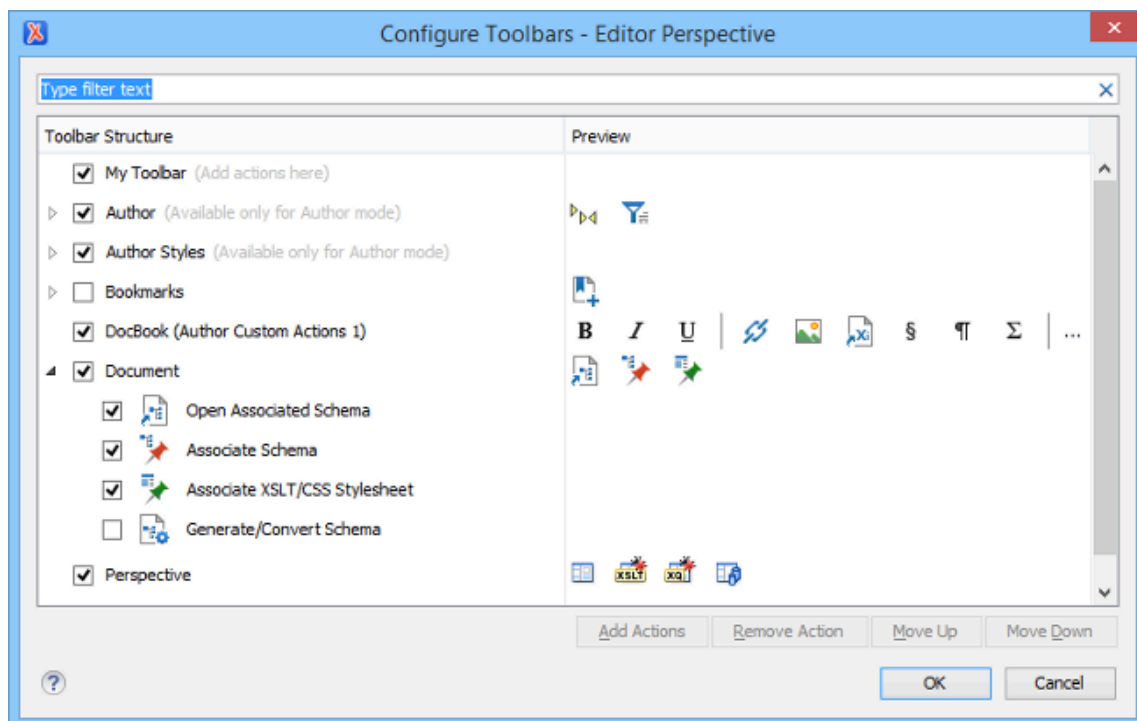
Configuring Toolbars

You can configure the toolbars in Oxygen JSON Editor to personalize the interface for your specific needs. You can choose which toolbars to show or hide in the current editor mode (**Text**, **Author**, **Design**, or **Grid**). 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 25. 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 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 (on page 639)* or 3rd party *plugins (on page 640)*.

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.

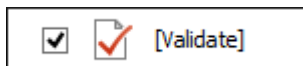
The **Configure Toolbars** dialog box also provides a variety of other ways to customize the layout in Oxygen JSON Editor.

Customize My Toolbar

You can customize the **My Toolbar** to include your most commonly used actions. By default, this toolbar is listed first. Also, it is hidden until you add actions to it and you can easily hide it with the **Hide "My Toolbar" Toolbar** action that is available when you right-click anywhere in the toolbar area.

Drop-down Menu Actions

Composite actions that are usually displayed as a drop-down menu can only be selected in one toolbar at a time. These actions are displayed in the **Configure Toolbars** dialog box with the name in brackets.



Configure External Tools Action

There is a  **Configure external tools** composite action that appears in the toolbar called **Tools**. It is a drop-down menu that contains any external tools that are configured in the **External Tools** preferences page.



Note:

If no external tools are configured, this drop-down menu is not shown in the toolbar.

Additional actions are available from the **Window** menu or contextual menu when invoked from a toolbar that allows you to further customize your layout. These actions include:

Reset Toolbars

To reset the layout of toolbars to the default setting, select the **Reset Toolbars** action from the contextual menu or **Window** menu.

Reset Layout

To reset the entire layout (including toolbars, editing modes, views, etc.) to the default setting, select **Reset Layout** from the contextual menu or **Window** menu.

Export Layout

You can use the **Export Layout** action that is available in the **Window** menu to export the entire layout of the application to share it with other users.

Hide Toolbars

You can use the **Hide Toolbar** action from the contextual menu to easily hide a displayed toolbar. When you right-click a toolbar it will be highlighted to show you which actions are included in that toolbar.

Related information

[Configuring the Layout of the Views and Editors \(on page 217\)](#)

Creating, Opening, Saving, and Closing Documents

Oxygen JSON Editor includes various features, actions, and wizards to assist you with creating new files and working with existing files. This section explains many of these features, including information on creating new documents, opening, saving, and closing existing files, searching documents, viewing file properties, and more.

Creating New Documents and Templates

Oxygen JSON Editor includes a helpful **New Document** wizard that allows you to customize and create new files from a large list of document types and built-in templates. You can also [create your own templates \(on page 229\)](#) and [share them with others \(on page 234\)](#).

To create a new document:

1. Click the .
2. Select the type of document that you want to create.



Tip:

You can use the text filter field at the top of the dialog box to search for a specific template.

New Document Wizard

Oxygen JSON Editor supports a wide range of document types. The **New Document** wizard presents the default associations between a file extension and the type of editor that opens the file. To customize these default associations, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **File Types** [\(on page 177\)](#).

The **New Document** wizard creates a skeleton document that may contain a root element, the document prolog, and possibly other child elements depending on options that are specific for each schema type. You can also [create your own custom document templates \(on page 229\)](#) and select them from this wizard.

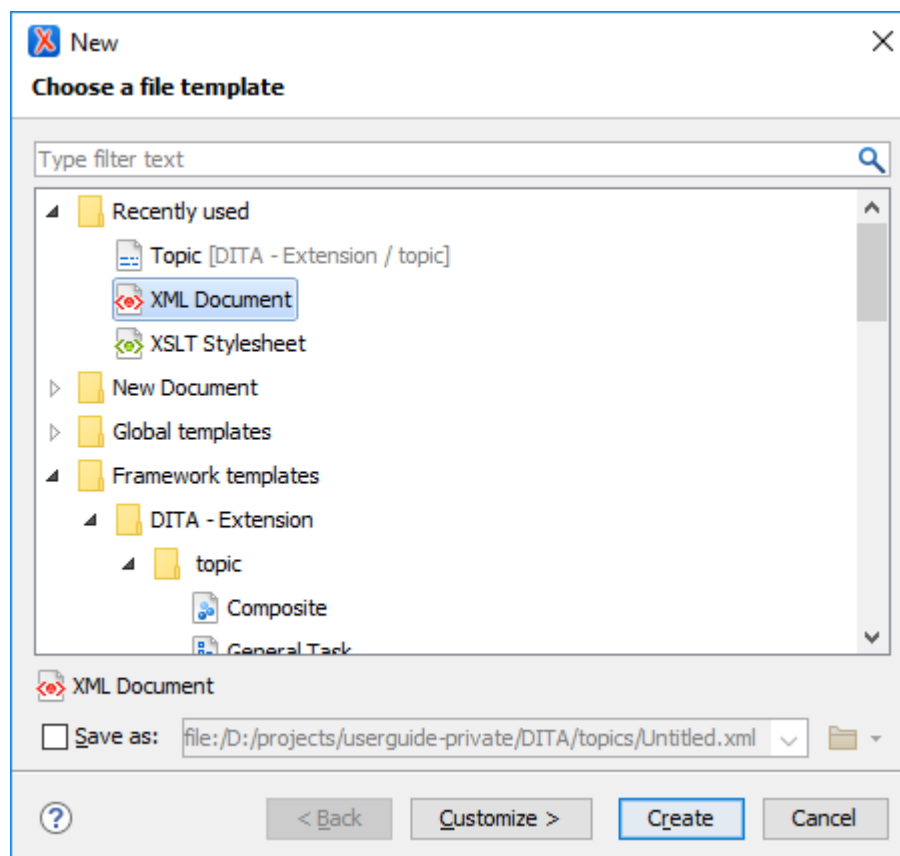
New Document Wizard

To create a new document using this wizard, follow these steps:

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

Result: The **New Document** wizard is displayed:

Figure 26. New Document Wizard



The first page of the wizard displays the supported document types and groups them in the following categories:

Recently Used

Contains the list of the most recently used file types. To clear the history of this folder, right-click an entry and select  **Remove all** (or select an entry and press **Ctrl + Delete** on your keyboard). To remove a single entry, right-click and select  **Remove** (or select the entry and press **Delete** on your keyboard).

User-defined template directory

You can add your own custom templates by [creating template files \(on page 229\)](#) in a directory and then adding that directory to the list of template directories that Oxygen JSON Editor uses in the [Document Templates preferences page \(on page 114\)](#). This user-defined directory will also appear in the **New Document** wizard.

Popular

Contains a list of popular framework templates. Each of these templates are marked as popular using a properties file that contains `popular` as one of the values for the `tags` property. For example, for the `OXYGEN_INSTALL_DIR/frameworks/dita/templates/topic/Topic.dita` template, there is a `Topic.properties` sibling file that contains `tags=popular`. Other document templates can also be added to the **Popular** category by [customizing them using a properties file for each one \(on page 230\)](#).

New Document

Contains the list of all supported document types. This list includes XML, CSS, .

Global Templates

Contains the list of built-in templates along with user-defined custom templates. You can [create your own custom document templates \(on page 229\)](#) and add them to the `templates` folder of the Oxygen JSON Editor installation directory.

Framework Templates

Contains the list of templates defined in the [Document Type configuration dialog box \(Templates tab\) \(on page 110\)](#) for each *framework*.

2. Select the type of document that you want to create.



Tip:

You can use the text filter field at the top of the dialog box to search for a specific template.

3. If you want to change the default name and path of the file, select the **Save as** option and specify the file path (the [Show "Save as" option to save newly created documents in the "New" document wizard option \(on page 134\)](#) must be selected in the **Save** preferences page). Otherwise, the file will be opened in a new tab with a default *untitled* name and the document path will not exist until you save it.



Note:

For DITA documents, the dialog box includes some additional options for generating a title, file name, and root ID attribute.

- If you want to use the default settings in the creation process, select **Create** at the bottom of the dialog box.

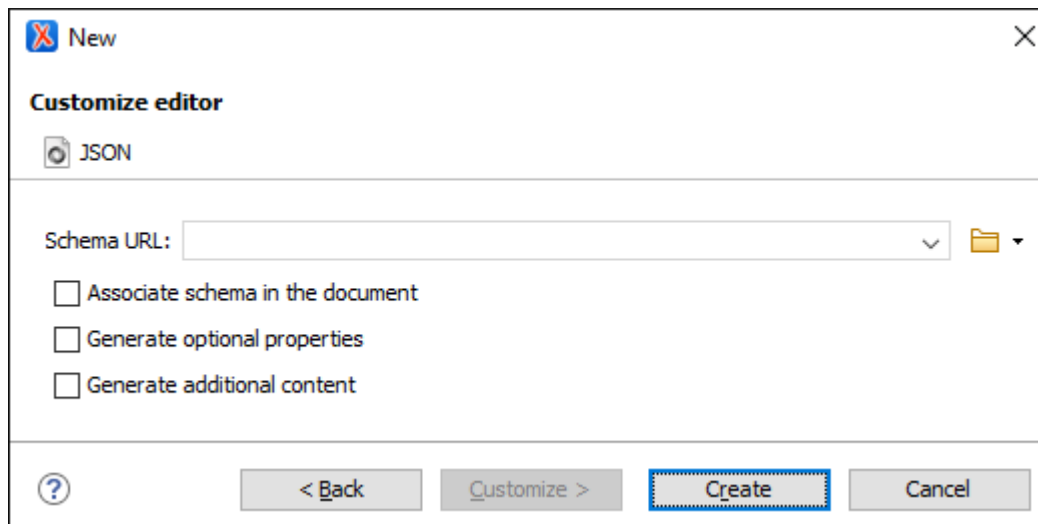
Result: The document is created using the default settings and the new file is opened in the appropriate editor.

- 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.

JSON Document Configuration Page

Figure 27. New JSON Configuration Wizard Page



If you select  **JSON** for the type of file you want to create and select the **Customize** option, the configuration dialog box will include the following options:

Schema URL

Specifies the path to a JSON schema file that will be used to generate *key-value* pairs.

Associate Schema in the Document

If you select this option, the JSON instance will be generated with the JSON schema associated directly in the document.

Generate Optional Properties

If you select this option, the JSON instance will be generated with optional properties that are defined in the JSON schema. Otherwise, only the required properties will be generated.

Generate Additional Content

If you select this option, the JSON instance will be generated with additional properties that are defined in the JSON schema as `additionalProperties` and additional items that are defined as `additionalItems` (in the case of an array).

YAML Document Configuration Page

Figure 28. New YAML Configuration Wizard Page

The screenshot shows a 'New' dialog box with a 'Customize editor' tab. Under the 'YAML' section, there is a 'Schema URL' field with a dropdown arrow and a folder icon. Below this are three checkboxes: 'Associate schema in the document', 'Generate optional properties', and 'Generate additional content'. At the bottom of the configuration area is a 'Property value' dropdown menu currently set to 'Default'. The bottom of the dialog features a help icon, a '< Back' button, a 'Customize >' button, a 'Create' button, and a 'Cancel' button.

If you select **YAML** for the type of file you want to create and select the **Customize** option, the configuration dialog box will include the following options:

Schema URL

Specifies the path to a JSON schema file that will be used to generate *key-value* pairs.

Associate Schema in the Document

If you select this option, the YAML instance will be generated with the JSON schema associated directly in the document.

Generate Optional Properties

If you select this option, the YAML instance will be generated with optional properties that are defined in the JSON schema. Otherwise, only the required properties will be generated.

Generate Additional Content

If you select this option, the YAML instance will be generated with additional properties that are defined in the JSON schema as `additionalProperties` and additional items that are defined as `additionalItems` (in the case of an array).

Property Value

Specifies the method of generating values for the properties. Possible values: **None**, **Default**, **Random**.

Creating New Document Templates

Oxygen JSON Editor allows you to create your own custom document templates and they will appear in the .

Creating a New Document Template

To create your own custom document template and have it appear in the new document 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 \(on page 197\)](#) in the template file content and they will be expanded when the files are opened. Also, see [Customizing Document Templates \(on page 230\)](#) for other template customization tips (for example, you could [add placeholders or hints \(on page 233\)](#) to assist authors).

2. Save the new document template and reference that location in Oxygen JSON Editor. There are several options for doing this:
 - **Saving the new template in a specific framework's directory** - Save the new template in a directory (for example, called **templates**) within that specific framework directory. Then open the **Document Type configuration dialog box (on page 89)** for that specific framework, go to the **Templates tab (on page 110)**, and click the **+** button in the bottom-right corner to add your new directory to the list. It is recommended that the reference be made relative to the framework directory (for example, `${frameworkDir}/templates`). You can also remove any existing entries in the list that aren't applicable or won't be used in your custom framework. Click **OK** to close the configuration dialog box and then **OK** or **Apply** to save your changes.
 - **Saving the new template in the Oxygen installation directory** - Save the new template in the **templates** directory of the Oxygen JSON Editor installation directory (`[OXYGEN_INSTALL_DIR]/templates`). Document templates saved in this directory will appear in the **Global templates** category in the .
 - **Saving the new template in a custom directory** - Save the new template in any directory of your choice and then add that directory to the list of templates in the **Document Templates preferences page (on page 114)**. This user-defined directory will appear in the along with all the new document templates that you save inside it. Click **OK** or **Apply** to save your changes.



Tip:

If you want to create a new template for a binary file (e.g. a zip archive), you need to add `.bin` to the end of the file name (for example, `*.zip.bin` or `*.epub.bin`). Otherwise, the files will be treated as XML/text documents and you will be prompted to choose the editor type.

**Attention:**

The name that you use to save the template will be the name that appears in the new document wizard, including capitalization, space, and characters (for example, **My Custom Template1.xml** will appear in the new file wizard as **My Custom Template1**). You can also configure the displayed name in a properties file by following the procedure found in the [Configure the Displayed Names for Document Templates \(on page 232\)](#) section.

3. Open the new document wizard (**New** toolbar button or) and you should see your custom template in the appropriate folder.

**Note:**

For DITA templates, they will also appear in the dialog box for creating new DITA topics, but if you [customize the template \(on page 230\)](#), you need to set the `type` property to **dita** in the corresponding properties file.

Related information

[Customizing Document Templates \(on page 230\)](#)

[Sharing Custom Document Templates \(on page 234\)](#)

Customizing Document Templates

Oxygen JSON Editor allows you to customize certain aspects of built-in 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 .

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 following these guidelines:
 - a. If you want to create a new properties file, you can use the **Properties** template found in the **New Document** folder in the . If you want to edit an existing template, you can find them within the subfolders in the **templates** folder for each framework (for example, the DITA topic properties file is located in: `OXYGEN_INSTALL_DIR/frameworks/dita/templates/topic/topic.properties`).
 - b. Use the same name as your custom template file except with a `.properties` extension (for example, `MyTemplate.properties`).
 - c. In this properties file, specify the paths to the icons that will be used in the new file wizard. The properties file should look like this:

```
type=general
smallIcon=../icons/Article_16.png
bigIcon=../icons/Article_48.png
```

**Tip:**

For DITA files, the `type` property must 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.

**Important:**

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 \(on page 229\)](#)), make sure that the path to the icons is relative to that directory.

2. Save the properties file in the same directory as your custom template.
3. Open the new file wizard () 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:

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 specifies 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 must 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 \(on page 231\)](#).
- 2. Save the properties file in the same directory as the document template.
- 3. Open the new document wizard ([using the methods described above \(on page 231\)](#)) 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.

**Note:**

The `filenamePrefix` and `filenameSuffix` properties can also have [editor variables \(on page 197\)](#) that do not require user interaction (i.e. editor variables that have `${ask()}` and `${answer()}` as values cannot be used).

Configure the Displayed Names for Document Templates

To change the name that is displayed for a document template, use the following procedure:

1. Create a new properties file or edit an existing one. If you create a new properties file, use the same name as the template file except with a **.properties** extension (for example, **MyTemplate.properties**).
2. Add a `displayName` property in the properties file:

```
displayName=My Template Name
```

**Tip:**

The names for [framework \(on page 639\)](#)-specific document templates (such as DITA *Topic* or DocBook *Article*, as you would see in the **Framework templates** folder in the **New** file wizard) can be translated via the internationalization support. In this case, the properties file should contain something like:

```
displayName=${i18n(tag)}
```

where *tag* refers to an entry in the **translation.xml** file for that specific framework (for example, **OXYGEN_INSTALL_DIR/frameworks/dita/i18n/translation.xml** for DITA).

3. Save the properties file in the same directory as the document template.
4. Open the new file wizard () and you should see the new name for the template.

Adding Placeholders or Hints in a Document Template

If a document template contains empty elements, it may not be clear to the Author what should be inserted in them. You can define placeholders in document templates that provide hints for Authors to help them understand what type of content should be added in any particular empty element within the document. The placeholder text is specified using a processing instruction and the placeholders are removed when the Author inserts content in the corresponding element.

To define placeholders in a document template to provide authors with hints, follow this procedure:

1. Edit the document template.
2. Add placeholders in the form of processing instructions within the elements where you want hints to be displayed when an Author creates a document from the template. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topic PUBLIC "-//OASIS//DTD DITA Topic//EN" "topic.dtd">
<topic id="pi">
  <title><?oxy-placeholder content="Enter a title"?></title>
  <shortdesc><?oxy-placeholder content="Writing short descriptions
    induces the writer to clarify the main thesis of the topic.
    We recommended a 50 word limit."?></shortdesc>
  <body>
    <p><?oxy-placeholder content="A paragraph element should be a self-contained
      unit dealing with one idea or point."?></p>
  </body>
</topic>
```



Important:

The elements that contain the placeholder processing instructions cannot contain other content/text, not even whitespace used for indentation. Otherwise, the placeholder will not be rendered properly.

3. Save the template file.
4. Use the to create a new document using your customized template and you should see the hints in the open document.

Resources

To see a visual demonstration of how to customize document templates and to get more ideas for other advanced customization possibilities, watch our Webinar: [Working with DITA in Oxygen - Customizing the Editing Experience](#).

Related information[Creating New Document Templates \(on page 229\)](#)[Sharing Custom Document Templates \(on page 234\)](#)

Sharing Custom Document Templates

Your [custom document templates \(on page 229\)](#) can be shared with the other members of your team so that they all have access to the templates in the . The best way to share them is by integrating them in an extended [framework \(on page 639\)](#) (document type) configuration and then sharing the whole framework with the other users.

Sharing Custom Document Templates

To share custom document templates with other members of your team:

1. , if you have not already done so.
2. [Create the new document template \(on page 229\)](#), if you haven't already done so.
3. Save the new template in a directory (for example, called **templates**) within your custom framework directory. Then open the **Document Type** configuration dialog box [\(on page 89\)](#) for that specific framework, go to the **Templates** tab [\(on page 110\)](#), and click the **+** button in the bottom-right corner to add your new directory to the list. It is recommended that the reference be made relative to the framework directory (for example, `${frameworkDir}/templates`). You can also remove any existing entries in the list that aren't applicable or won't be used in your custom framework.
4. Click **OK** to close the configuration dialog box and then **OK** or **Apply** to save your changes.
5. All that remains is to share the entire framework with anyone who needs to have access to the custom templates.

Opening Documents

To open a document in Oxygen JSON Editor, do one of the following:

- Go to to display the **Open File** dialog box.
- Go to **File > Open URL** to display a dialog box where you can specify a URL (defined by a protocol, host, resource path, and an optional port) or use the browsing actions in the  **Browse for remote file** drop-down menu.
- Select the **Open** or **Open with** action from the contextual menu of the [view \(on page 251\)](#).

Opening the Current Document in a 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 menu. 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, . This will take precedence over the default system application settings.

Saving Documents

You can save the document you are editing with one of the following actions:

- **File >  Save.**
- **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 All** - Saves all open documents.

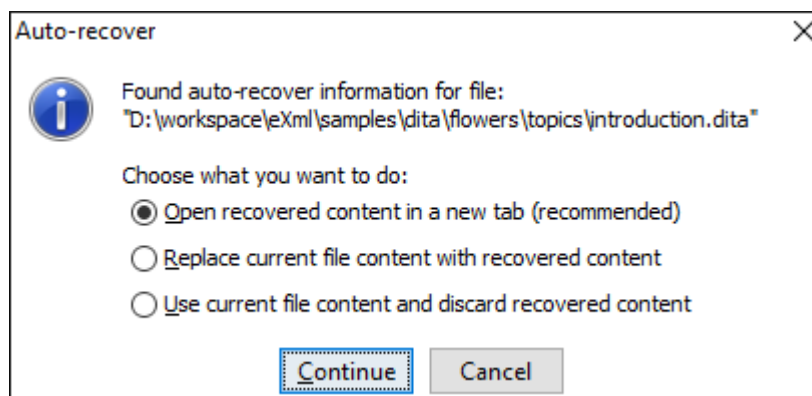
Auto Recover Documents

Oxygen JSON Editor includes an *Auto Recover* feature to help prevent losing unsaved content if you encounter an application or system crash. The feature is enabled by default and it automatically saves documents you are working on to a specified auto-recover file location. At every specified interval, all documents that have been modified since the last auto-save are saved to the specified location.

This feature is controlled by two options in the **Save** preferences page. You can disable it, or configure how often content is saved by selecting the desired value in the drop-down list of the **Save auto-recover information every option (on page 134)**, and you can specify the location of the saved documents in the **Auto-recover file location option (on page 134)**.

In the event of an application or system crash, once you restart the application, Oxygen JSON Editor looks for an auto-recover file for each document that is either automatically or manually reopened. If an auto-recover file is found, a dialog box is displayed with options for how to handle the recovered information.

Figure 29. Auto Recover Dialog Box



The dialog box offers the following choices:

- **Open recovered content in a new tab** - Opens the recovered document in a new tab.



Tip:

You can use the  **Compare Files** tool (available in the **Tools** menu) to compare the recovered content with the last saved version of the document.

- **Replace current file content with recovered content** - Replaces the content of the last saved version of the document with the content of the recovered version of the document and removes the auto-recover file from disk.
- **Use current file content and discard recovered content** - Discards the recovered document and retains the last saved version of the document.



Notes About the Auto-Recover Feature:

- The *Auto Recover* feature works for both local and remote files.
- For DITA projects, the *Auto Recover* feature also works for DITA maps opened in the **DITA Maps Manager**.
- The *Auto Recover* feature does NOT work if there is not enough space available on the disk where the [auto-recover file location is specified \(on page 134\)](#).
- The *Auto Recover* feature does NOT work on files opened in the [huge file editor \(on page 310\)](#) (if you select the **Optimize loading for huge files** option when opening large documents [\(on page 308\)](#)).

Closing Documents

To close open documents, you can simply click the close icon (✕) for the particular editor tab or use one of the following actions that are available by right-clicking the current editor tab (or from the **File** menu):

Close ()

Closes the currently selected editor.

If multiple files are opened, this action is available to close all opened editors in the current group/stack of tabs except for the one you are currently viewing.

Close All

If multiple files are opened, this action is available to close all opened editors in the current group/stack of tabs. If this action is selected from the **File** menu, it closes all opened editors in **all** groups/stacks of tabs.

Working with Remote Documents

Oxygen JSON Editor supports editing remote files, using the FTP, SFTP, WebDAV protocols. You can edit remote files in the same way you edit local files.

A WebDAV resource can be locked when it is opened in Oxygen JSON Editor by selecting the **Lock WebDAV files on open option (on page 183)** to prevent other users to modify it concurrently on the server. If a user tries to edit a locked file, Oxygen JSON Editor displays an error message that contains the lock owner's name. The lock is released automatically when the editor for that resource is closed in Oxygen JSON Editor.

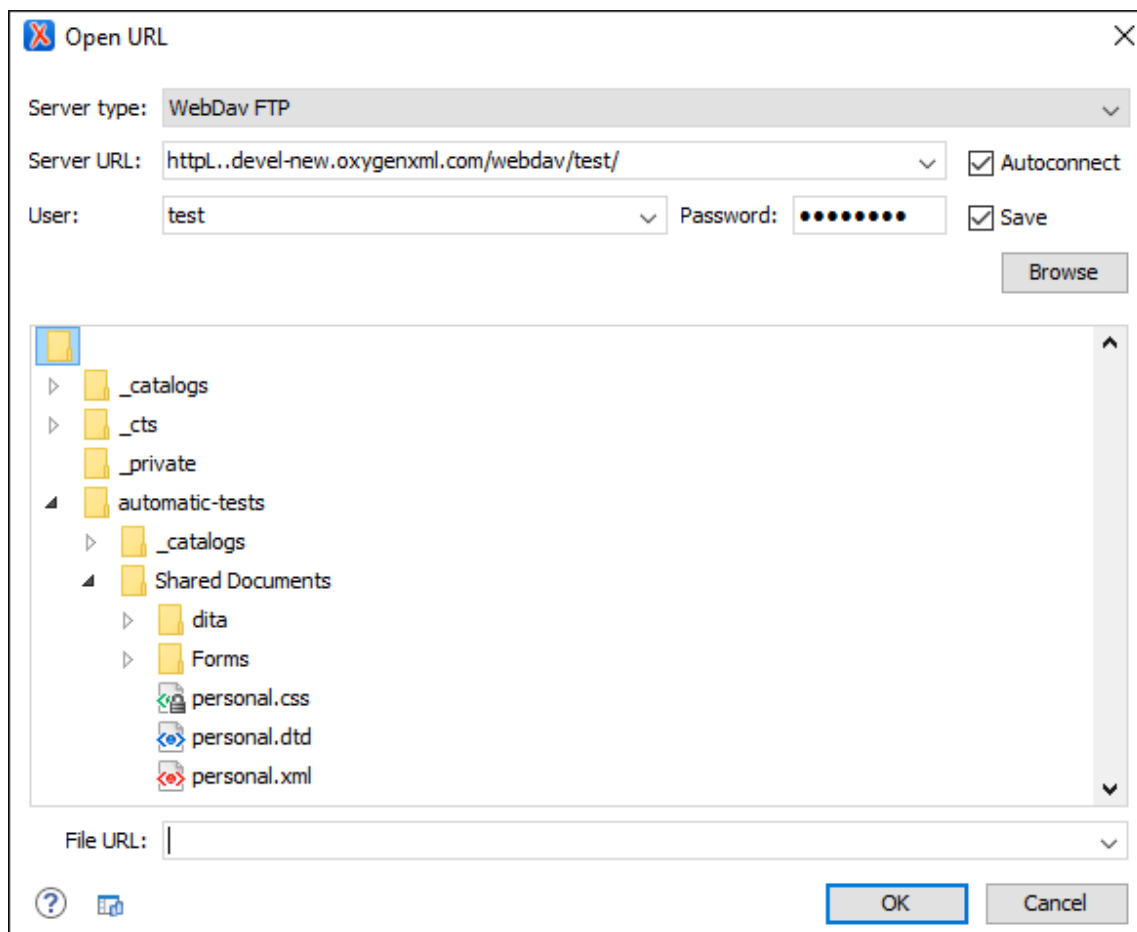
To avoid conflicts with other users when you edit a resource stored on a SharePoint server, you can **Check Out** the resource.

To improve the transfer speed, the content exchanged between Oxygen JSON Editor and the HTTP / WebDAV server is compressed using the GZIP algorithm.

Open URL

To open this dialog box, go to **File >  Open URL** (or click the ** Open URL** toolbar button), then choose the ** Browse for remote file** option from the drop-down action list.

Figure 30. Open URL Dialog Box



The displayed dialog box is composed of the following:

Server Type

Specifies the type of server. You can choose between:

- **WebDav FTP** - For generic HTTP/FTP/WebDav and other servers.
- **SharePoint Online** - For SharePoint Online servers.
- **SharePoint On-Premises** - For SharePoint (older version) servers.

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.

**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.

Save

If selected, the user and password are saved between editing sessions. The password is kept encrypted in the options file.

Browse

When you click 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.

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*).

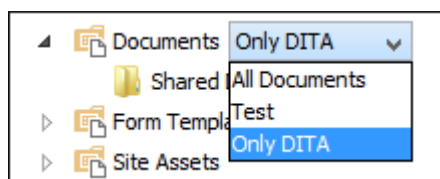
Depending on the type of node, a contextual menu offers customized actions that can be performed on that node. The contextual menu of a folder allows you to create new folders and documents, import folders and files, and to rename and delete the folder.

**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.

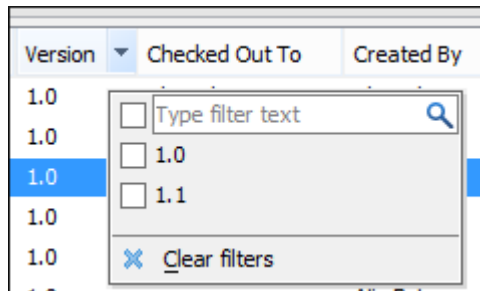
Figure 31. Drop-Down Menu to Select Which Items to Display



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.

Figure 32. 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 file browser dialog box by right-clicking a tree node and selecting the *Change permissions* menu item.

In this dialog box, the usual Unix file permissions *Read*, *Write*, and *Execute* are granted or denied for the file owner, owner group, and the rest of the users. The aggregate number of permissions is updated in the *Permissions* text field when it is modified with one of the checkboxes.

WebDAV over HTTPS

If you want to access a WebDAV repository across a non-secure network, Oxygen JSON Editor allows you to load and save the documents over the HTTPS protocol (if the server understands this protocol) so that any data exchange with the WebDAV server is encrypted.

When a WebDAV repository is first accessed over HTTPS, the server hosting the repository will present a security certificate as part of the HTTPS protocol, without any user intervention. Oxygen JSON Editor will use this certificate to decrypt any data stream received from the server. For the authentication to succeed you should make sure the security certificate of the server hosting the repository can be read by Oxygen JSON Editor. This means that Oxygen JSON Editor can find the certificate in the key store of the Java Runtime

Environment where 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

If Oxygen JSON Editor cannot connect to an HTTPS-capable server and an error message appears stating that it is "unable to find a valid certification path to the requested target", the HTTPS server is most likely either configured to use a self-signed certificate or to use a certificate issued by an unknown authority that the *Java Runtime Environment (JRE)* used by Oxygen JSON Editor does not trust.



Note:

For Windows, starting with version 26.0, by default, Oxygen JSON Editor uses the trusted root certificates from the Windows certificate store instead of the JRE cacerts store. To trust a certificate, the root certificate should be imported in the Windows Trusted Root certificates store.



Tip:

To make Oxygen JSON Editor accept a certificate even if it is invalid, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#), go to **Connection settings > HTTP(S)/WebDAV**, and select the **Automatically accept a security certificate, even if invalid** option.

To trust a certificate, follow this procedure:

1. Export a certificate into a local file using any HTTPS-capable web browser:

Chrome or Edge

- a. Navigate to the page that uses the certificate.
- b. Right-click the page and select **Inspect**.
- c. Select the **Security** tab.
- d. Click **View Certificate**. **Step Result:** A **Certificate** dialog box is displayed.
- e. Select the **Details** tab of the **Certificate** dialog box.
- f. Click the **Export** button.
- g. In the resulting dialog box, for the **Save as type** option, select **DER-encoded binary, single certificate (*.der)**.
- h. Save the certificate to the local file **server.der**.

Safari

- a. Navigate to the page that uses the certificate.
- b. If there is a "This connection is not private" message, click **Show Details** and in the expanded panel, click **view the certificate**.

- c. Otherwise, in the address bar, click the *padlock icon* on the left side of the website name and in the displayed pop-up, click **Show Certificate**.
 - d. Another pop-up box is displayed showing information about the **certificate**. Drag the large **certificate** icon to a *Finder* window. A **.cer** file will be created in the indicated folder from *Finder*.
2. Import the local file into the JRE running Oxygen JSON Editor:
 - a. Open a text-mode console with administrative rights. If Oxygen JSON Editor has been installed in a user's home directory and includes a bundled JRE, administrative rights are not required. In all other cases, administrative rights will be required.
 - b. Go to the **lib/security** directory of the JRE running Oxygen JSON Editor. You can find the home directory of the JRE in the *java.home* property that is displayed in the **About** dialog box (**System properties** tab).

**Note:**

On macOS, for the distribution of Oxygen JSON Editor that bundles the JRE from Oracle, the JRE uses the **.install4j/jre.bundle/Contents/Home/jre/lib/security/cacerts** path within its installation directory.

- c. Run the following command:

```
..\..\bin\keytool -import -trustcacerts -file server.cer -keystore cacerts
```

The **server.cer** file contains the server certificate, created during the previous step. The *keytool* requires a password before adding the certificate to the JRE *keystore* ([on page 639](#)). The default password is `changeit`. If someone changed the default password, then that person is the only one who can perform the import.

**Tip:**

If you need to import multiple certificates, you need to specify a different alias for each additional imported certificate with the `-alias` command-line argument, as in the following example:

```
..\..\bin\keytool -import -alias myalias1 -trustcacerts -file
server1.cer -keystore cacerts
```

```
..\..\bin\keytool -import -alias myalias2 -trustcacerts -file
server2.cer -keystore cacerts
```

3. Restart Oxygen JSON Editor.

Related information

[HTTP\(S\) Preferences \(on page 182\)](#)

HTTP Authentication Schemes

Oxygen JSON Editor supports the following HTTP authentication schemes:

- **Basic** - The *basic* authentication scheme defined in the [RFC2617 specifications](#).
- **Digest** - The *digest* authentication scheme defined in the [RFC2617 specifications](#).
- **NTLM** - The *NTLM* scheme is a proprietary Microsoft Windows Authentication protocol (considered to be the most secure among currently supported authentication schemes).



Note:

For NTLM authentication, the user name must be preceded by the name of the domain it belongs to, as in the following example:

```
domain\username
```

- **Kerberos** ([on page 243](#)) - 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 JSON Editor implements the *Single sign-on* property (meaning that you can log on once and gain access to multiple services without being prompted to log on for each of them), based on the **Kerberos** protocol and relies on a *ticket-granting ticket (TGT)* that Oxygen JSON Editor obtains from the operating system.



Restriction:

This *Single sign-on* support is not available for SharePoint integrations.

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
```

Switching, Moving, or Hiding Editor Tabs

Each file that has been opened has a tab at the top of the editing pane and there are several ways to switch between tabs or move them, and you can even hide the tabs to only show the currently open file.



Note:

If multiple file tabs are left open when you close the application, upon startup, Oxygen JSON Editor will not load the file content until you switch to the corresponding file tab. The tabs remain visible as placeholders until the focus is switched to them. This helps to improve the application's startup time. If you want to disable this feature (meaning that the previously open files will all be re-loaded at



startup), deselect the **Load file content only when switching to its corresponding editor tab** option in the **Global** preferences page (*on page 77*).

Switching Editor Tabs

You can switch between editor tabs by using any of the following methods:

Mouse and Scroll Wheel

Of course, you can switch to a different editor tab by left-clicking the tab with your mouse, but when there are too many open tabs to fit on the screen, you can hover over the tab stripe and use the scroll wheel on your mouse to scroll to the left or right (same as using the two arrows on the far-right of the tab stripe).

Buttons on the Far-Right of the Tab Stripe (◀ ▶ ☰)

You can use the arrow buttons (◀ ▶) on the right side of the tab stripe to scroll to the left or right and the ☰ **Show List** button opens a pop-up window that displays all the open file tabs and allows you to select and switch to a specific open file.

Ctrl + Tab (Command + Tab on macOS) [NOTE: Ctrl + Page Down (Ctrl + Option + Right Arrow on macOS) does the same]

Switches to the next open tab in the order specified in the **Order of switching between editor tabs** option (*on page 78*).

Ctrl + Shift + Tab (Command + Shift + Tab on macOS) [NOTE: Ctrl + Page Up (Ctrl + Option + Left Arrow on macOS) does the same]

Switches to the previous open tab in the order specified in the **Order of switching between editor tabs** option (*on page 78*).

Window > Switch editor tab (Ctrl + F9 (Command + F9 on macOS))

This action opens a dialog box that allows you to switch to a particular editor tab by selecting it from a filterable list. This is especially helpful when you have a large amount of open file tabs and you want to switch to a certain tab this is not shown on the screen. It includes a search filter field and several options to help you find specific open file tabs.

The **Switch Editor Tab** dialog box contains the following options and features:

Search Filter

You can enter text in the filter field at the top of the dialog box to filter the list and search for specific open files. You can enter any number of terms, separated by space, and wildcards are allowed (for example, * to match any sequence of characters, or ? to match a single character). This field also has a history drop-down that allows you to select previously used search terms.

Match all terms

If this option is selected, only the files that match all of your search terms will be displayed. If you use a wildcard in the search filter, this option is automatically disabled.

Include file paths

If this option is selected, the search is expanded to include file paths, and also the paths are displayed in this dialog box.

Case sensitive

If this option is selected, the search operation will be case-sensitive.

List of Open File Tabs

All files that are currently open are displayed in the upper part of the main pane of the dialog box, followed by recently closed files. Files that have been modified but not yet saved are prefixed by an asterisk. To switch to a particular file tab, double-click the file or select it and click **OK**.

Moving Editor Tabs

You can move editor tabs by using any of the following methods:

Mouse Drag

You can use your mouse to drag editor tabs to a new location on the tab stripe.

Ctrl + Alt + Comma

Moves the current file tab one position to the left.

Ctrl + Alt + Period

Moves the current file tab one position to the right.

Hiding Editor Tabs

If you want to hide all the file tabs and only show the currently open file, select **Hide editor tabs** from the **Window** menu. This does not close the other tabs, just hides them. You can still navigate between tabs using keyboard shortcuts (**Ctrl + Tab**, **Ctrl + Shift + Tab**, **Ctrl + F6**, **Ctrl + Shift + F6**) or by selecting **Next editor** or **Previous editor** from the **Window** menu.

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 open editors in the current group/stack of tabs except for the one you are currently viewing.

Close Files to the Right

Closes all open editors to the right of the currently selected editor.

Close All

If multiple files are opened, this action is available to close all open editors.

Move editor tab to the left (Ctrl + Alt + Comma)

Moves the current editor tab one position to the left.

Move editor tab to the right (Ctrl + Alt + Period)

Moves the current editor tab one position to the right.

Reopen last closed editor Ctrl + Alt + T (Command + Option + T on macOS)

Reopens the last closed editor.

Maximize Editing 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 open files in the current group/stack of tabs to the current project.

Copy Location

Copies the disk location of the file.

Show in Explorer (Show in Finder on macOS)

Opens the Explorer to the file path of the file.

Viewing File Properties

The **Properties** view displays information about the currently edited document. The information includes:

- Character encoding.
- Full path on the file system.
- Schema used for content completion and document validation.
- Document type name and path.
- Associated transformation scenario.
- Read-only state of a file.
- Bidirectional text (left to right and right to left) state.
- Total number of characters in the document.

- Line width.
- Indent with tabs state.
- Indent size.

The view can be accessed from **Window > Show View > Properties**.

To copy a value from the **Properties** view in the clipboard (for example, the full file path), use the **Copy** action available on the contextual menu of the view.

Simple Text Editor

Oxygen JSON Editor provides a simple text editor that supports various document types and includes various editing features.

Types of Files That are Supported in the Simple Text Editor

The types of files that can be created and edited in the simple text editor include:

- Java
- C++
- C
- Dockerfile
- PHP
- Perl
- Properties
- SQL
- PowerShell
- Batch
- Python
- Text

Features Available in the Simple Text Editor

When editing files in the simple text editor, the features that are available include the following:

- **Project Support** - The unique [features that are designed to help you work with projects \(on page 248\)](#) are available for all types of files.
- **Shortcut Actions** - Many of the shortcut actions that are available in **Text** mode are also available in the simple text editor.
- **Drag and Drop** - The normal drag and drop support is available in the simple text editor.
- **Syntax Highlighting** - Non-XML files also support syntax highlighting with dedicated coloring schemes. To customize them, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Syntax Highlight (on page 151)**. Select and expand the appropriate section in the top pane for the type of file you are editing and you can see the effects of your changes in the **Preview** pane.

- **Find/Replace** - You can use the  **Find/Replace** action ([on page 273](#)) to find or replace all the occurrences of a word or string of characters in any type of file that you are editing.
- **File Comparison Tool** - The **Compare Files** tool ([on page 313](#)) can also be used to compare non-XML files.

Using Projects to Group Documents

Oxygen JSON Editor includes a **Project view** ([on page 251](#)) that helps you organize your projects. Oxygen JSON Editor offers a variety of helpful features for working with projects and makes it easy to share your projects with other members of your team. This section presents various unique features that will help you to create and work with projects.

Creating a New Project

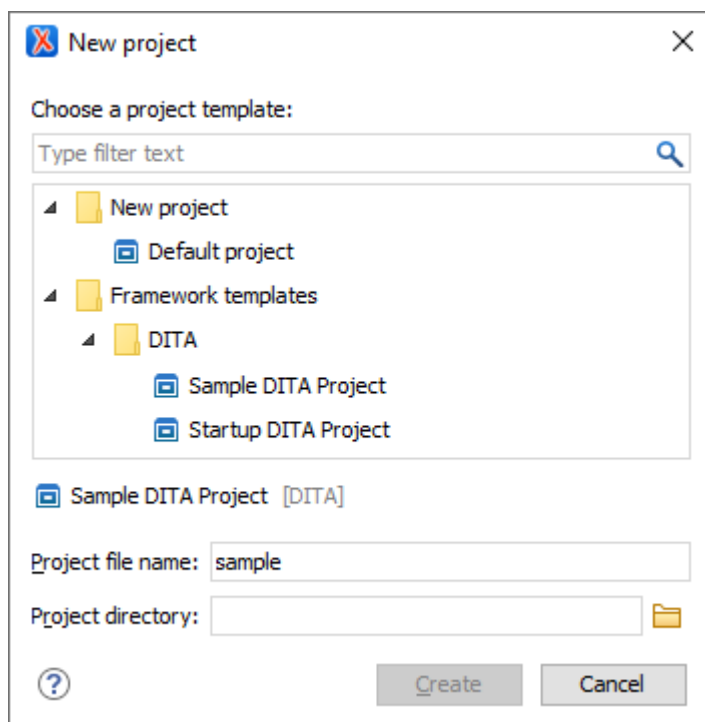
Oxygen JSON Editor allows you to organize your XML-related files into projects. This helps you manage and organize your files and also allows you to perform batch operations (such as validation and transformation) over multiple files. You can also [share your project settings and transformation/validation scenarios \(on page 261\)](#) with other users. Use the **Project view** ([on page 251](#)) 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 new project wizard:

Figure 33. New Project Wizard



With the exception of the *Default project* template, which is a pseudo-template and does not exist on the local disk (it is used only to create a new **.xpr** file), project templates are actually ZIP archives (with a **.zxpr** extension) and are stored within the file template directory structure (for example, **frameworks\dita\templates\sample-project\Sample DITA Project.zxpr**).

**Tip:**

Archives with a **.zxpr** extension can be edited in the [Archive Browser view \(on page 480\)](#).

After selecting a project template, you can specify the following:

Project file name

Specifies the name of the new project file. Oxygen JSON Editor provides a default proposal for the file name based on the following rules:

- If there is an **.xpr** file inside the archive, its name is used.
- Otherwise, the name of the template is used.

Project directory

Specifies the directory where the archive content will be extracted.

**Note:**

The archive should not contain an extra single folder as the root. For the **Project directory** path to work properly, the archive must have the **.xpr** file on the first level, along with the other resources (files and folders).

Once you are done, click the **Create** button to begin the creation process. Oxygen JSON Editor extracts the content from the archive inside the path specified in the **Project directory** field.

Editor Variables in Project Templates

By default, the editor variables inside project resources created from a project template are not resolved. To start having them resolved, the project template [must be customized \(on page 230\)](#) by using the `expandEditorVariablesIncludeFilter` property. This filter determines the resources where the editor variables will be resolved. If you need to exclude a subset of resources from the set specified by the `expandEditorVariablesIncludeFilter` property, the `expandEditorVariablesExcludeFilter` property can be used.

**Note:**

Usually, project files (***.xpr**), framework files (***.framework**), and framework extension scripts (***.exf**) should be excluded from the editor variable resolving process.

The values of the inclusion and exclusion filters can be file paths relative to the project directory that can use wildcards or simply wildcards. Each filter can have multiple values, separated by spaces.

Possible filter values:

- `./*` - Matches all resources from the first level in the project directory.
- `*` or `./**` - Matches all resources on all levels inside the project directory.
- `dir1/dir2/*.dita` - Matches all `.dita` files from `[PROJECT_DIR]/dir1/dir2`, but not from subdirectories of `dir2`.
- `dir1/dir2/**/*.dita` - Matches all `.dita` files from `[PROJECT_DIR]/dir1/dir2`, including those from subdirectories of `dir2`.
- `dir1/**/*.*` - Matches all resources on all levels inside `[PROJECT_DIR]/dir1`.
- `dir1/article1.xml`, `dir2/article2.xml` - Matches only the two `.xml` files.
- `./**/*_suffix.md`, `./**/prefix_*.html` - Matches all `.md` files with names that end with `_suffix` and all `.html` files with names that start with `prefix_`.

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 macOS - ).

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 macOS - ).

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 project contextual menu and the [Project view \(on page 251\)](#) will display the existing files and subdirectories. If your files are scattered among 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 (macOS 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 \(on page 251\)](#) 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 \(on page 251\)](#) are marked with a special icon.



Note:

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

For more information on managing projects and their content, see [Project View \(on page 251\)](#).

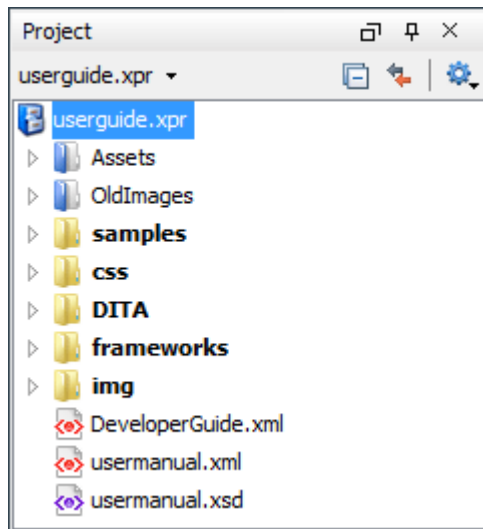
For more details about how you can share projects with other users, see [Sharing a Project - Team Collaboration \(on page 261\)](#).

Related information

[Using Projects to Group Documents \(on page 248\)](#)

Project View

The **Project** view is designed to assist you with organizing and managing related files grouped in the same XML project. The actions available in the contextual menu and on the toolbar associated to this panel allows you to create XML projects and provide shortcuts to various operations for the project documents.

Figure 34. Project View

By default, the view is positioned on the left side of Oxygen JSON Editor, 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).

Project View Toolbar

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 some actions to open a project or create a new one. You can use this history drop-down menu to quickly switch to a recently opened project. If you enable the **Remember layout changes for each project** option in the **Application Layout** preferences page (*on page 85*), the application will remember the layout, open files, and editing location for your session when you switch projects.

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

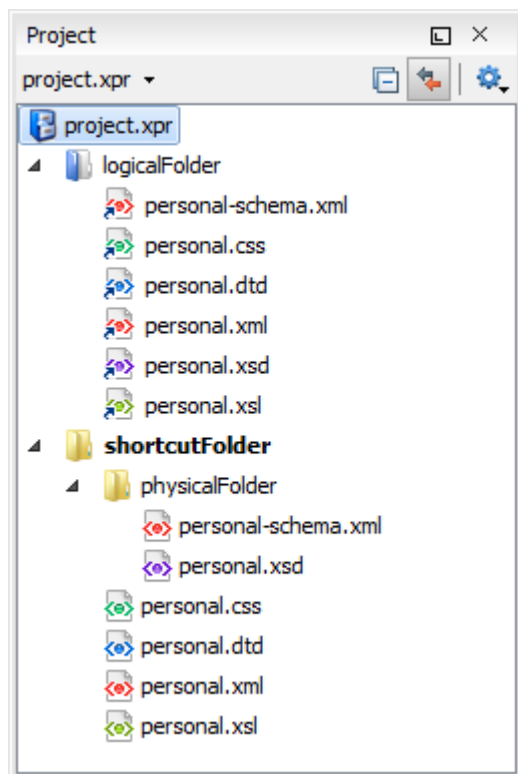
When selected, the currently edited file (from the main editor or from the DITA Maps Manager view) is highlighted in the project tree, if the file is found in the project.

File Explorer Area

The rest of the view is basically a file explorer similar to most other commonly used file explorers. The XML project (**.xpr** file) is a logical container with a collection of resources (folders and files). The types of resources displayed include:

- **Logical folders with Linked folders/files** - Marked with a blue icon on Windows and Unix/Linux () and a magenta icon on macOS (), they help you group files within the project. This folder type is used as containers for linked resources (shortcuts). The icons for file shortcuts include a shortcut symbol () and names of folder shortcuts are displayed in bold text. The logical folders are created on the project root or inside other logical folders by using the contextual menu action **New > Logical Folder**, and the linked folders/files are added using **Add Files**, **Add Folder**, or by dragging and dropping files/folders from the view or the system file explorer.  **Remove from Project** can be used to remove them from the project and the  **Remove from Disk (Shift+Delete)** action can be used to remove them from both the project and the local file system.
- **Physical folders and files** - Marked with the operating system-specific icon for folders (usually a yellow icon on Windows and a blue icon on macOS). 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**, , **Copy**, and , **Paste**) or by dragging and dropping files/folders from the view or the system file explorer. Also, the contextual menu action  **Remove from Disk (Shift+Delete)** can be used to remove them from the project and local file system.

Figure 35. Project View with Both Types of Resources



Creating New Projects

The following action is available from the **New** menu when right-clicking any item, the **Project** menu, or from the drop-down menu in the top-left of the **Project** view:

 **New Project**

Opens a wizard that assists you with creating a new project. For more details, see [Creating a New Project \(on page 248\)](#).

Managing Project Contents

There are various contextual menu actions, shortcuts, and ways to organize the folders and files inside the project:

Creating New Folders and Files

Right-click any item > New > File

Opens a **New file wizard** ([on page 225](#)) that helps you create a new file and adds it to the project structure.

Right-click any item in a physical folder > 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.

Right-click any item in a logical folder > New > Logical Folder

Creates a logical folder in the tree structure (the icon is a magenta folder on macOS ).

Right-click on a logical folder > 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.

Adding Resources

You can add resources by using drag and drop (or  **Copy** and  **Paste**) actions from within the **Project** view or dragging them from the system file explorer. Files may have multiple instances within the folder system, but cannot appear twice within the same folder.

Adding Resources to Logical Folders

You can add resources to logical folders by using the following actions available in the contextual menu when invoked on a logical folder (or the project's root container):

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 for this action is different on macOS ).

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.

Removing Folders and Files

To remove logical folders or the linked resources inside them from the project, use  **Remove from Project** from the contextual menu (or press **Delete** on your keyboard).

To remove folders or files from both the project and the local file system, use  **Remove from Disk** from the contextual menu (or press **Shift+Delete** on your keyboard).

Moving Folders and Files

You can move the resources by using drag and drop actions from within the **Project** view (the **Enable drag-and-drop in Project view** option must be selected in the [View preferences page \(on page 185\)](#)).

You can also use the usual  **Cut**,  **Copy**, and  **Paste** actions to move resources in the project.

You can also move certain types of files (such as XML) or folders by using the **Refactoring > Move resource** action from the contextual menu. This action opens the **Move resource** dialog box that includes the following options:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, based upon the selected scope. You can select or configure the scope by using the  button.

Renaming Folders and Files

There are several ways to rename a folder or file in the project (this works for both physical and linked resources):

- Select **Rename** from the contextual menu.
- Press **F2** on your keyboard.
- Select the item, then click the name, and type the new name.

You also can rename certain types of files (such as XML) or folders by using the **Refactoring > Rename resource** action from the contextual menu. This action opens the **Rename resource** dialog box that includes the following options:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references of the renamed resource** - Select this option to update the references to the resource you are renaming. You can select or configure the scope by using the  button.

Opening Files

There are several ways to open a file:

- Double-click the file.
- Select it and press **Enter** on your keyboard.
- Right-click the file and select **Open**.
- If there are no other files open in the editor area, you can drag the file from the project tree and drop it in the editor area.
- If you want to choose the application or location where to open it, you can right-click the file and select **Open with**.

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.

Other Contextual Menu Actions

Numerous other actions are available in the contextual menu, depending on the type of file or folder where it is invoked from (some actions are available for multiple selected files):

Show in submenu

Explorer (Finder on macOS)

On Windows and macOS, the parent directory of the selected file or folder is presented in a specific Explorer/Finder window and the selected resource is highlighted. On Linux, the selected file or folder is not highlighted after opening its parent in the file explorer.

Terminal

Opens a console (terminal) at the location of the selected physical resource. If the resource is a file, it will start at the parent directory.

Copy Location

Copies an application-specific URL for the selected resource to the clipboard.

Refactoring submenu

Oxygen JSON Editor includes some refactoring operations that help you manage the structure of your documents. The following actions are available from the contextual menu in the

Refactoring submenu:

Rename resource (Available for certain types of XML documents or folders)

Opens the **Rename resource** dialog box ([on page 259](#)) where you can change the name of a resource. It also includes an option to update the references to the renamed resource and you can choose between various scopes for the operation.

Move resource (Available for certain types of XML documents or folders)

Opens the **Move resource** dialog box ([on page 259](#)) where you can choose a destination and change the name of a resource. It also includes an option to update the references to the moved resource and you can choose between various scopes for the operation.

Refresh

Refreshes the content.

Find/Replace in Files

Opens the **Find/Replace in Files** dialog box ([on page 277](#)) that allows you to find and replace text in multiple files.

XPath in Files

Opens the **XPath/XQuery Builder** view ([on page 474](#)) 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 ([on page 267](#)).

Check Spelling in Files

Allows you to [check the spelling of multiple files \(on page 299\)](#).

Format and Indent Files

Opens the **Format and Indent Files** dialog box ([on page 640](#)) that allows you to configure the format and indent ([pretty-print \(on page 640\)](#)) action that will be applied on the selected documents.

Compare

Allows you to compare multiple files or directories and the order of your selection determines where they are opened in the **Compare Files** ([on page 313](#)) or **Compare Directories** ([on page 332](#)) 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 \(on page 316\)](#). 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.

HTML to XML Well-formed (Available when selecting multiple resources)

Batch converts the selected HTML documents to be XML well-formed. This means that missing end tags will be added to applicable elements, unclosed tags will be properly closed, and quotes will be added to attribute values that were missing the quotes.

**Notes:**

- All selected HTML files are backed up before being processed (same path/name but with the ".bak" extension added at the end).
- Any detected conversion errors are grouped and listed in a dedicated tab in the **Results** pane at the bottom of the application.
- A brief report is displayed at the end of the operation.

Transform submenu

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.

Validate submenu

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. For EPUB files, this action triggers an **EPUB Validate and Check for Completeness** ([on page 483](#)) 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.

**Properties**

Displays the properties of the current file in a **Properties** dialog box.

Project Menu Actions

The following actions are available in the **Project** menu:

New Project

Opens a wizard that assists you with creating a new project. For more details, see [Creating a New Project \(on page 248\)](#).

Open Project (Ctrl + F2 (Command + F2 on macOS))

Opens an existing project. Alternatively, you can open a project by dropping an Oxygen JSON Editor XPR project file from the file explorer into the **Project** panel.



Notice:

When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

Save Project As

Allows you to save the current project under a different name.



Validate all project files

Checks if the project files are well-formed and their mark-up conforms with the specified DTD, XML Schema, or Relax NG schema rules. It returns an error list in the message panel.



Filters

Opens the **Project filters** dialog box that allows you to decide which files and directories will be shown or hidden.



Change Search and Refactor operations scope

Opens a dialog box that allows you to define the context of search and refactor operations.

Show Project View

Displays the **Project** view.

Reopen Project

Contains a list of links of previously used projects. This list can be emptied by invoking the **Clear history** action.

Moving/Renaming Resources in the Project View

The **Refactoring** submenu in the contextual menu of the **Project view (on page 251)** provides actions for moving or renaming certain types of XML resources in the current project while offering the option to update the references to the resources.

Moving Resources

You can move certain types of files (such as XML) by using the **Refactoring > Move resource** action from the contextual menu. This action opens the **Move resource** dialog box that includes the following options:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, based upon the selected scope. You can select or configure the scope by using the  button.

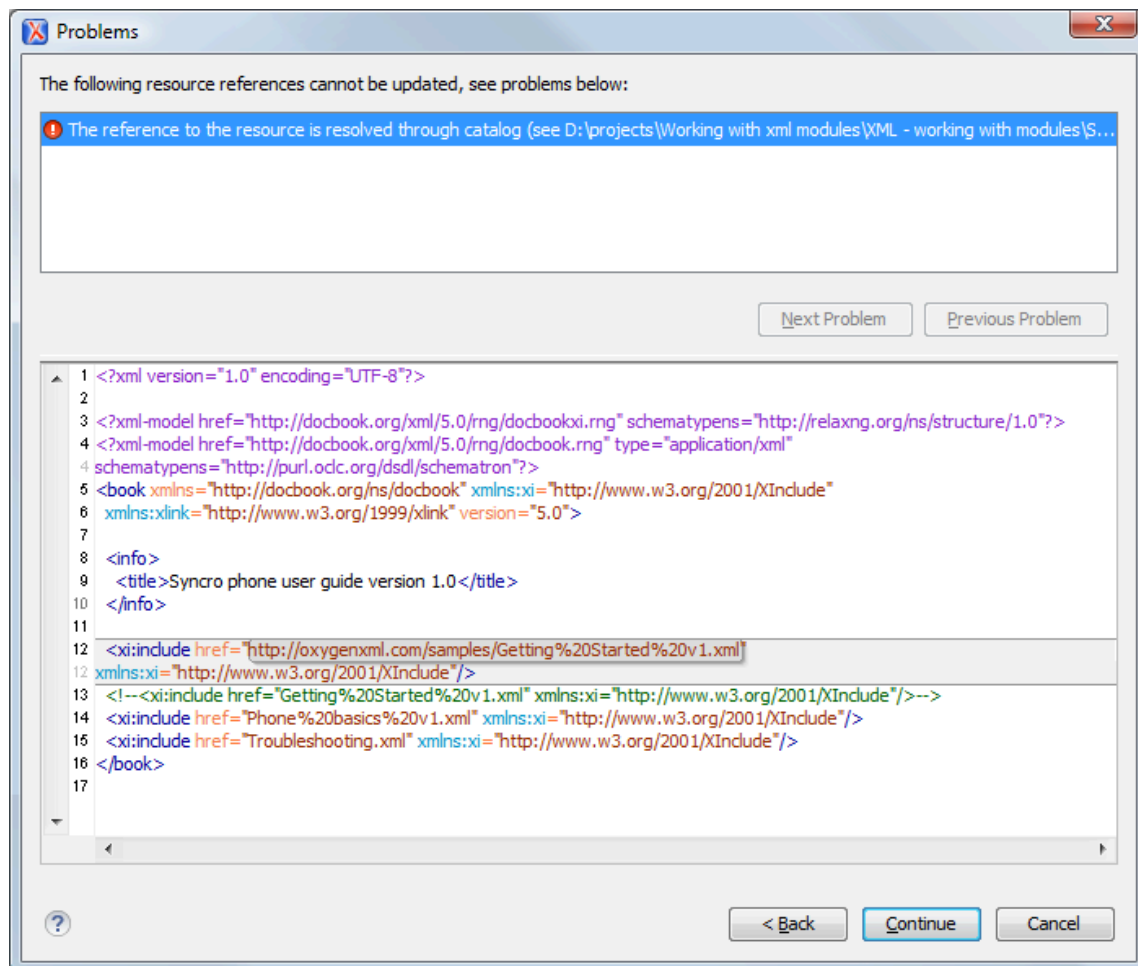
Renaming Resources

You can rename certain types of files (such as XML) by using the **Refactoring > Rename resource** action from the contextual menu. This action opens the **Rename resource** dialog box that includes the following options:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references of the renamed resource** - Select this option to update the references to the resource you are renaming. You can select or configure the scope by using the  button.

Problems Updating References of Moved/Renamed Resources

In some cases, the references of a moved or a renamed resource cannot be updated. For example, when a resource is resolved through an [XML Catalog \(on page 640\)](#) or when the path to the moved or renamed resource contains entities. For these cases, Oxygen JSON Editor displays a warning dialog box.

Figure 36. Problems Dialog Box

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 JSON Editor also allows you to store preferences, transformation scenarios, and validation scenarios at [project level \(on page 640\)](#) 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 has access to the same resources stored in the project file.

Sharing Preferences (Creating a Project-Level Options File)

To share options that are configured in certain preferences pages, you can store them in a *project file* (**.xpr** file extension) that can easily be shared with others. To do so, follow these steps:

1. [Recommended] You may want to use a fresh install for this procedure to ensure that you do not copy personal or local preferences.
2. In the **Project view (on page 251)**, create a project or open an existing one.
3. Open the **Preferences** dialog box (**Options > Preferences**) ([on page 74](#)).
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 (on page 640)** in each page.

**Note:**

Some pages do not have the **Project Options** button, since the options they host might contain sensitive data (such as passwords, for example) that is unsuitable for sharing with other users.

5. Click **OK** and close the **Preferences** dialog box.

All explicitly set values are now saved in the project file. You can then share the project file so that your team will have the same option configuration that you stored in the project file.

**Note:**

The project file extension (**.xpr**) must be preserved when the file is distributed to others.

**Notice:**

When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

Sharing 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** ([on page 251](#)), 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** ([on page 640](#)) in each validation scenario you want to be included in the project file.
3. Click **OK** to store the scenario in the project file.

You can then share the project file so that your team will have access to the same validation scenarios that you stored in the project file. When you create a scenario at the project level, the URLs from the scenario become relative to the project URL.

**Note:**

The project file extension (**.xpr**) must be preserved when the file is distributed to others.

**Notice:**

When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

Using Git for Collaboration

To assist you with team collaboration, sharing projects, and version control, an add-on is available that contributes a built-in Git client directly in Oxygen JSON Editor. The **Git Client** is developed independent of the normal **Oxygen** release cycle, so it is updated and improved more frequently than the main application. It is an optional tool so it needs to be installed as an add-on. Once installed, a **Git Staging** view is available that includes various actions that perform common Git commands, such as *push*, *pull*, *change branch*, *commit*, and more. It also includes a built-in tool for comparing and merging changes.

For more details, see [Git Client Add-on \(on page 486\)](#).

Using Subversion (SVN) for Collaboration

Oxygen JSON Editor also 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** [\(on page 251\)](#). This action opens the Syncro SVN Client and shows the selected project file in the **Working Copy** view.

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 JSON Editor project file (**.xpr**) that will be shared by all content authors.
2. Configure your own formatting preferences. To do this, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#), 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** [\(on page 640\)](#) 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** [\(on page 251\)](#).
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.

Search and Find/Replace Features

Oxygen JSON Editor includes advanced search capabilities to help you locate documents and resources. The search features are powered by [Apache Lucene](#). Apache Lucene is a free open source information retrieval

software library. You can perform simple text searches or more complex searches using the [Apache Lucene - Query Parser Syntax](#).



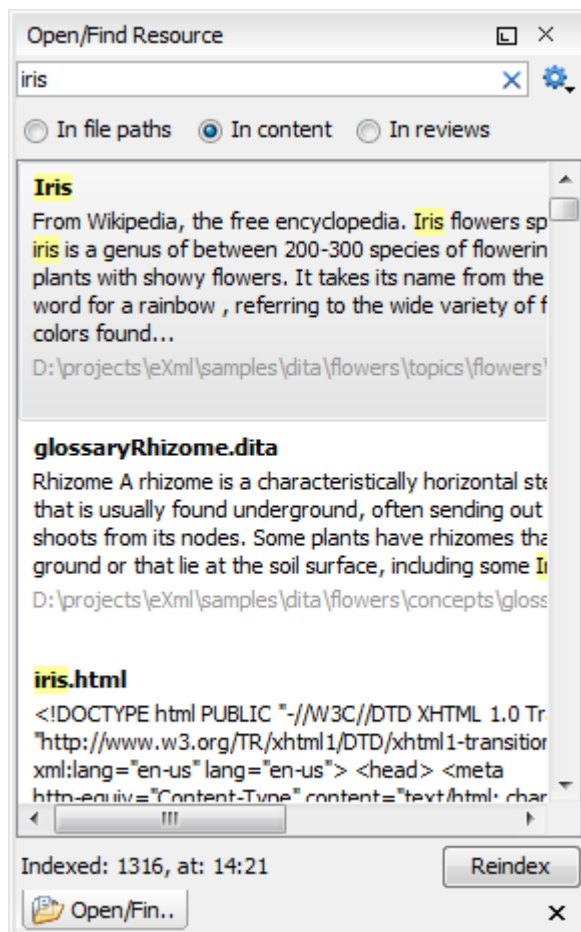
Note:

When Oxygen JSON Editor performs the indexing of resources, referenced content is not taken into account. For example, when DITA documents are indexed, the content referenced in a [@conref](#) or [@conkeyref](#) attribute is not parsed. The files that make up the index are stored on disk in the `[user_home_directory]\AppData\Roaming\com.oxygenxml.jsoneditor\lucene` folder.

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 JSON Editor layout, next to the **Project view** (on page 251). If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

Figure 37. Open/Find Resource View



You can use this view to find a file in the current Oxygen JSON Editor 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) by selecting the **In reviews** option (on page 265).

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 macOS)** or **Ctrl + UpArrow (Command + UpArrow on macOS)** on your keyboard. Pressing only the **DownArrow** key moves the selection to the list of results.



Note:

Searches are not case-sensitive. For example, if you search for `car` you get the same results as when you search for `Car`.



Tip:

Suffix searches are also supported, both for searching in the content of your resources and in their name. For this, you can use wildcards. If you search for `*ing` with the **in content** option selected, you will find documents that contain the word *presenting*. If you search for `*/samples/*.gif` with the **in file paths** option selected, you will find all the *gif* images from the **sampLes** directory.

Options Available in the View

The **Open/Find Resource** view offers the following options:

-  **Settings** - Drop-down menu that includes the following settings for the view:
 - **Clear Index** - Clears the index.
 - **Show description** - Presents the search results in a more compact form, displaying only the title and the location of the resources.
 - **Options** - Opens the [Open/Find Resource preferences page \(on page 178\)](#) where you can configure various search options. For example, you can specify a **Content language** that differs from the default UI language in case your document contains multiple languages.
- **In file paths** ([on page 272](#)) - Select this option to search for resources by their name or by its path (or a fragment of its path).
- **In content** ([on page 270](#)) - Select this option to search through the content of your resources.
- **In reviews** ([on page 272](#)) - 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 JSON Editor internal editors.
- **Open with** - Allows you to choose to open the document in the **Internal editor** or an external **System application**.
- **Show in Explorer** - Identifies the document in the system file explorer.
- **Copy Location** - Copies the file path and places it in the clipboard.

Indexing Process

The content of the resources used to search in is parsed from an index. The indexing is performed both automatically and on request. Automatic indexing is performed when you modify, add, or remove resources in the currently indexed project. If the index was never initialized, the index 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 \(on page 178\)](#).

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 click 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 JSON Editor project after the last caching operation. In this case, re-indexing the project files with the **Reindex** button enables the file to be found. The date and time of the last index operation are displayed below the file list.

Opening the Results

Once you find the files that you want to open, select them in the list and click the **Open** button (or double-click them). Each of the selected files is opened in [the editor associated with the type of the file \(on page 177\)](#).



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.

Resources

For more information about the **Open/Find Resource** feature and its search capabilities, watch our video demonstration:

<https://www.youtube.com/embed/PENoDNdaGao>

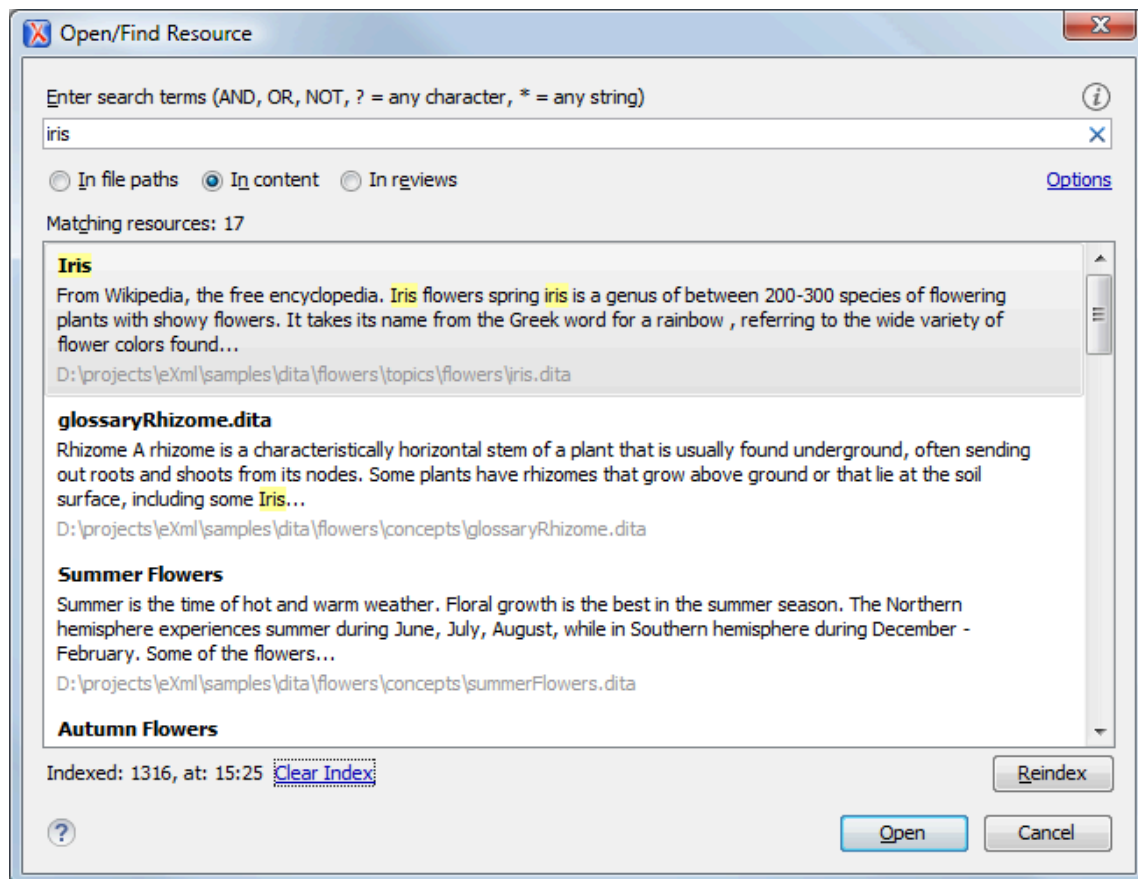
Related information

[Open/Find Resource Dialog Box \(on page 267\)](#)

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 macOS))**. 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 38. Open/Find Resource Dialog Box



You can use this dialog box to find a file in the current Oxygen JSON Editor 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).

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 macOS)** or **Ctrl + UpArrow (Command + UpArrow on macOS)** 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** ([on page 272](#)) - Select this option to search for resources by their name or by its path (or a fragment of its path).
- **In content** ([on page 270](#)) - Select this option to search through the content of your resources.
- **In reviews** ([on page 272](#)) - 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** ([on page 178](#)) where you can configure various search options. For example, you can specify a **Content language** that differs from the default UI language in case your document contains multiple languages.
- **Clear Index** - Clears the index.
- **Reindex** - Use this option to reindex your resources.

Contextual Menu Actions

A contextual menu is available on each search result and provides actions applicable to that particular document. These actions include:

- **Open** - Opens the document in one of Oxygen JSON Editor internal editors.
- **Open with** - Allows you to choose to open the document in the **Internal editor** or an external **System application**.

- **Show in Explorer** - Identifies the document in the system file explorer.
- **Copy Location** - Copies the file path and places it in the clipboard.

Indexing Process

The content of the resources used to search in is parsed from an index. The indexing is performed both automatically and on request. Automatic indexing is performed when you modify, add, or remove resources in the currently indexed project. If the index was never initialized, the index 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 \(on page 178\)](#).

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 click 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 JSON Editor project after the last caching operation. In this case, re-indexing the project files with the **Reindex** button enables the file to be found. The date and time of the last index operation are displayed below the file list.

Opening the Results

Once you find the files that you want to open, select them in the list and click the **Open** button (or double-click them). Each of the selected files is opened in [the editor associated with the type of the file \(on page 177\)](#).

Resources

For more information about the **Open/Find Resource** feature and its search capabilities, watch our video demonstration:

<https://www.youtube.com/embed/PENoDNdaGao>

Related information

[Open/Find Resource View \(on page 264\)](#)

[Open/Find Resource Preferences Page \(on page 178\)](#)

Searching in Content

To perform a search through the content of your resources, open the **Open/Find Resource** dialog box ([on page 267](#)) (from the **Find** menu or with **Ctrl + Shift + R (Command + Shift + R on macOS)**) or the **Open/Find Resource** view ([on page 264](#)) (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](#).

Complex Query Patterns Using Lucene Syntax

Using the [Apache Lucene - Query Parser Syntax](#) means you can perform any of the following searches:

- **Term Searches**

Searching for plain text:

```
Garden Preparation
```

- **Element-Specific Searches**

Searching for content that belongs to a specific element:

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

- **Wildcard Searches**

Using wildcards to make your search more permissive:

```
Garden Prepar?tion
```

- **Fuzzy Searches**

If you are not sure of the exact form of a term that you are interested in, use the fuzzy search to find the terms that are similar to the search term. To perform a fuzzy search, use the `~` symbol after the word that you are not sure of:

```
Garden Preparing~
```

- **Proximity Searches**

Use proximity searches to find words that are within a specific distance away. To perform a proximity search, use the `~` symbol at the end of your search. For example, to search for the word **Garden** and the word **Preparation** within 6 words of each other use:

```
"Garden Preparation"~6
```

- **Range Searches**

Use range searches to match documents whose element values are between the lower and upper bound specified in the range query. For example, to find all documents whose titles are between **Iris** and **Lilac**, use:

```
title:{Iris TO Lilac}
```

The curly brackets denote an exclusive query. The results you get when using this query are all the documents whose titles are between **Iris** and **Lilac**, but not including **Iris** and **Lilac**. To create an inclusive query use square brackets:

```
title:[Iris to Lilac]
```

- **Term Boosting 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 Flower**, a good idea is to prioritize the word **Autumn** since the word **Flower** occurs more often. To prioritize a word use the `^` symbol:

```
Autumn^6 Flower
```

- **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 \(on page 267\)](#) (from the **Find** menu or with **Ctrl + Shift + R (Command + Shift + R on macOS)**) or the [Open/Find Resource view \(on page 264\)](#) (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 \(on page 267\)](#) (from the **Find** menu or with **Ctrl + Shift + R (Command + Shift + R on macOS)**) or the [Open/Find Resource view \(on page 264\)](#) (by default, located on the left side of the editor), select the **In reviews** option, and in the search field enter the terms that you want to search for.

The following options are available:

- **Type** - Specifies whether you want to search for content in comments, tracked change insertions/deletions, or highlighted content.
- **Author** - Displays all the authors of the edits in your resources. The authors are collected when indexing. You can set a specific author for your search or search all of them.
- **Time** - Specifies the time when the edits that you are searching through were created.

Both the view and the dialog box display the edits that contain the search results and their parent topics along with a short description. To hide this description, go to **Settings** and deselect the **Show Description** option.

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 macOS)**. It is also invoked by the **Find/Replace** contextual menu action found in certain views.

You can use the **Find/Replace** dialog box to perform the following operations:

- Replace occurrences of target defined in the **Find** area with a new fragment of text defined in **Replace with** area.
- Find all the occurrences of a word or string of characters (that can span over multiple lines) in the document you are editing. This operation also takes into account all the whitespaces contained in the fragment you are searching for. 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 after you click the **Find All** button.

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 macOS)** on your keyboard.
- Use the **Insert newline** contextual menu action.

You can use [Perl-like regular expressions syntax \(on page 287\)](#) 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** on your keyboard.

The *replace* operation can bind regular expression capturing groups (`$1`, `$2`, etc.) from the find pattern.

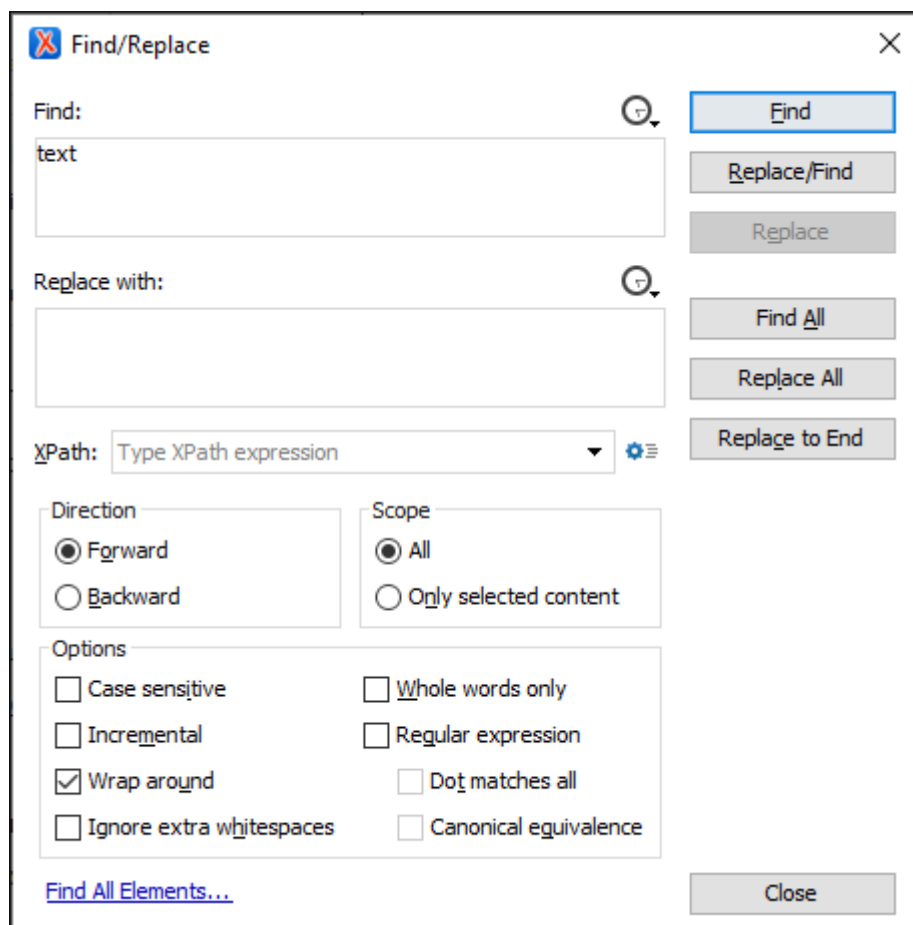


Tip:

To replace the `<tag-name>` start tag and its attributes with the `<new-tag-name>` tag use as **Find** the expression `<tag-name(\\s+)(.*)>` and as **Replace with** the expression `<new-tag-name$1$2>`.

Find/Replace Dialog Box

Figure 39. The Find/Replace Dialog Box



The **Find/Replace** dialog box contains the following options:

Find text area box

This is where you enter the 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** ([on page 276](#)). For example, to search for a space character you can use the `\u0020` code.

You can use the  **History** button to select from a list of the most recently used expressions. Use the  **Clear history** action from the bottom of the lists to remove these expressions.

Replace with text area box

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** ([on page 276](#)) 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.

You can use the  **History** button to select from a list of the most recently used 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. The cursor position does not affect the result of the XPath evaluation. The context of the XPath expression evaluation from the **Find/Replace** dialog box is the XML document root. The XPath is used for determining the intervals to be searched from the document, so the XPath result must be a node-set.

**Tip:**

You can use the [Content Completion Assistant \(on page 638\)](#) to help you input XPath expressions that are valid in the current context. See [Working with XPath Expressions \(on page 474\)](#) for more information and some common examples of how to write XPath expressions.

Clicking the  **XPath Options** button opens a preferences page where you can configure some XPath-related options.

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 content/lines.

Options section**Case sensitive**

When selected, the search operation follows the exact letter case of the text entered in the **Find** field.

Incremental

The search operation is started every time you type or delete a letter in the **Find** text box.

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.

Ignore extra whitespaces

If selected, the application normalizes the content (collapses any sequence of whitespace characters into a single space) and trims its leading and trailing whitespaces when performing the search operation. This is helpful when searching for spaced-separated words since line breaks and indentation between words will not affect the results. This option is automatically disabled if the **Regular expression** option is selected.

Whole words only

Only entire occurrences of a word are included in the search operation. This option is automatically disabled if the **Regular expression** option is selected.

Regular expression

When this option is selected, you can use regular expressions in [Perl-like regular expressions syntax \(on page 287\)](#) to look for specific pieces of text.

- **Dot matches all** - A dot used in a regular expression also matches end of line characters.
- **Canonical equivalence** - If selected, two characters will be considered a match if, and only if, their full *canonical (on page 637)* decompositions match. For example, the ã symbol can be inserted as a single character or as two characters (the a character followed by the tilde ~ character). This option is not selected by default.

Find button

Executes a find operation for the next occurrence of the target. It stops after highlighting the find match in the editor panel.

Replace/Find button

Executes a replace operation for the target followed by a find operation for the next occurrence.

Replace button

Executes a replace operation for the target without going to the next occurrence.

Find All button

Executes a find operation and displays all results in the **Results** view.

Replace All button

Executes a replace operation in the entire scope of the document.

Replace to End button

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/Replace in Multiple Files

The **Find/Replace in Files** feature enables you to define *Search for* or *Search for and Replace* operations across 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  **Find/Replace in Files** button on the main toolbar.
- The contextual menu of the **Project** view ([on page 251](#)).

The operation works on both local and remote files from an (S)FTP, WebDAV, or CMS server.

Find/Replace in Files Dialog Box

Figure 40. Find / Replace in Files Dialog Box (When Opened from the Toolbar Button)

The dialog box contains the following options:

Text to Find section

The first text field is where you enter the 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.

The rest of the options in this section can be used to refine your search:

Case sensitive

When selected, the search operation follows the exact letter case of the value entered in the **Text to find** field.

Whole words only

Only entire occurrences of a word are included in the search operation. This option is automatically disabled if either the **Ignore extra whitespaces** or **Regular expression** options are selected.

Ignore extra whitespaces

If selected, the application normalizes the content (collapses any sequence of whitespace characters into a single space) and trims its leading and trailing whitespaces when performing the search operation. This is helpful when searching for spaced-separated words since line breaks and indentation between words will not affect the results. This option is automatically disabled if the **Regular expression** option is selected.

Regular expression

When this option is selected, you can use regular expressions in [Perl-like regular expressions syntax \(on page 287\)](#) to look for specific pieces of text.

- **Dot matches all** - A dot used in a regular expression also matches end of line characters.
- **Canonical equivalence** - If selected, two characters will be considered a match if, and only if, their full [canonical \(on page 637\)](#) decompositions match. For example, the ã symbol can be inserted as a single character or as two characters (the a character followed by the tilde ~ character). This option is not selected by default.

Restrict to XPath

The XPath 2.0 expression you input in this combo is used for restricting the search scope. The XPath is used for determining the intervals to be searched from the document, so the XPath result must be a node-set.

Example: Use the XPath filter expression `//*[not(local-name() = 'uicontrol')]` to skip over the contents of any `<uicontrol>` element.



Tip:

You can use the [Content Completion Assistant \(on page 638\)](#) to help you input XPath expressions that are valid in the current context. See [Working with XPath Expressions \(on page 474\)](#) for more information and some common examples of how to write XPath expressions.

Clicking the  **XPath Options** button opens a preferences page where you can configure some XPath-related options.

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 is searched, skipping '<?...?>' (for example, <?processing instruction?>).
- **CDATA** - Searches inside content of CDATA sections.
- **DOCTYPE** - Searches inside content of DOCTYPE sections.
- **Entities** - Only the entity names are searched.

The two buttons **Select All** and **Deselect All** allow a simple activation and deactivation of all types of XML components.



Note:

Even if you select all options of the **Enable XML search options** section, the search is still XML-aware. If you want to perform the search over the entire file content, deselect **Enable XML search options**.

Replace with section

Use the text field in this section to specify a character string to replace the target with. It may contain regular expression group markers if the search expression is a regular expression and the **Regular expression** checkbox is selected.

Make backup files with extension

In the replace process Oxygen JSON Editor makes backup files of the modified files. The default extension is **.bak**, but you can change the extension as you prefer.

Scope section

The options available in this section depend on the context (how the dialog box was opened). Select one of the listed options to specify the scope for the operation. The possible options include:

Selected project resources

Searches only in the selected files.

Project files

Searches in all files from the current project.

All opened files

Searches in all files opened in Oxygen JSON Editor. You are prompted to save all modified files before any operation is performed.

Current file directory

The search is done in the directory of the file opened in the current editor panel. If there is no open file, this option is not available.

Opened archive (only available if opened from the Archive Browser view)

The search is done in an archive opened in the [Archive Browser \(on page 480\)](#) view.

Specified path

Use this option to specify the search path.

Filters section

The options available in this section depend on the context (how the dialog box was opened) and they can be used to filter the search operation. The possible options include:

Include files

Narrows the scope of the operation only to the files that match the given filters. For example, you can choose to filter the search to only include files with a certain file extension (such as **.xml*).

Recurse subdirectories

When selected, the search is performed recursively for the specified scope. The one exception is that this option is ignored if the scope is set to **All opened files**.

Include hidden files

When selected, the search is also performed in the hidden files.

Include archives

When selected, the search is also done in all individual file entries from all supported ZIP-type archives.

Show separate results for each search expression

When selected, the application opens a new tab to display the result of each new search expression. When the option is unchecked, the search results are displayed in the *Find in Files* tab, replacing any previous search results.

Always open selected results in Text mode

If selected, double-clicking results will always open the documents in **Text** mode (even if the particular document type is set to open in **Author** mode, by default). If not selected (default state), double-clicking results will open the documents in whatever editing mode is specified as the default for that document type. For example, by default, DITA documents will open in **Author** mode (as specified in the default framework configuration for DITA document types). Specialized XML documents such as XSLT or XML Schema will continue being opened in the **Text** editing mode.

Find All

Use the **Find All** button to execute the search operation. The results are displayed in a view that allows grouping the results as a tree with two levels.

Replace All

Use the **Replace All** button to execute the search operation and replace all occurrences with the specified string. When you replace a fragment of text, Oxygen JSON Editor offers an option to 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 the **Replace All** option with caution. Global searches may result in matching strings being replaced in instances that were not originally intended.



Note:

- You can use [Perl-like regular expression syntax \(on page 287\)](#) to match patterns in text content. The *replace* operation can bind regular expression capturing groups (\$1, \$2, etc.) from the find pattern.
- Exclusion patterns are accepted. For example, `*.java, !*Test.java` would search for all files with a `.java` extension, with the exception of any file whose name ends in `Test`.
- 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 JSON Editor uses by default the UTF-8 encoding for files of type XML, that is for files with one of the extensions: `.xml`, `.xsl`, `.fo`, `.xsd`, `.rng`, `.nvd1`, `.sch`, `.wsdl` or an extension associated with the XML editor type (on page 177). For the other files it uses the encoding configured for non-XML files (on page 114).



- You can cancel a long operation at any time by pressing the **Cancel** button of the progress dialog box, but doing so will not revert any replacements that have been processed up to that point.
- Since the content of read-only files cannot be modified, the **Replace** operation does not process those files. For every such file, a warning message is displayed in the message panel.

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 macOS**)) or from the shortcut **Find All Elements** that is available in the **Find / Replace** dialog box (on page 273). It assists you in defining XML element / attribute search operations in the current document.

Figure 41. Find All Elements Dialog Box

The dialog box can perform the following actions:

- Find all the elements with a specified name.
- Find all the elements that contain, or does not contain, a specified string in their text content.
- Find all the elements that have a specified attribute.
- Find all the elements that have an attribute with, or without, a specified value.

You can combine all of these search criteria to filter your results.

The following fields are available in the dialog box:

- **Element name** - The qualified name of the target element to search for. You can use the drop-down menu to find an element or enter it manually. It is populated with valid element names collected from the associated schema. To specify *any* element name, leave the field empty.

**Note:**

Use the qualified name of the element (`<namespace prefix>:<element name>`) when the document uses this element notation.

- **Element text** - The target element text to search for. The drop-down menu beside this field allows you to specify whether you are looking for an exact or partial match of the element text. For *any* element text, select **contains** from the drop-down menu and leave the field empty. If you leave the field empty but select **equals** from the drop-down menu, only elements with no text will be found. Select **not contains** to find all elements that do not include the specified text.
- **Attribute name** - The name of the attribute that must be present in the element. You can use the drop-down menu to select an attribute or enter it manually. It is populated with valid attribute names collected from the associated schema. For *any* or no attribute name, leave the field empty.

**Note:**

Use the qualified name of the attribute (`<namespace prefix>:<attribute name>`) when the document uses this attribute notation.

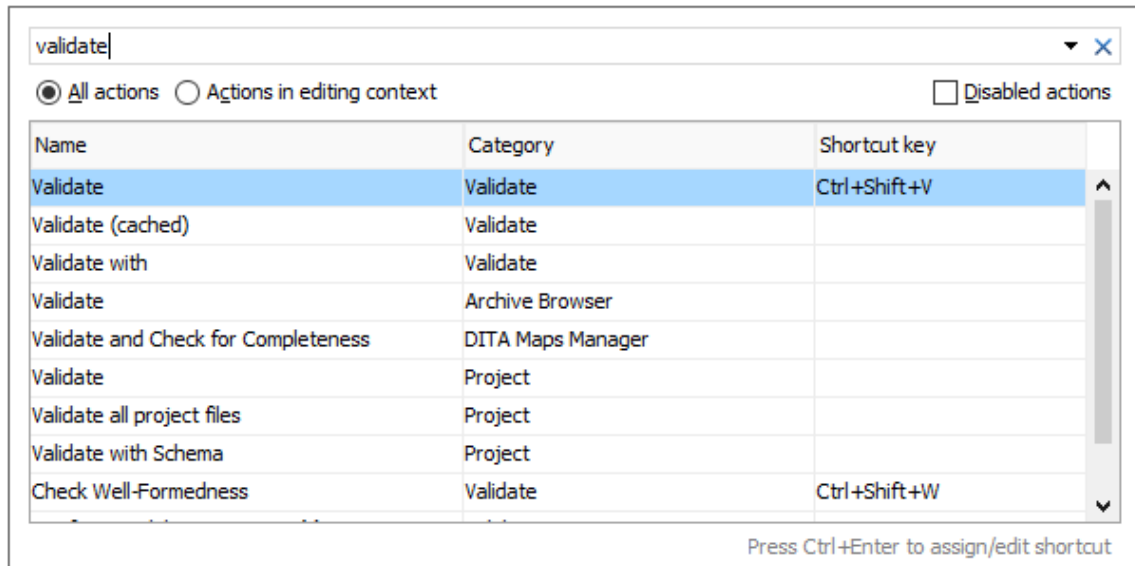
- **Attribute value** - The drop-down menu beside this field allows you to specify that you are looking for an exact or partial match of the attribute value. For *any* or no attribute value, select **contains** from the drop-down menu and leave the field empty. If you leave the field empty but select **equals** from the drop-down menu, only elements that have at least an attribute with an empty value will be found. Select **not contains** to find all elements that have attributes without a specified value.
- **Case sensitive** - When this option is selected, operations are case-sensitive.

When you select **Find All**, Oxygen JSON Editor tries to find the items that match all the search parameters. The results of the operation are presented as a list in the message panel.

Find and Invoke Actions

Oxygen JSON Editor includes a **Find action** feature that provides a quick way to find actions that are available throughout the application. You can also assign shortcuts for particular actions and invoke actions using this feature.

The **Find action** operation is available in the **Find** or **Help** menus and it opens a pop-up window where all the actions are presented in a sortable, filterable table.

Figure 42. Find Action Pop-Up Window

This pop-up window includes the following features, options, and controls:

Search Field

You can use the search field at the top to search for a specific action and it includes a history drop-down menu for quickly preforming recently-used search criteria. You can use the **X Delete** button to the right of the search field to clear the current text from the search field. You can also search for actions using certain keyboard shortcuts (excluding the common editing commands such as **Delete**, **Home**, **End**, **Delete**, **Ctrl+A**, **Ctrl+C**, **Ctrl+V**, etc.)

Filtering Options

All actions

Filters the table to display all available actions.

Actions in editing context

Filters the table to display available actions based on the current editing context where the application is focused (for example, if the current focus is a particular side-view, the table displays actions that are available in that side-view).

Disabled actions

Filters the table to also display actions that are currently disabled.

Double-Click

You can double-click an action in the table (or select an action and press **Enter**) to execute the particular action. Some actions will not have an effect if they are not allowed in the current editing context.

Accessibility Shortcuts

The following keyboard shortcuts can be used to enjoy this feature using only a keyboard:

- **Ctrl + Alt + K** - Opens the **Find action** pop-up window feature.
- **Up arrow / Down arrow** - Navigates the table vertically and switches from the search field to the table, and vice versa.
- **Tab / Shift + Tab** - Navigates between the radio filtering options and the checkbox option.
- **Left arrow / Right arrow** - Toggles the selection for the radio filtering options.
- **Space** - Toggles the checkbox option.
- **Enter** - Executes the selected action.
- **Ctrl + Enter** - Opens a dialog box where you can assign a keyboard shortcut for the selected action.
- **Ctrl + Up arrow / Ctrl + Down arrow** - Accesses the history drop-down when the search field is in focus.
- **Esc** - Closes the **Find action** pop-up window feature.

Actions Table

Displays the available actions based on the selected filtering options or search criteria. Some actions might be disabled/deactivated depending on the current editing context. When the **Disabled action** filtering option is selected, the disabled actions are displayed at the end of the results in the table.



Note:

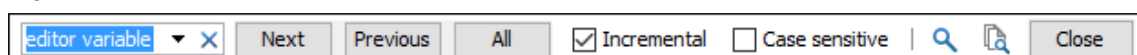
It is possible for certain actions not to be displayed in the actions table if they are created and implemented in other ways (for example, if they are implemented only to be available in a contextual menu).

Quick Find Toolbar

A reduced version of the **Find / Replace** dialog box (on page 273) is available as a **dockable toolbar** (on page 217). To display it, press the **Alt + Shift + F (Command + Option + F on macOS)** key combination or select the **Find > Quick Find** action. By default, the toolbar is displayed at the bottom of the Oxygen JSON Editor window, above the status bar, but can be changed at any time by dragging (and docking) it to a different location. To hide the toolbar, use the **Close** button.

All matches are highlighted in the current editor.

Figure 43. Quick Find Toolbar



The toolbar offers the following controls:

- **Search input box** - This is where you can insert the text you want to search for. The input box keeps a history of the last used search text. The background color of the input box turns red when no match is found.
- **Next** - Advances to the next match. You can also use the **Enter** key to jump forward to the next match.

- **Previous** - Jumps to the previous match. You can also use **Shift+Enter** to jump backward to the previous match.
- **All** - Highlights all matches of the search string in the current document.
- **Incremental** - If selected, the search operation is started every time you type or delete a character in the search input box.
- **Case sensitive** - If selected, the search operation follows the exact letter case of the search text.
-  **Find/Replace** - Opens the **Find/Replace** dialog box (*on page 273*).
-  **Find/Replace in Files** - Opens the **Find/Replace in Files** dialog box (*on page 277*).
- **Close** - Closes the **Quick Find** toolbar.

Keyboard Shortcuts for Finding the Next and Previous Match

Navigating from one match to the next or previous one is very easy to perform using the **F3** and **Shift + F3** (**Command + Shift + G on macOS**) keyboard shortcuts. They are useful for quickly repeating the last find action performed in the **Find / Replace** dialog box (*on page 273*), taking into account the same find options.



Restriction:

These shortcuts only take XPath expressions into account if the **Find / Replace** dialog box remains opened. Once you close it, the XPath expressions are no longer considered.

Regular Expressions Syntax

Oxygen JSON Editor uses the [Java regular expression syntax](#). It is **similar** to that used in Perl 5, with several exceptions. Thus, Oxygen JSON Editor does not support the following constructs:

- The conditional constructs `{X}` and `(condition)X|Y`.
- The embedded code constructs `{code}` and `??{code}`.
- The embedded comment syntax `##comment`.
- The preprocessing operations `\l`, `\u`, `\L`, and `\U`.

When using regular expressions, note that some sets of characters from [XPath/XML Schema/Schematron](#) are slightly different than the ones used by Oxygen JSON Editor/Java in the text searches from the **Find/Replace** dialog box (*on page 273*) and **Find/Replace in Files** dialog box (*on page 277*). The most common example is with the `\w` and `\W` set of characters. To ensure consistent results between the two, it is recommended that you use the following constructs in the **Find/Replace** dialog box (*on page 273*) and **Find/Replace in Files** dialog box (*on page 277*):

- `/w` - `[#x0000-#x10FFFF] - [\p{P}\p{Z}\p{C}]` instead of `\w`
- `/W` - `[\p{P}\p{Z}\p{C}]` instead of `\W`

There are some other notable differences that may cause unexpected results, including the following:

- In Perl, `\1` through `\9` are always interpreted as back references. A backslash-escaped number greater than 9 is treated as a back reference if at least that many sub-expressions exist. Otherwise, it is interpreted, if possible, as an octal escape. In this class octal escapes must always begin with a zero. In Java, `\1` through `\9` are always interpreted as back references, and a larger number is accepted as a back reference if at least that many sub-expressions exist at that point in the regular expression. Otherwise, the parser will drop digits until the number is smaller or equal to the existing number of groups or it is one digit.
- Perl uses the `g` flag to request a match that resumes where the last match left off.
- In Perl, embedded flags at the top level of an expression affect the whole expression. In Java, embedded flags always take effect at the point where they appear, whether they are at the top level or within a group. In the latter case, flags are restored at the end of the group just as in Perl.
- Perl is forgiving about malformed matching constructs, as in the expression `*a`, as well as dangling brackets, as in the expression `abc]`, and treats them as literals. This class also accepts dangling brackets but is strict about dangling meta-characters such as `+`, `?` and `*`.

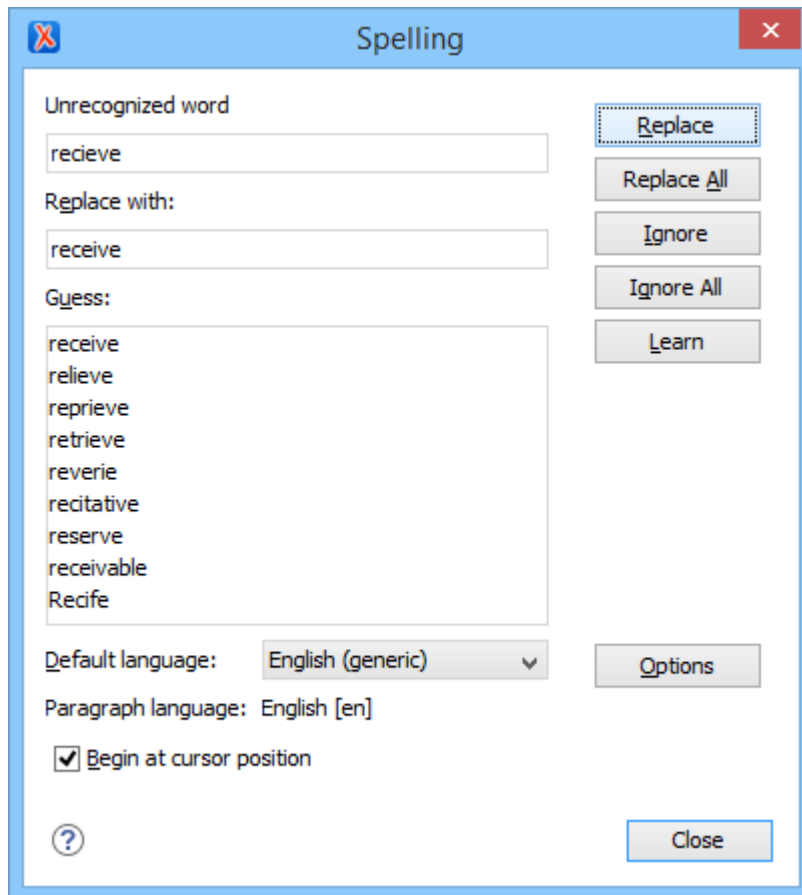
Related information

[Comparison between the Java and Perl 5 regular expression syntax](#)

Spell Checking

Oxygen JSON Editor includes an [automatic \(as-you-type\) spell checking feature \(on page 297\)](#), as well as a manual spell checking action to open a **Spelling** dialog box that offers a variety of options.

To manually check spelling in the current document, use the  **Check Spelling** action on the toolbar or from the **Edit** menu.

Figure 44. Check Spelling Dialog Box

The **Spelling** dialog box contains the following:

Unrecognized word

Displays the word that cannot be found in the selected dictionary. The word is also highlighted in the XML document.

Replace with

The character string that will replace the misspelled word.

Guess

Displays a list of suggested words to replace the unknown word. Double-click a word to automatically insert it in the document and resume the spell checking process.

Default language

Allows you to select the default language dictionary used by the spelling engine.

Paragraph language

In an XML document, you can mix content written in multiple languages. You can set the language code in the `@lang` or `@xml:lang` attribute for any particular section and Oxygen JSON Editor will automatically instruct the spell checker engine to apply the appropriate language dictionary for that section.

Begin at cursor position

Instructs the spell checker to begin checking the document starting from the current cursor position.

Action Buttons

Replace

Use this button to replace the unrecognized word with the selected word from the **Replace with** field.

Replace All

Use this button to replace all occurrences of the unrecognized word with the selected word from the **Replace with** field, starting from the cursor's position to the end of the document.



Note:

This action is case-sensitive.

Ignore

Ignores the first occurrence of the unrecognized word and allows you to continue checking the document. Oxygen JSON Editor skips the content of the XML elements [marked to be ignored \(on page 297\)](#).

Ignore All

Ignores all instances of the unrecognized word in the current document.

Learn

Adds the unrecognized word to the list of valid words.

Options

Opens the [Spell Check preferences page \(on page 156\)](#) where you can configure various options regarding the feature.

Spell Check Dictionaries and Term Lists

Oxygen JSON Editor uses the **Hunspell** engine for the spell checking feature. The Hunspell spell checking engine is open source and has an LGPL license. It is designed for languages with rich morphology and complex compounding or character encoding. Each language-country variant combination have their own specific dictionaries. Oxygen JSON Editor includes the following built-in dictionaries for the spell checker:

- English (US) [en_US]
- English (UK) [en_GB]
- French [fr]
- German [de_DE]
- Spanish [es_ES]

Other Hunspell Dictionaries

You can also download Hunspell dictionaries for other languages and add them to the Oxygen JSON Editor spell checker. An example of a website that includes numerous dictionary files is: <http://extensions.services.openoffice.org/dictionary>.

If you cannot find a Hunspell dictionary that is already built for your language, you can build the dictionary you need. To build a full Hunspell dictionary, follow [these instructions](#) and then add the dictionary to the Oxygen JSON Editor spell checker by following [this procedure \(on page 292\)](#).

Personalized Term Lists

Authoring in certain areas of expertise (for example, the pharmaceutical or automobile industries) might require the use of specific terms that are not part of the standard spell checker dictionary. To avoid marking these terms as errors, Oxygen JSON Editor provides a way of [adding personalized term lists \(on page 294\)](#) to the spell check engine. This involves creating a term list file that the spell checker will recognize and it is similar to the file Oxygen JSON Editor uses for storing [learned words \(on page 296\)](#).

The term list files are specific for each language and can be specific to each domain or area of expertise (for example, *legal*, *medical*, *automotive*). They can also be used to control forbidden words.

Related information

[Adding Custom Spell Check Dictionaries \(on page 292\)](#)

[Adding Custom Spell Check Term Lists \(on page 294\)](#)

[Building and Testing Hunspell Dictionaries](#)

Adding Custom Dictionaries and Term Lists

The Oxygen JSON Editor spell checker allows you to add customized Hunspell dictionaries and personalized term lists. The Hunspell dictionary mechanism requires a dictionary file (with a **.dic** file extension) and an affix file (with the **.aff** file extension). The personalized term lists are custom files (with the **.tdi** file extension) that you can create to include specialized terms or specify forbidden words in the Oxygen JSON Editor spell checker.

You can [add dictionaries \(on page 292\)](#) and [personalized term lists \(on page 294\)](#) to the default folder where they are stored or specify your own custom locations. You can view the default storage location in the [Spell Check Dictionaries preferences page \(on page 158\)](#) and the **Include dictionaries and term list from option (on page 159)** allows you to choose a custom storage location. All the dictionaries and term lists for a particular language that are found in either location are merged and used by the spell checker in Oxygen JSON Editor.

Related information

[Replacing a Spell Check Dictionary \(on page 295\)](#)

[Editing the Spell Checking Dictionaries](#)

Adding Custom Spell Check Dictionaries

There are three possible scenarios for adding Hunspell dictionaries to the Oxygen JSON Editor spell checker:

- You can download a pre-built Hunspell dictionary and add it to the spell checking mechanism.
- You can create a custom Hunspell dictionary file that defines your own list of words and add it to the spell checking mechanism.
- You can build your own full Hunspell dictionary and add it to the spell checking mechanism.

Download and Add a Pre-Built Hunspell Dictionary

To add a downloaded pre-built dictionary, follow these steps:

1. Download the files needed for your dictionary. You will need a *dictionary* file (with a **.dic** file extension) and an *affix* file (with the **.aff** file extension). If the dictionary does not include an affix file (**.aff**), you can create one and leave it empty, but it is needed for the mechanism to work properly. An example of a website that includes numerous dictionary files is: <http://extensions.services.openoffice.org/dictionary>.



Important:

The name of the files should begin with a two letter prefix for the language code, followed by an underscore or hyphen, then two letters that indicate the country code, followed by another underscore or hyphen, and then a descriptive name (for example, **en_US_medical.dic** for a medical dictionary in the US version of the English language, or for a less specific English medical dictionary, you could omit the country code like this: **en_medical.dic**). For a list of language codes, see https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.

2. Open the **Preferences** dialog box (**Options > Preferences**) (*on page 74*) and go to **Editor > Spell Check > Dictionaries** (*on page 158*).
3. Choose one of the following two options for adding the downloaded files.
 - a. Copy both files (**.dic** and **.aff**) to the default directory displayed in the **Dictionaries and term lists default folder** option (*on page 158*).
 - b. Copy both files (**.dic** and **.aff**) to any other directory, select the **Include dictionaries and term list from** option (*on page 159*), and select that directory. If you choose this option, make sure you read *this important note* (*on page 159*).
4. Restart the application for the spell checker to start using the new dictionary.

Create a Custom Hunspell Dictionary that Defines a List of Words

To create a custom Hunspell dictionary that defines your own list of words, follow these steps:

1. Create a *dictionary* file (with a **.dic** file extension) and an *affix* file (with the **.aff** file extension). The affix file (**.aff**) can be left empty, but it is needed for the mechanism to work properly.

**Important:**

The name of the files should begin with a two letter prefix for the language code, followed by an underscore or hyphen, then two letters that indicate the country code, followed by another underscore or hyphen, and then a descriptive name (for example, **en_US_medical.dic** for a medical dictionary in the US version of the English language, or for a less specific English medical dictionary, you could omit the country code like this: **en_medical.dic**). For a list of language codes, see https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.

2. In the dictionary file (**.dic** extension), add the words you want to be included in your custom dictionary. Add one word per row and the first line needs to contain the number of words, as in the following example:

```
2
parabola
asimptotic
```

**Tip:**

Words stored in dictionaries are not handled as case-sensitive. Therefore, you do not need to include both uppercase and lowercase versions of the words.

**Note:**

If you save the **.dic** file using UTF-8 encoding, then the corresponding **.aff** file should specify the encoding as a property inside it (if you do not specify the encoding, the default platform encoding will be used):

```
SET UTF-8
```

3. Open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Editor > Spell Check > Dictionaries** (on page 158).
4. Choose one of the following two options for saving the files.
 - a. Save both files (**.dic** and **.aff**) to the default directory displayed in the **Dictionaries and term lists default folder** option (on page 158).
 - b. Save both files (**.dic** and **.aff**) to any other directory, select the **Include dictionaries and term list from** option (on page 159), and select that directory. If you choose this option, make sure you read [this important note](#) (on page 159).
5. Restart the application for the spell checker to start using the new dictionary.

Build and Add a Full Hunspell Dictionary

To build and add a full Hunspell dictionary, follow these steps:

1. Create your Hunspell dictionary. For more information on how to do this, see: [Editing the Spell Checking Dictionaries](#).

Step Result: You should end up with a *dictionary* file (with a **.dic** file extension) and an *affix* file (with an **.aff** file extension). The affix file (**.aff**) can be empty, but it is needed for the mechanism to work properly.



Important:

The name of the files should begin with a two letter prefix for the language code, followed by an underscore or hyphen, then two letters that indicate the country code, followed by another underscore or hyphen, and then a descriptive name (for example, **en_US_medical.dic** for a medical dictionary in the US version of the English language, or for a less specific English medical dictionary, you could omit the country code like this: **en_medical.dic**). For a list of language codes, see https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.

2. Open the **Preferences** dialog box (**Options > Preferences**) (*on page 74*) and go to **Editor > Spell Check > Dictionaries** (*on page 158*).
3. Choose one of the following two options for saving the files.
 - a. Save both files (**.dic** and **.aff**) to the default directory displayed in the **Dictionaries and term lists default folder** option (*on page 158*).
 - b. Save both files (**.dic** and **.aff**) to any other directory, select the **Include dictionaries and term list from** option (*on page 159*), and select that directory. If you choose this option, make sure you read [this important note](#) (*on page 159*).
4. Restart the application for the spell checker to start using the new dictionary.

Related information

[Adding Custom Spell Check Term Lists](#) (*on page 294*)

[Editing the Spell Checking Dictionaries](#)

Adding Custom Spell Check Term Lists

You can create personalized term lists that are used to store specialized terms or control forbidden words. They can then be added to one of the directories that store the spell check dictionaries, and the spell checker will merge them with all the dictionaries and other term lists for a particular language.

Create and Add Personalized Term Lists

To create and add a personalized term list, follow these steps:

1. Create a *term list* file (with the **.tdi** file extension). The name of the file must begin with a two letter prefix that indicates the language it should be attached to, followed by an underscore or hyphen, and then a descriptive name (for example, **en_US_myterms.tdi** for term list in the US version of the English language or **en_myterms.tdi** for a less specific English term list). For a list of language codes, see https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.

2. In the term list file (`.tdi` extension), add the terms you want to be included in your custom dictionary. If you need to specify forbidden terms, those words simply need to be preceded by an asterisk. Add one word per row, as in the following example:

```
parabola
asimptotic
*hyperbola
```

**Note:**

Words stored in term lists are not handled as case-sensitive. Therefore, you do not need to include both uppercase and lowercase versions of the words.

3. Open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Editor > Spell Check > Dictionaries** (on page 158).
4. Choose one of the following two options for saving the file.
 - a. Save the file (`.tdi`) to the default directory displayed in the **Dictionaries and term lists default folder** option (on page 158).
 - b. Save the file (`.tdi`) to any other directory, select the **Include dictionaries and term list from** option (on page 159), and select that directory. If you choose this option, make sure you read [this important note](#) (on page 159).
5. Restart the application for the spell checker to start using the new term list.

Related information

[Adding Custom Spell Check Dictionaries](#) (on page 292)

Replacing a Spell Check Dictionary

There are several possible scenarios for replacing an existing Hunspell dictionary for the Oxygen JSON Editor spell checker:

- You can download a pre-built Hunspell dictionary and replace an existing dictionary with it.
- You can build your own full Hunspell dictionary and replace an existing dictionary with it.

Download a Pre-Built Hunspell Dictionary and Replace an Existing One

To replace an existing dictionary with a downloaded pre-built dictionary, follow these steps:

1. Download the files needed for your dictionary. You will need a *dictionary* file (with a `.dic` file extension) and an *affix* file (with the `.aff` file extension). If the dictionary does not include an affix file (`.aff`), you can create one and leave it empty, but it is needed for the mechanism to work properly. An example of a website that includes numerous dictionary files is: <http://extensions.services.openoffice.org/dictionary>.
2. Open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Editor > Spell Check > Dictionaries** (on page 158).
3. Choose one of the following two options to replace existing files.

- a. Replace the existing files (**.dic** and **.aff**) for the particular language in the default directory displayed in the **Dictionaries and term lists default folder** option (on page 158). Leave the **Include dictionaries and term list from** option deselected.
- b. Replace existing files (**.dic** and **.aff**) for the particular language in a directory specified in the **Include dictionaries and term list from** option (on page 159). If you choose this option, make sure you read [this important note](#) (on page 159).

**Important:**

Do not alter the naming convention. The name of the files must begin with a two letter prefix that indicates the language it should be attached to (for example, **en_US.dic** for a US English dictionary or **en.dic** for a less specific English dictionary). For a list of language codes, see https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.

4. Restart the application for the spell checker to start using the new dictionary.

Build a Full Hunspell Dictionary and Replace an Existing One

To replace an existing dictionary with a full Hunspell dictionary that you build, follow these steps:

1. Follow these instructions: [Building and Testing Hunspell Dictionaries](#).

Step Result: You should end up with a *dictionary* file (with a **.dic** file extension) and an *affix* file (with the **.aff** file extension). The affix file (**.aff**) can be empty, but it is needed for the mechanism to work properly.

2. Open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Editor > Spell Check > Dictionaries** (on page 158).
3. Choose one of the following two options to replace existing files.
 - a. Replace the existing files (**.dic** and **.aff**) for the particular language in the default directory displayed in the **Dictionaries and term lists default folder** option (on page 158). Leave the **Include dictionaries and term list from** option deselected.
 - b. Replace existing files (**.dic** and **.aff**) for the particular language in a directory specified in the **Include dictionaries and term list from** option (on page 159). If you choose this option, make sure you read [this important note](#) (on page 159).
4. Restart the application for the spell checker to start using the new dictionary.

Related information

[Adding Custom Dictionaries and Term Lists](#) (on page 291)

Learned Words

Spell checker engines rely on dictionaries to decide if a word is spelled correctly. To instruct the spell checker engine that an unknown word is actually correctly spelled, you need to add that word to a list of learned words. There are two ways to do this:

- Invoke the contextual menu on an unknown word, then select **Learn word**.
- Click the **Learn** button from the **Spelling dialog box** ([on page 288](#)) that is invoked by using the  **Check Spelling** action on the toolbar.

**Note:**

To delete items from the list of learned words, use the **Delete learned words** option in the [Editor > Spell Check > Dictionaries](#) preferences page ([on page 158](#)).

Related information

[Adding Custom Spell Check Term Lists](#) ([on page 294](#))

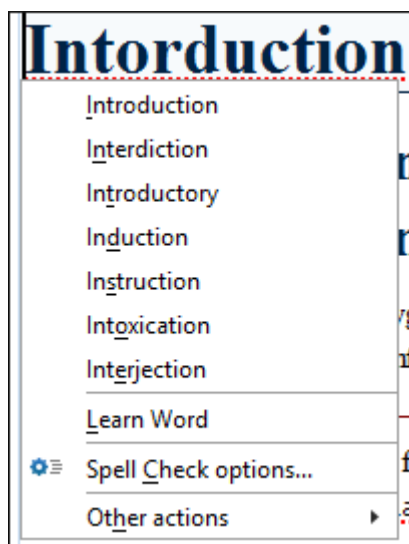
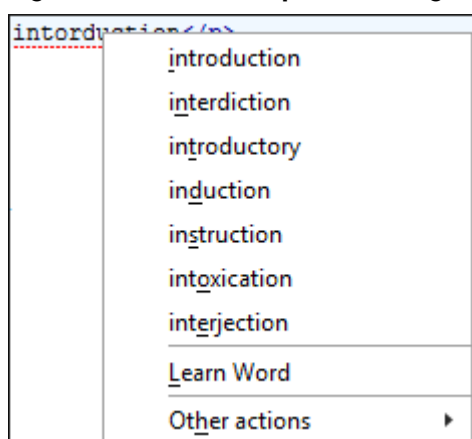
Ignored Words (Elements)

You may want the content of certain XML elements to always be skipped during the spell check process (for example, `<programlisting>`, `<codeblock>`, `<codeph>`, `<filepath>`, or `<screen>`). This can be done in one of several ways:

- You can skip through them manually, word by word, using the **Ignore** button in the **Spelling dialog box** ([on page 288](#)) that is invoked by using the  **Check Spelling** action on the toolbar.
- You can automatically skip the content of certain elements by maintaining a set of known element names that should never be checked. You can manage this set of element names by using the **Ignore elements** section ([on page 158](#)) in the **Spell Check** preferences page.

Automatic Spell Check

Oxygen JSON Editor 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](#) ([on page 156](#)). When the **Automatic Spell Check option** ([on page 156](#)) is selected, unknown words are underlined and some actions are available in the contextual menu to help you correct the word or prevent the word from being reported in the future.

Figure 45. Automatic Spell Checking in Author Mode**Figure 46. Automatic Spell Checking in Text Mode**

The contextual menu includes the following actions:

Delete Repeated Word

Allows you to delete words that were repeated in consecutive order.

List of Suggestions

A list of words suggested by the spell checking engine as possible replacements for the unknown word.

Learn Word

Allows you to add the current unknown word to the persistent dictionary of [learned words \(on page 296\)](#).

Spell check options (Available in Author mode only)

Opens the [Spell Check preferences page \(on page 156\)](#).

Other actions

This submenu give you access to all the usual contextual menu actions.

Related information[Learned Words \(on page 296\)](#)

Spell Check Multiple Files

The  **Check Spelling in Files** action allows you to check the spelling on multiple local or remote documents. This action is available in the following locations:

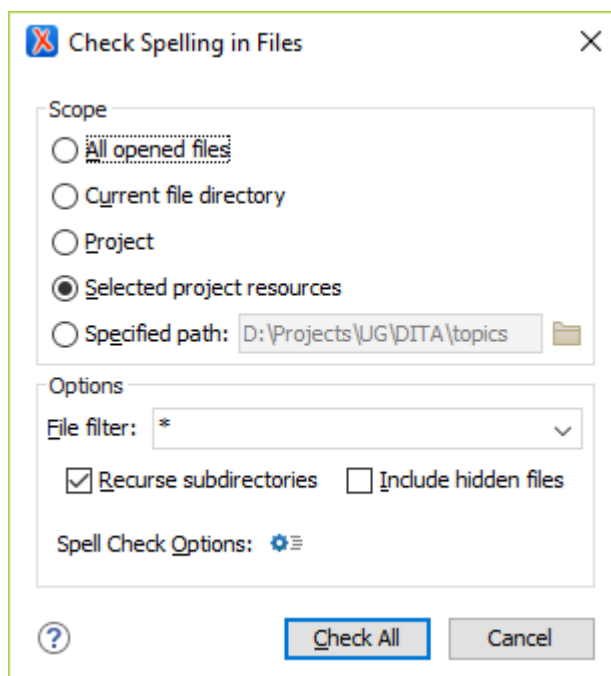
- The **Edit** menu.
- The contextual menu of the **Project view** ([on page 251](#)).

This action opens the **Check Spelling in Files** dialog box that allows you to define the scope and several other options. After you configure the settings for the operation, click the **Check All** button to check the spelling in all specified files. The spelling corrections are displayed in the **Results** view at the bottom of the editor and you can group the reported errors as a tree with two levels.

**Tip:**

If you want to instruct the spell checking engine to not report a particular word as being a spelling error in the future, use the **Learn Word(s)** action from the contextual menu in the **Results** view.

Figure 47. Check Spelling in Files Dialog Box (Invoked from Project View)



The following scopes are possible, depending on where the action was invoked:

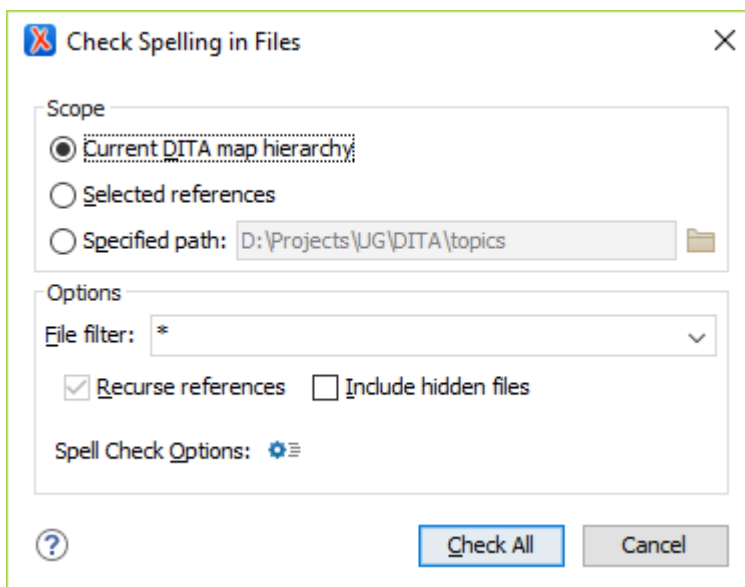
- **All opened files** - The spell check is performed in all open files.
- **Current file directory** - All the files in the folder of the currently edited file.
- **Current DITA map hierarchy** - Option available when the dialog is invoked from the **DITA Maps Manager** view. Checks the spelling in all references contained in the DITA map.

- **Project** - All files from the current project.
- **Selected project resources** - 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** - Allows you to filter the files from the selected scope.
- **Recurse subdirectories** - When selected, the spell check is performed recursively for the specified scope. The one exception is that this option is ignored if the scope is set to **All opened files**.
- **Include hidden files** - When selected, the spell check is also performed in the hidden files.
- **Spell Check Options** - The spell check processor uses the options available in the [Spell Check preferences page \(on page 156\)](#).

Figure 48. Check Spelling in Files Dialog Box (Invoked from the DITA Maps Manager View)



AutoCorrect Misspelled Words

Oxygen JSON Editor includes an *AutoCorrect* feature to automatically correct misspelled words, as well as to insert certain symbols or other text, as you type in **Author** mode. Oxygen JSON Editor includes a default list of commonly misspelled words and symbols, but you can modify the list to suit your needs. You can also choose to have the *AutoCorrect* feature use suggestions from the main spell checker. The suggestions will only be used if the misspelled words are not found in the [Replacements Table \(on page 129\)](#).

When enabled, the *AutoCorrect* feature can be used to do the following:

- Automatically correct misspelled words while you edit in **Author** mode. The actual operation of replacing a word is triggered by a space, dash, or certain punctuation characters (, . ; : ? ! ' ")] }).
- Easily insert symbols. For example, if you want to insert a ® character, you would type (R).
- Quickly insert text fragments.
- Quickly insert XML fragments. For example, if you enter a hyphen (-) in an empty paragraph followed by a space, it will automatically be converted to a list with a list item.

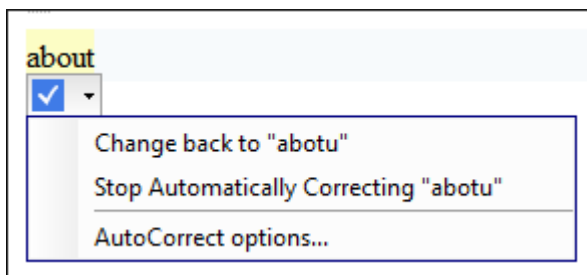
AutoCorrect is enabled by default. To configure this feature, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Edit Modes > Author > AutoCorrect**.

AutoCorrect Drop-down Actions

After the automatic operation of replacing a misspelled word (triggered by a space, dash, or certain punctuation characters), the affected string is highlighted. The highlight is removed upon the next editing action (text insertion or deletion). If you hover over the highlight, a small widget appears below the word. If you hover over the widget, it expands and you can click it to present a drop-down list that includes the following actions:

- **Change back to "[original word]"** - Reverts the correction back to its original form.
- **Stop Automatically Correcting "[original word]"** - This option is presented if the correction is performed based on the [AutoCorrect Replacements Table \(on page 129\)](#) and selecting it will delete the corresponding entry from the Replacements Table.
- **Learn Word "[original word]"** - This option is presented if the [Use additional suggestions from the spell checker option \(on page 128\)](#) is selected in the **AutoCorrect** preferences page [\(on page 128\)](#) and the correction is performed based on the *Spell Checker*. Selecting this option will add the item to the list of [learned words \(on page 296\)](#).
- **AutoCorrect options** - Opens the [AutoCorrect preferences page \(on page 128\)](#) that allows you to configure the feature.

Figure 49. AutoCorrect Drop-down Actions



AutoCorrect Case-Sensitivity

The *AutoCorrect* feature results in the following types of substitutions regarding case-sensitivity:

- Words with all lower-case characters will be replaced with lower-case substitutions (for example, "abotu" is replaced with "about").
- Words with irregular-case characters will be replaced with lower-case substitutions ("ABotU" is replaced with "about").
- Words with all upper-case characters will be replaced with upper-case substitutions ("ABOTU" is replaced with "ABOUT").
- Words starting with an upper-case character will be replaced with substitutions having the same pattern ("Abotu" is replaced with "About").

**Note:**

The *AutoCorrect* feature also uses the list of **ignored elements from the Spell Check preferences page (on page 158)**. All elements (along with their descendant elements) included in this list will be ignored by the *AutoCorrect* engine.

Related information

[Spell Checking \(on page 288\)](#)

Add Dictionaries for the AutoCorrect Feature

To add new dictionaries for the *AutoCorrect* mechanism (on page 300), or to replace an existing one, follow these steps:

1. Download an *AutoCorrect* dictionary file for the desired language. The file needs to have a **.dat** file extension. An example of a website that includes some *AutoCorrect* dictionary files is: [OpenOffice Extensions Search Page](#).
2. Open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Editor > Edit Modes > Author > AutoCorrect > Dictionaries** (on page 130).
3. Choose one of the following two options for adding the downloaded files:
 - a. Copy the downloaded **.dat** file to the default directory displayed in the **Dictionaries default folder** option. (on page 130). Note that if you are replacing an existing dictionary file, this is the best option.
 - b. Copy the downloaded **.dat** file to any other directory, select the **Include dictionaries from** option (on page 130), and select that directory. If you choose this option, make sure you read [this important note \(on page 130\)](#).
4. Restart the application for the *AutoCorrect* mechanism to start using the new dictionary.

Working with Special Characters and Encoding

While regular characters make up the English and European alphabets and the corresponding basic set of figures and symbols, there are many other *special characters* that belong to various other language representations, such as Arabic, Indian, Japanese, Chinese, or Korean. Oxygen JSON Editor provides support for special characters in various ways:

Opening and Saving Documents

The [Unicode standard](#) provides support for all the character symbols in all known languages and Oxygen JSON Editor [provides support for all Unicode characters \(on page 303\)](#). There are various encoding options and features to help determine how to handle [documents with unsupported characters \(on page 304\)](#).

Fonts

Oxygen JSON Editor provides the ability to [choose the fonts to be used in the various editing modes \(on page 83\)](#). In some cases, changing the font may be a solution when special characters are not rendered as expected.

For special characters that are not included in any of the default fonts, Oxygen JSON Editor tries to find that symbol in a [fallback font \(on page 305\)](#).



Tip:

For documents written in languages that use special characters (such as 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. To change the font in Oxygen JSON Editor, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#), go to **Appearance > Fonts**. You can select a font for each editing mode in this preferences page.

Navigation and Layout

Oxygen JSON Editor supports bidirectional text, such as Arabic, Hebrew, and certain Asian languages, or other special characters that are combined into a single glyph.

Unicode Support

Unicode is a standard for providing consistent encoding, representation, and handling of text. There is a unique Unicode number for every character, independent of the platform and language. Unicode is internationally recognized and is required by modern standards (such as XML, Java, JavaScript, LDAP, CORBA 3.0, WML, etc.).

Oxygen JSON Editor provides support for the Unicode standard, enabling your XML application to be targeted across multiple platforms, languages, and countries without re-engineering. Internally, the Oxygen JSON Editor uses 16-bit characters covering the Unicode Character set.

As a Java application, Oxygen JSON Editor includes a default Java input method for typing characters with Unicode codes. However, the default input method does not cover all the Unicode codes (for example, the codes for some accented characters or characters found in East Asian languages). Such characters can be inserted in the editor panel of Oxygen JSON Editor either with [the Character Map dialog box \(on page 307\)](#) available from **Edit > Insert from Character Map** or by installing a Java input method that supports the insertion of the needed characters. The [installation of a Java input method](#) depends on the platform (Windows, macOS, Linux, etc.) and is the same for any Java application.



Note:

Oxygen JSON Editor may not be able to display characters that are not supported by the operating system (either not installed or unavailable).

**Tip:**

On windows, you can enable the support for **CJK** (Chinese, Japanese, Korean) languages from **Control Panel / Regional and Language Options / Languages / Install files for East Asian languages**.

Related information

[Unicode Fallback Font Support \(on page 305\)](#)

[Inserting Special Characters with the Character Map \(on page 306\)](#)

Opening and Saving Documents with Unsupported Characters

When loading documents, Oxygen JSON Editor reads the document prolog to determine the specified encoding type. This encoding is then used to instruct the Java Encoder to load support for and to save the document using the specified code chart. When the encoding type cannot be determined, Oxygen JSON Editor displays the **Available Java Encodings** dialog box that provides a list of all encodings supported by the Java platform.

Opening Documents with Unsupported Characters

When opening a document in Oxygen JSON Editor, if it contains characters that are not supported by the specified encoding standard (these unrecognized characters are rendered as an empty box) , the application determines how to handle them based upon the setting specified in the **Encoding Errors Handling** option in the **Encoding preferences page (on page 115)**. The default setting is **REPORT**, which means an error message is displayed for characters that cannot be represented in the specified encoding. If the option is set to **REPLACE**, the character is replaced with a standard replacement character for the particular encoding. If the option is set to **IGNORE**, the error is ignored and the character is not rendered.

Saving Documents with Unsupported Characters

When saving a document edited in the **Text**, **Grid**, or **Design** modes, if it contains characters that are not supported by the encoding declared in the document prolog, Oxygen JSON Editor displays a notification that you need to resolve the conflict before saving the document.

When saving a document edited in the **Author** mode, all characters that fall outside the detected encoding will be automatically converted to hexadecimal character entities.

When saving a document with UTF-16 encoding, 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 one created on a macOS platform (where the byte order mark is *UnicodeBig*). The byte order and the BOM of an existing document are preserved when the document is edited and saved. This behavior can be changed in Oxygen JSON Editor from the **Encoding preferences page (on page 114)**.

Unicode Fallback Font Support

Oxygen JSON Editor provides fonts for most common Unicode ranges. However, if you use [special symbols or characters \(on page 306\)](#) 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 \(on page 303\)](#) as possible. When a display system encounters a character that is not part of the range of any of the available fonts, Oxygen JSON Editor will try to find that symbol in a *fallback* font.

Example of a Scenario Where a Fallback Font is Needed

Suppose that you need to insert the wheelchair symbol (♿ - U+267F) into your content in a Windows operating system. By default, Oxygen JSON Editor does not render this symbol correctly since it is not included in any of the default fonts. It is included in **Segoe UI Symbol**, but this font is not part of the default fonts that come with Oxygen JSON Editor. To allow Oxygen JSON Editor to recognize and render the symbol correctly, you can add **Segoe UI Symbol** as a *fallback* font.

Adding a Fallback Font in Windows (7 or Later)

To add a fallback font to the Oxygen JSON Editor installation, use the following procedure:

1. Start Windows Explorer and browse to the `[OXYGEN_INSTALL_DIR]/jre/lib/fonts` directory.
2. Create a directory called `fallback` (if it is not already there).
3. Copy a font file (True Type Font - TTF) that includes the special characters into this directory.



Tip:

You could, for example, copy the *Segoe UI Symbol Regular* font from `C:\Windows\Fonts`.

4. Restart Oxygen JSON Editor for the changes to take full effect.

Result: Whenever Oxygen JSON Editor finds a character that cannot be rendered using its standard fonts, it will look for the glyph in the fonts stored in the `fallback` folder.

Adding a Fallback Font in Other Platforms

For macOS 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 JSON Editor, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#), go to **Appearance > Fonts**.
4. Click the **Choose** button for the particular editing mode (**Editor** for **Text** mode) and select your custom font from the drop-down list in the subsequent dialog box.
5. Restart Oxygen JSON Editor for the font changes to take full effect.

Related information[Unicode Support \(on page 303\)](#)[Inserting Special Characters with the Character Map \(on page 306\)](#)

Inserting Special Characters with the Character Map

Oxygen JSON Editor includes a **Character Map** for inserting special characters. It can also be used to find the decimal, hexadecimal, or *character entity* equivalent for a particular character or symbol.

Inserting Special Characters

To insert a special character at the current location within a document, follow these steps:

1. Open the **Character Map** dialog box ([on page 307](#)) by selecting **More symbols** from the  **Symbols** drop-down menu on the toolbar (if this button is not displayed, right-click in the toolbar area, select **Configure Toolbars** and [chosen to display the Symbols toolbar \(on page 222\)](#)).
2. Find the symbol you want to insert and double-click it (or select it and click **Insert**).

**Tip:**

The most recently used characters and some of the most common characters are listed when you click the  **Symbols** drop-down button so you can easily insert any of those characters by simply selecting it from the drop-down.

Finding the Decimal, Hexadecimal, or Character Entity Equivalent

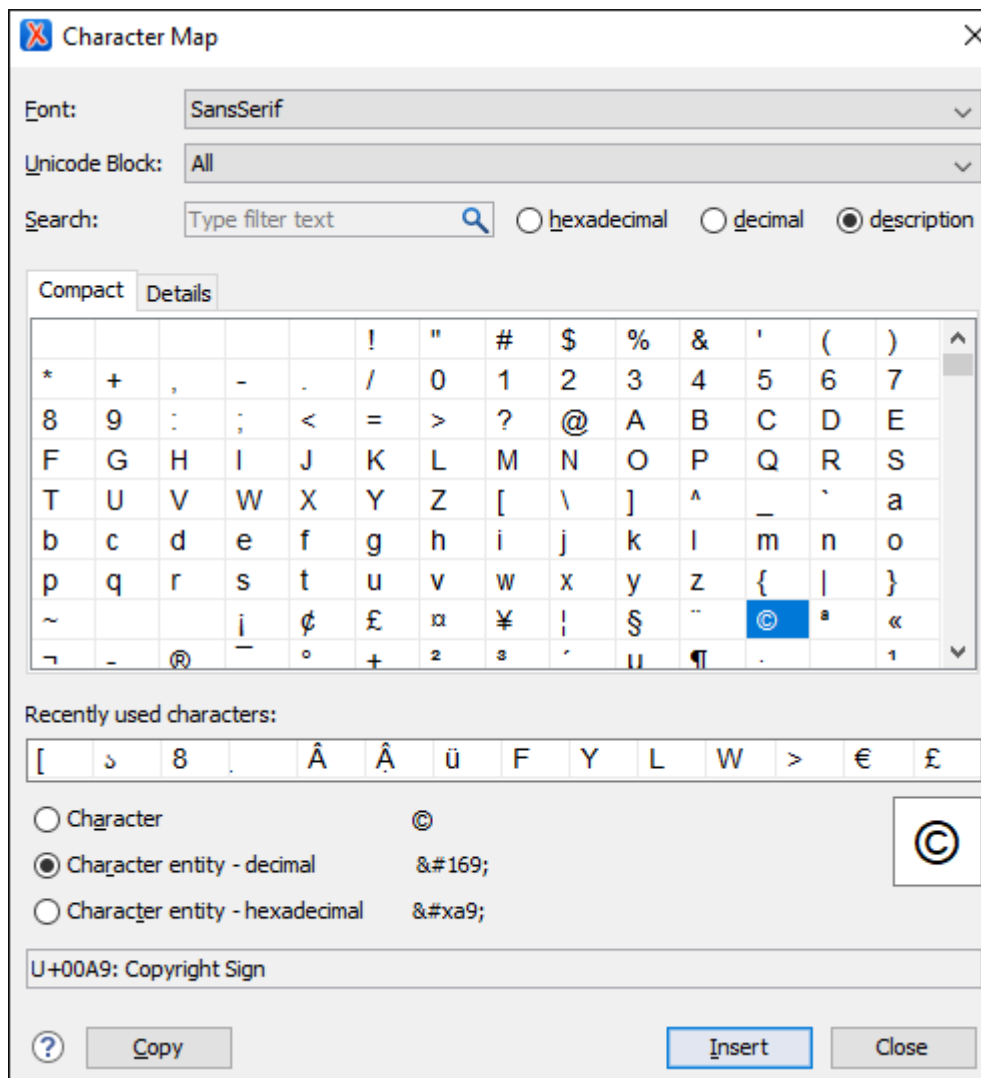
You can see the hexadecimal value for any character that is already inserted in your document by placing the cursor right after the character and you can see its value in the status bar at the bottom of the application.

For other characters, or to find the decimal equivalent, or even the *character entity* equivalent, following these steps:

1. Open the **Character Map** dialog box ([on page 307](#)) by selecting **More symbols** from the  **Symbols** drop-down menu on the toolbar (if this button is not displayed, right-click in the toolbar area, select **Configure Toolbars** and [chosen to display the Symbols toolbar \(on page 222\)](#)).
2. Find the symbol and select it. You can use the filters and the **Search** field at the top of the dialog box to narrow the search.
3. Click the **Details** tab on top of the preview window to see the decimal, hexadecimal, and description of the character. The *character entity* equivalent (both its decimal and hexadecimal values) are displayed at the bottom of the dialog box.

Character Map Dialog Box

Figure 50. 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 that will have characters displayed.

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 ([on page 308](#)). If you enter a character description in the **Search** field, the **description** is selected automatically.

Character Table Section

The characters that are available to be inserted are listed in two tabs:

- **Compact** - Matrix-like table that displays a visual representation of the characters.
- **Details** - Displays the available characters in a tabular format, presenting their decimal and hexadecimal value along with their description.

Recently Used Characters Section

Displays the symbols that you have used recently and you can also select one from there to insert it in the current document.

Character Mode Section

The next section of the dialog box allows you to select how you want the character to appear in the **Text** editing mode. You can choose between the following:

- **Character**
- **Character entity - decimal**
- **Character entity - hexadecimal**

You can see the character or code that will be inserted in **Text** mode next to the selections in this section and a box on the right side of the dialog box allows you to see the character that will be inserted in **Author** mode. You can also see the name and range name of a character either at the bottom of the dialog box, or in a tooltip when hovering the cursor over the character.

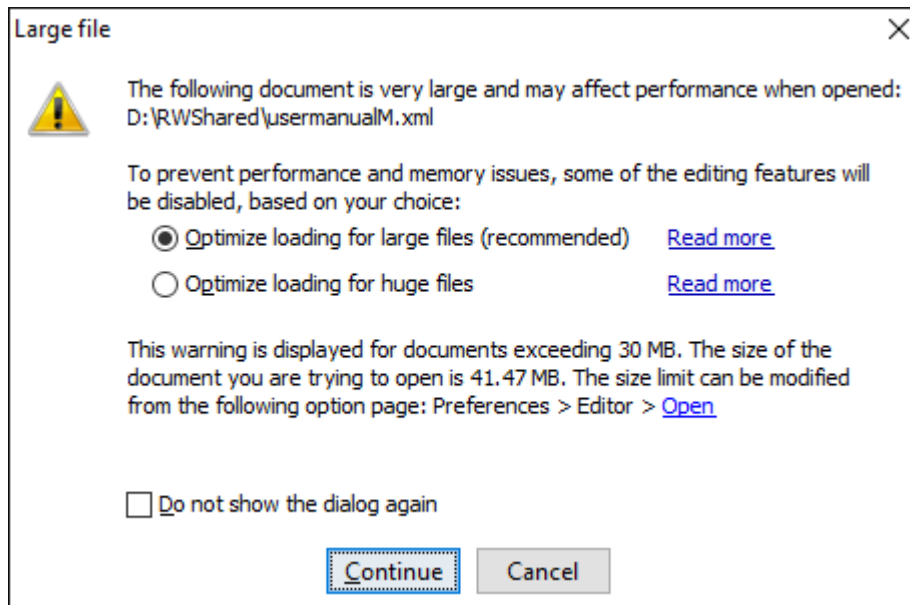
Click the **Insert** button to insert the selected character in the current editor at the cursor position. You will see the character in the editor if [the editor font \(on page 83\)](#) is able to render it. The **Copy** button copies it to the clipboard without inserting it in the editor.

Related information

[Working with Special Characters and Encoding \(on page 302\)](#)

Loading Large Documents

When you open a document with a file size larger than the limit configured in [Open preferences \(on page 133\)](#), Oxygen JSON Editor prompts you to choose whether you want to optimize the loading of the document for large files or for huge files.

Figure 51. Large File Prompt Dialog Box

If your file has a size smaller than 300 MB, the recommended approach is **Optimize loading for large files** (*on page 309*). For documents that exceed 300 MB, the recommended approach is **Optimize loading for huge files** (*on page 310*).

Optimize Loading for Large Files

If you open a document that exceeds the limit configured in **Open preferences** (*on page 133*) (the default limit is 30 MB), a **dialog box will be displayed** (*on page 308*) prompting you to choose whether you want to optimize the loading of the document for large files or for huge files. If you choose the **Optimize loading for large files** option (typically recommended for files smaller than 300 MB), a special memory optimization is implemented so that the total memory allocated for the application is not exceeded. A temporary buffer file is created on disk and the available free disk space needs to be at least double the size of the file you want to open.

When opening a large file in this optimized editing environment, some editing features are disabled, including:

- The file can only be opened in **Text** mode.
- The automatic validation is not available.
- The XPath filter is disabled in the **Find/Replace dialog box** (*on page 273*).
- The bidirectional Unicode support (right-to-left writing) is disabled.
- The **Format and indent the document on open** option (*on page 135*) is automatically deselected for non-XML documents. For XML documents, the formatting is done while optimizing the memory usage by ignoring the options set in the **Format preferences page** (*on page 135*).
- Localizations for the results of an XPath expression will be less precise.

Related information

[Optimize Loading for Huge Files](#) (*on page 310*)

Optimize Loading for Huge Files

If you open a document that exceeds the limit configured in [Open preferences \(on page 133\)](#) (the default limit is 30 MB), a [dialog box will be displayed \(on page 308\)](#) prompting you to choose whether you want to optimize the loading of the document for large files or for huge files. If you choose the **Optimize loading for huge files** option (typically recommended for files larger than 300 MB), the file is split in multiple pages (each approximately 1MB in size). Each page is individually loaded (and edited) in **Text** mode by using a special horizontal slider located at the top of the editing area.

Figure 52. Huge File Editor Horizontal Slider



When opening a file in this special huge file editor, some editing features are disabled, including:

- For XML files, the UTF-8, UTF-16, ASCII, Windows-1252, and ISO 8859-1 encodings are supported. No other encoding is supported.
- The file can only be opened in **Text** editing mode.
- The automatic validation is disabled.
- The XPath filter is disabled in the [Find/Replace dialog box \(on page 273\)](#).
- The bidirectional Unicode support (right-to-left writing) is disabled.
- The **Format and indent the document on open** option [\(on page 135\)](#) is automatically deselected for non-XML documents. For XML documents, the formatting uses less memory by ignoring the options set in [the Format preferences page \(on page 135\)](#).
- The **Outline** view is not supported.
- The file content is soft wrapped by default.
- The [Find/Replace dialog box \(on page 273\)](#) only supports the **Find** action.
- Saving changes is only possible if the **Safe save option (on page 134)** (in the **Save** preferences page) is enabled.
- The **undo** operation is not available if you go to other pages and come back to the modified page.

Related information

[Optimize Loading for Large Files \(on page 309\)](#)

Documents with Long Lines

When working with documents that contain lines of text that exceed the boundaries of your monitor, you might want to see the text wrapped. To do so, use one of the following methods:

- Press **Ctrl + Shift + Y (Command + Shift + Y on macOS)** to toggle the line wrap feature for the current document only.
- Select the [Line wrap \(on page 118\)](#) option in the **Text** preferences page to apply the line wrap to all documents.

Features that Might be Affected by Wrapping Lines of Text

Documents that contain thousands of characters per line can affect the performance of Oxygen JSON Editor **Text** mode. When a certain line length limit is reached (controlled from the [Optimize loading for documents with lines longer than \(Characters\) \(on page 133\)](#) option), Oxygen JSON Editor prompts you to wrap the lines of text. By doing so, the following features may be affected to maintain a reasonable level of productivity:

- The editor uses the `Monospaced` font.
- You cannot set font styles.
- Automatic validation is disabled.
- [Automatic spell checking \(on page 297\)](#) is disabled.
- When editing XML documents, the **XPath** field is disabled in the [Find/Replace dialog box \(on page 273\)](#).
- Less precise localization for executed XPath expressions in XML documents. The XPath executions use SAX sources for a smaller memory footprint. It is recommended to use 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.

Handling Read-Only Files

If a file marked as read-only is opened in Oxygen JSON Editor you can by default perform modifications to it. This behavior is controlled by the [Can edit read only files option \(on page 116\)](#). 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 \(on page 246\)](#). 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.

Scratch Buffer

The **Scratch Buffer** view can be used for storing fragments of arbitrary text during the editing process. It can be used to drop bits of paragraphs (including arbitrary XML markup fragments) while rearranging and editing the document and also to drag and drop fragments of text from the Scratch Buffer to the editor panel. The **Scratch Buffer** is basically a text area offering XML syntax highlight. The view's contextual menu contains basic edit actions such as **Cut**, **Copy**, and **Paste**.

If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

Compare Files or Directories

Oxygen JSON Editor provides a simple means of performing file and folder comparisons. You can see the differences in your files and folders and merge the changes. You can also use the file comparison to compare fragments or files inside zip-based archives.

There are two types of comparison tools: **Compare Directories** or **Compare Files**. These utilities are available from the **Tools** menu or can be opened as stand-alone applications from the Oxygen JSON Editor installation folder (`diffDirs.exe` and `diffFiles.exe`).

Starting the Tools from a Command Line

The comparison tools can also be started by using command-line arguments. In the installation folder there are two executable shells (`diffFiles.bat` and `diffDirs.bat` on Windows, `diffFiles.sh` and `diffDirs.sh` on macOS and Linux). To specify files or directories to compare, you can pass command-line arguments to each of these shells. The arguments can point to file or folder paths in directories or archives (supported formats: *zip*, *docx*, and *xlsx*).

Directory Comparison Example

To start a [comparison between the two directories \(on page 332\)](#), use the following construct:

```
diffDirs.bat/diffDirs.sh [directory path 1] [directory path 2]
```

If you pass only one argument, you are prompted to manually choose the second directory or archive.

For example, to start a comparison between two Windows directories, the command line would look like this:

```
diffDirs.bat "c:\documents new" "c:\documents old"
```

**Tip:**

If there are spaces in the path names, surround the paths with quotes.

File Comparison Example

To start a [comparison between 2 or 3 files \(on page 313\)](#), use the following construct: `diffFiles.bat/diffFiles.sh [path to left file] [path to right file] [path to base file]`.

If three files are specified, the tool will start in the [3-way comparison mode \(on page 316\)](#). If only two files are specified, the tool will start in the [2-way comparison mode \(on page 313\)](#). The first specified file will be added to the left panel in the comparison tool, the second file to the right panel, and the optional third file will be the base (ancestor) file used for a 3-way comparison. If you pass only one argument, you are prompted to manually choose another file.

For example, to do a 3-way comparison on Windows, the command line would look like this:

```
diffFiles.bat "c:\docs\file 1" "c:\docs\file 2" c:\docs\basefile
```

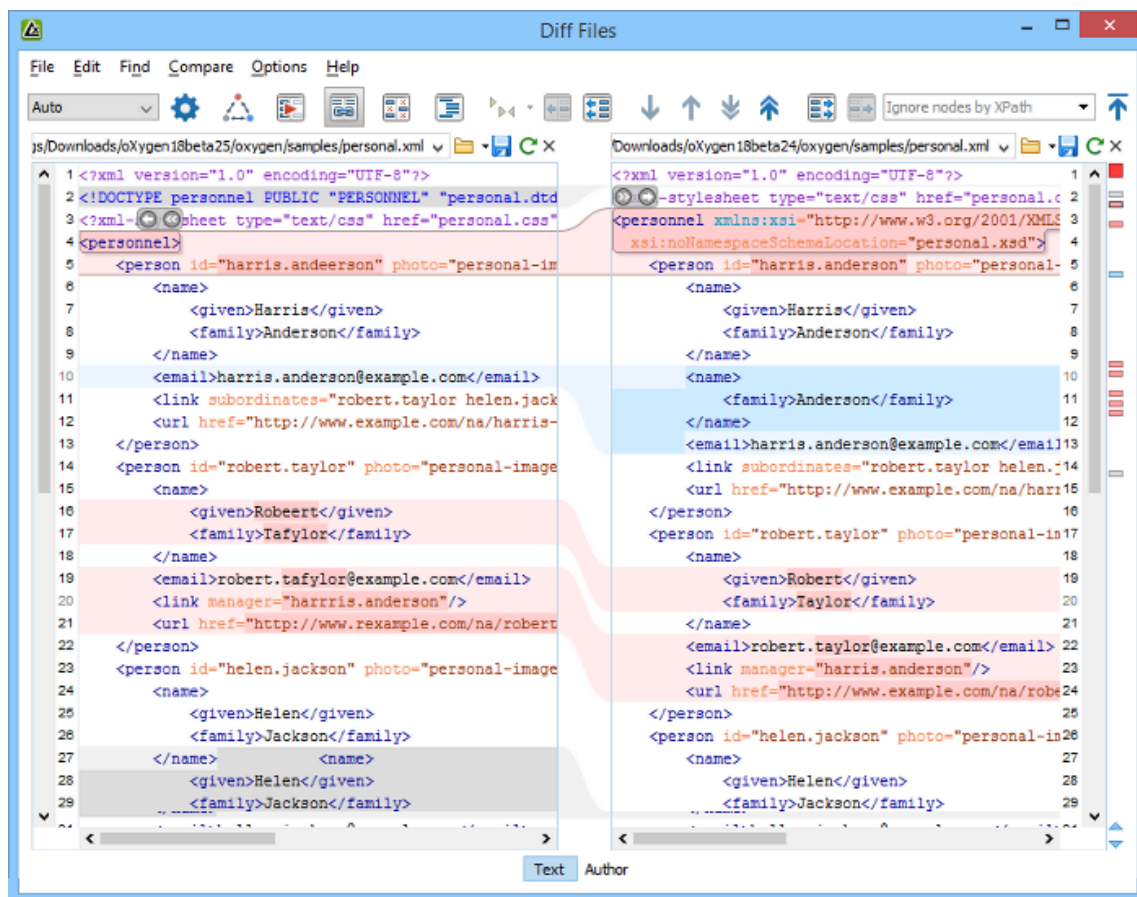
**Tip:**

If there are spaces in the path names, surround the paths with quotes.

Compare Files Tool

The built-in **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 JSON Editor installation folder (`diffFiles.exe`).

Figure 53. 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 actions in the  **Browse** drop-down menu.

Step Result: The selected files are opened in the two side-by-side editors. A text editing mode is used to offer a better view of the differences.

2. To highlight the differences between the two files, click the  **Perform File Differencing** button from the toolbar.
3. You can use the drop-down menu on the left side of the toolbar to change the [algorithm \(on page 315\)](#) 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 54. Two-Way Differences

1	<?xml version="1.0" encoding="UTF-8"?>		<?xml version="1.0" encoding="UTF-8"?>	1
2	<!DOCTYPE personnel PUBLIC "PERSONNEL"		<?xml-stylesheet type="text/css" href="	2
3	<?xml-stylesheet type="text/css" href="		<personnel xmlns="http://www.oxygenxm	3
4	<personnel>		<person id="robert.taylor" photo=	4

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 \(on page 168\)](#), 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.

Comparing Fragments (Copy/Paste)

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 macOS))** button, before pasting the fragments. Other notes for pasting fragments:

- As long as the fragment is more than 10 characters, the application will attempt to automatically detect the content type. It can detect the following types: XML, DTD, CSS, JSON, and Markdown (if it starts with #). If one of those content types is detected, the fragments will be displayed with syntax highlights.
- 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 \(on page 324\)](#) and in the [various menus \(on page 327\)](#) (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 JSON Editor offers the following two-way diff algorithms to compare files or fragments:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Characters** - Computes the differences at character level, meaning that it compares two files or fragments looking for identical characters. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **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.

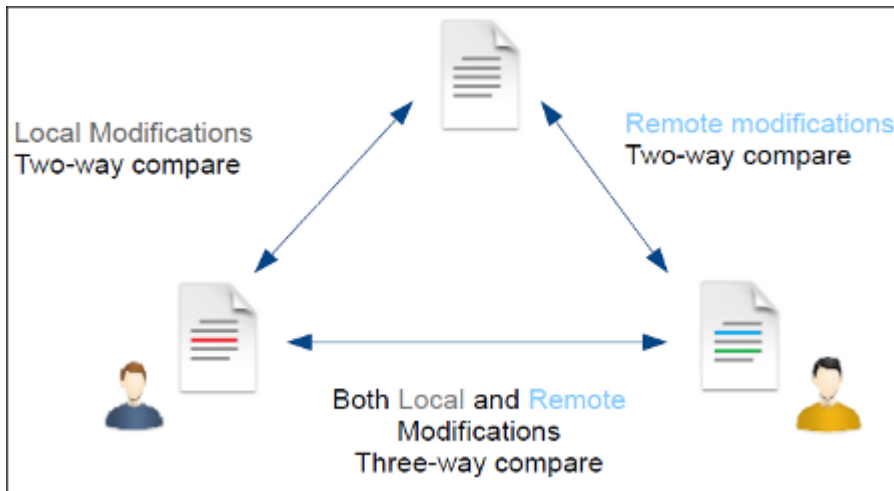
Three-Way Comparisons

Oxygen JSON Editor also includes a three-way comparison feature to help you solve conflicts and merge changes between multiple modifications. It is especially helpful for teams who have multiple authors editing

and committing the same documents. It provides a comparison between a local change, another change, and the original base revision. Some additional advantages include:

- Visualize and merge content that was modified by you and another member of your team.
- Marks differences correctly even when the document structure is rearranged.
- Allows you to merge XML-relevant modifications.

Figure 55. Three-Way Comparison



Compare Files

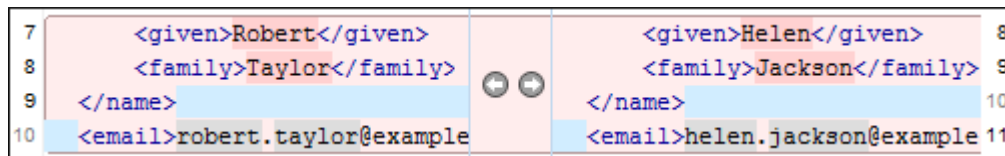
To perform a three-way comparison, follow these steps:

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 actions in the ▾ **Browse** drop-down menu.

Step Result: The selected files are opened in the two side-by-side editors. A text editing mode is used to offer a better view of the differences.

2. Click the **Three-Way Comparison** button on the toolbar and select the base (original) file in the **Base** field. You can specify the path by using the text field, the history drop-down, or the browsing actions 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 \(on page 315\)](#) 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 (original) file. The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes.

Figure 56. 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 \(on page 168\)](#), 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 \(on page 324\)](#) and in the [various menus \(on page 327\)](#) (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

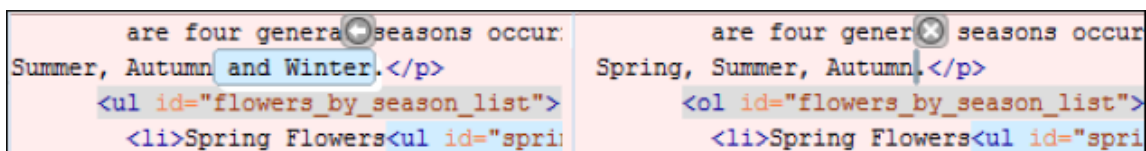
Oxygen JSON Editor offers the following three-way diff algorithms to compare files:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Second-Level Comparisons

For both two-way and three-way comparisons, Oxygen JSON Editor automatically performs a second-level comparison for the **Lines**, **XML Fast**, and **XML Accurate** algorithms. After the first comparison is finished, the second-level comparison for the **Lines** algorithm is processed on text nodes using a word level comparison, meaning that it looks for identical words. For the **XML Fast** and **XML Accurate** algorithms, the second-level comparison is processed using a [syntax-aware comparison \(on page 316\)](#), meaning that it looks for identical *tokens*. This second-level comparison makes it easier to spot precise differences and you can merge or reject the precise modifications.

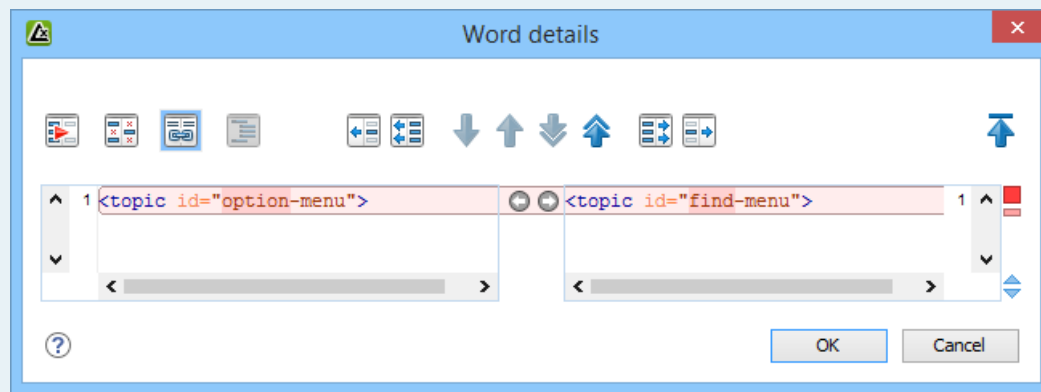
Figure 57. Second-Level Diff Comparison



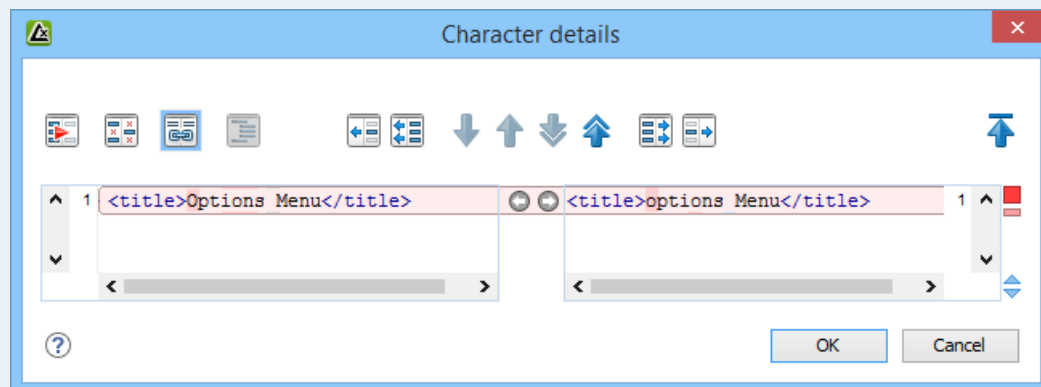
Note:

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 58. Word Level Comparison**

To do a character level comparison, select **Show Character Level details** from the contextual menu or **Compare** menu.

Figure 59. Character Level Comparison**Related information**

[Files Comparison Preferences Page \(on page 166\)](#)

[Compare Directories Tool \(on page 332\)](#)

Starting File Comparison Tool from a Command Line

The file comparison tool can be started by using command-line arguments. In the installation folder there is an executable shell (**diffFiles.bat** on Windows, **diffFiles.sh** on macOS and Linux). To specify the files to compare, you can pass command-line arguments using the following construct: `diffFiles.bat/diffFiles.sh [path to left file] [path to right file] [path to 3-way base file]`.

If three files are specified, the tool will start in the [3-way comparison mode \(on page 316\)](#). If only two files are specified, the tool will start in the [2-way comparison mode \(on page 313\)](#). The first specified file will be added to the left panel in the comparison tool, the second file to the right panel, and the optional third file will be the base (ancestor) file used for a 3-way comparison. If you pass only one argument, you are prompted to manually choose another file.

If you want to launch the file comparison tool from an external application with specified files and you want the file browsing buttons at the top of both panels to be hidden, you should use the `-ext` argument as the first command. There are some additional arguments that are allowed and to see all the details for the command-line construct, type `diffFiles.bat --help` in the command line.

Example:

To do a 3-way comparison, the command line might look like this:

Windows

```
diffFiles.bat "c:\docs\file 1" "c:\docs\file 2" c:\docs\basefile
```



Tip:

If there are spaces in the path names, surround the paths with quotes.

Linux

```
diffFiles.sh home/file1 home/file2 home/basefile
```

macOS

```
diffFiles.sh documents/file1 documents/file2 documents/basefile
```

How to Integrate the File Comparison Tool with Git

The file comparison tool can be integrated with Git clients. It requires that you configure your `.gitconfig` file and then you can simply start the tool from the command line.

To integrate the **Compare Files** tool with your Git client, follow this procedure:

1. Use one of the following methods to instruct your Git client to use the *Oxygen Compare Files* tool:
 - **Manual Configuration** - Locate your Git user-specific configuration file (`.gitconfig`) and edit it with a text editor (for example, in Windows, the `.gitconfig` file is most likely located in your user home directory). Add (or replace) the following lines:

```
[diff]
    tool = oxygendiff

[merge]
    tool = oxygendiff

[difftool "oxygendiff"]
    cmd = '[pathToOxygenInstallDir]/diffFiles.exe' -ext $REMOTE $LOCAL $LOCAL

[mergetool "oxygendiff"]
    cmd = '[pathToOxygenInstallDir]/diffFiles.exe' -ext $LOCAL $REMOTE $BASE $MERGED
    trustExitCode = true

[difftool]
    prompt = false
```

**Note:**

For macOS, the **cmd** lines would start with something like: **sh "/Applications/Oxygen XML Editor/diffFiles.sh"**. For Linux, the **cmd** lines would start with something like: **sh "/Oxygen XML Editor/diffFiles.sh"**.

**Tip:**

On Redhat 7, the following command would work, where the whole command is quoted and then inside that, the path to **diffFiles.sh** is quoted:

```
[difftool "oxygendiff"]
    cmd = '/home/user/Oxygen XML Editor 21/diffFiles.sh' -ext $REMOTE $LOCAL
    $LOCAL

[mergetool "oxygendiff"]
    cmd = '/home/user/Oxygen XML Editor 21/diffFiles.sh' -ext $LOCAL $REMOTE
    $BASE

    $MERGED trustExitCode = true
```

- **Command Line Configuration** - To automatically configure the **.gitconfig** file, you can run the following commands from a command line:

```
git config --global diff.tool oxygendiff
git config --global difftool.oxygendiff.cmd '[Oxygen install dir]/diffFiles.exe -ext
$REMOTE $LOCAL $LOCAL'
git config --global merge.tool oxygendiff
git config --global mergetool.oxygendiff.cmd '[Oxygen install dir]/diffFiles.exe
-ext $LOCAL $REMOTE $BASE $MERGED'
git config --global mergetool.oxygendiff.trustExitCode true
```

**Note:**

For macOS, the *Oxygen* file comparison tool would be specified in the second and fourth commands with something like: **sh "/Applications/Oxygen XML Editor/diffFiles.sh"**. For Linux, it would be something like: **sh "/Oxygen XML Editor/diffFiles.sh"**.

2. To start the **Compare Files** tool and see a comparison of changes for a particular file, run the following command from a command line:

```
git difftool [PathToFile]
```

**Tip:**

If the file you want to compare has conflicts, you can start the **Compare Files** tool as a *merge conflict resolution* tool by running the following command:

```
git mergetool [PathToFile]
```

For more information about the Git *diff* tool syntax, see <https://git-scm.com/docs/git-diff>.

For more information about the Git *mergetool* syntax, see <https://git-scm.com/docs/git-mergetool>.

How to Integrate the File Comparison Tool with Sourcetree

The file comparison tool can be integrated with Sourcetree so that you can use it to compare changes. The advantages of doing this include:

- The *Oxygen Compare Files* tool presents the files side-by-side and makes it much easier to determine real changes.
- The *Oxygen Compare Files* tool includes XML comparison algorithms.
- The *Oxygen Compare Files* tool includes various options for configuring the comparison.
- The *Oxygen Compare Files* tool allows you to navigate through changes.

To integrate the **Compare Files** tool with Sourcetree, follow this procedure, depending on your operating system:

Windows

1. In Sourcetree, go to **Tools > Options**.
2. Go to the **Diff** tab.
3. In the **External Diff/Merge** section, configure the settings as follows:
 - **External Diff Tool** - Select **Custom**.
 - **Diff Command** - Enter the path of the *Oxygen diffFile.exe* file (for example: **c:\Programs\Oxygen XML Editor\diffFiles.exe**).
 - **Arguments** - Enter **-ext \$REMOTE \$LOCAL \$LOCAL**.
 - **Merge Tool** - Select **Custom**.
 - **Diff Command** - Enter the path of the *Oxygen diffFile.exe* file (for example: **c:\Programs\Oxygen XML Editor\diffFiles.exe**).
 - **Arguments** - Enter **-ext \$LOCAL \$REMOTE \$BASE \$MERGED**.
4. Click **OK**.

Result: In Sourcetree, you can now compare file changes with the *Oxygen Compare Files* tool by simply selecting **External Diff** from the contextual menu, **Actions** menu, or **Ctrl+D**.

macOS

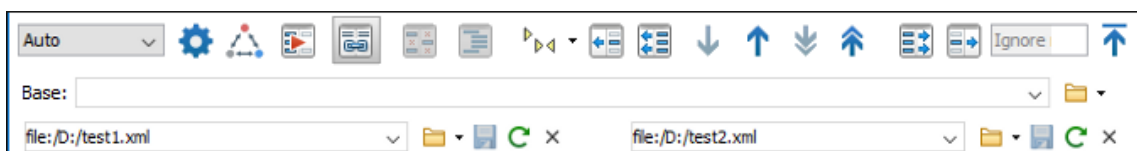
1. In Sourcetree, go to **Sourcetree > Preferences**.
2. Go to the **Diff** tab.
3. In the **External Diff/Merge** section, configure the settings as follows:
 - **External Diff Tool** - Select **Custom**.
 - **Diff Command** - Enter a command-line argument to launch the *Oxygen diffFiles.sh* file (for example: `sh "/Applications/Oxygen XML Editor/diffFiles.sh"`).
 - **Arguments** - Enter `-ext $REMOTE $LOCAL $LOCAL`.
 - **Merge Tool** - Select **Custom**.
 - **Diff Command** - Enter a command-line argument to launch the *Oxygen diffFiles.sh* file (for example: `sh "/Applications/Oxygen XML Editor/diffFiles.sh"`).
 - **Arguments** - Enter `-ext $LOCAL $REMOTE $BASE $MERGED`.
4. Close the preferences dialog box.

Result: In Sourcetree, you can now compare file changes with the *Oxygen Compare Files* tool by simply selecting **External Diff** from the contextual menu or **Actions** menu.

Toolbar and Contextual Menu Actions of the Compare Files Tool

The toolbar of the **Compare Files** tool contains operations that can be performed on the source and target files or XML fragments. Many of the actions are also available in the contextual menu.

Figure 60. 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. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words. This algorithm is not available when the file comparison is in **Author** comparison mode.

- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **Syntax Aware** - Computes differences for the file types or fragments known by Oxygen JSON Editor, taking the syntax (the specific types of tokens) into consideration. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **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 \(on page 166\)](#) where you can configure various options.



Three-Way Comparison

Toggle action that allows you to perform a three-way comparison between the two files displayed in the two editing panes and a base (ancestor) file.



Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.



Synchronized scrolling

Toggles synchronized scrolling on or off so that a selected difference can be seen on both sides of the application window. This option is on by default.



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. This option is not available when in the **Author** comparison mode.



Format and Indent Both Files (**Ctrl + Shift + P** (**Command + Shift + P** on macOS))

Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.



Note:

When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.



Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.

**Copy All Changes from Right to Left**

Copies all changes from the file in the right panel to the file in the left panel.

**Next Block of Changes (Ctrl + Period (Command + Period on macOS))**

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

**Note:**

A change block groups one or more consecutive lines that contain at least one change.

**Previous Block of Changes (Ctrl + Comma (Command + Comma on macOS))**

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

**Next Change (Ctrl + Shift + Period (Command + Shift + Period on macOS))**

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

**Previous Change (Ctrl + Shift + Comma (Command + Shift + M on macOS))**

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

**Copy All Changes from Left to Right**

Copies all changes from the file in the left panel to the file in the right panel.

**Copy Change from Left to Right**

Copies the selected difference from the file in the left panel to the file in the right panel.

Ignore Nodes by XPath

You can use this text field to enter an [XPath expression \(on page 474\)](#) to ignore certain nodes from the comparison. It will be processed as XPath version 2.0. You can also enter the name of the node to ignore all nodes with the specified name (for example, if you want to ignore all ID attributes from the document, you could simply enter `@id`). This field is only available when comparing XML documents using the **XML Fast** or **XML Accurate** algorithms.

**Note:**

If an XPath expression is specified in the [Ignore nodes by XPath option \(on page 168\)](#) 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 macOS))

Jumps to the first change.

Base

Available for [three-way comparisons \(on page 316\)](#). It is the base file that will be compared with the files opened in the left and right editors. You can specify the path to the file by using the text field, its history drop-down, or the browsing actions in the  ▾ **Browse** drop-down menu.

Left-Side (Source) File

You can specify the path to the file to be compared on the left side (source) by using the text field, its history drop-down, or the browsing actions in the  ▾ **Browse** drop-down menu.



Save

Saves the changes made in the source (left-side) file.



Reload

Reloads the source (left-side) file.



Close

Closes the source (left-side) file.

Right-Side (Target) File

You can specify the path to the file to be compared on the right side (target) by using the text field, its history drop-down, or the browsing actions in the  ▾ **Browse** drop-down menu.



Save

Saves the target (right-side) file.



Reload

Reloads the target (right-side) file.



Close

Closes the target (right-side) file.

Compare Files Tool Menus

The menus in the **Compare Files** tool contain some of the same actions that are on the toolbar, as well as some common actions that are identical to the same actions in the Oxygen JSON Editor menus. The menu actions include:

File Menu

Source > Open

Browses for a file that will be displayed in the left panel.

Source > Open URL

Browses for a remote file that will be displayed in the left panel.

Source > Open File from Archive

Browses an archive for a file that will be displayed in the left panel.

Source > Reload

Reloads the file in the left panel.

Source > Save

Saves the changes made to the file in the left panel.

Source > Save As

Allows you to choose a destination to save the file in the left panel.

Source > Close

Closes the file in the left panel.

Target > Open

Browses for a file that will be displayed in the right panel.

Target > Open URL

Browses for a remote file that will be displayed in the right panel.

Target > Open File from Archive

Browses an archive for a file that will be displayed in the right panel.

Target > Reload

Reloads the file in the right panel.

Target > Save

Saves the changes made to the file in the right panel.

Target > Save As

Allows you to choose a destination to save the file in the right panel.

Target > Close

Closes the file in the right panel.

Base > Open

Browses for a file that will be compared with both files in a [three-way comparison \(on page 316\)](#).

Base > Open URL

Browses for a remote file that will be compared with both files in a [three-way comparison \(on page 316\)](#).

Base > Open File from Archive

Browses an archive for a file that will be compared with both files in a [three-way comparison \(on page 316\)](#).

**Save Results as HTML (Available in Text mode only)**

Generates an HTML file that contains detailed information about the comparison result.

Save Comparison as Document with Tracked Changes (Available for two-way comparison in Author mode only)

Allows you to merge two compared documents based on the differences detected and save the results as a specified file that includes the special change tracking marks. You can load the resulting file in **Oxygen's Author** mode to review the changes that resulted from the merge process and you can accept or reject them. Note that if the documents to be compared already contain tracked changes, they will be automatically accepted before generating the output file.

Close (Ctrl + W (Command + W on macOS))

Closes the application.

Edit Menu**Cut**

Cut the selection from the currently focused editor panel to the clipboard.

**Copy**

Copy the selection from the currently focused editor panel to the clipboard.

**Paste**

Paste content from the clipboard into the currently focused editor panel.

Select all

Selects all content in the currently focused editor panel.

**Undo**

Undo changes in the currently focused editor panel.

**Redo**

Redo changes in the currently focused editor panel.

Find Menu**Find/Replace**

Perform *find/replace* operations in the currently focused editor panel.

Find Next

Go to the next match using the same options as the last *find* operation. This action runs in both editor panels.

Find Previous

Go to the previous match using the same options as the last *find* operation. This action runs in both editor panels.

Compare Menu



Three-Way Comparison

Toggle action that allows you to perform a three-way comparison between the two files displayed in the two editing panes and a base (ancestor) file.



Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.



Next Block of Changes (Ctrl + Period (Command + Period on macOS))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.



Note:

A change block groups one or more consecutive lines that contain at least one change.



Previous Block of Changes (Ctrl + Comma (Command + Comma on macOS))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.



Next Change (Ctrl + Shift + Period (Command + Shift + Period on macOS))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.



Previous Change (Ctrl + Shift + Comma (Command + Shift + M on macOS))

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.



Last Change (Ctrl + E (Command + E on macOS))

Jumps to the last change.



First Change (Ctrl + B (Command + B on macOS))

Jumps to the first change.

 Copy All Changes from Right to Left

Copies all changes from the file in the right panel to the file in the left panel.

 Copy All Changes from Left to Right

Copies all changes from the file in the left panel to the file in the right panel.

 Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.

 Copy Change from Left to Right

Copies the selected difference from the file in the left panel to the file in the right panel.

 Show Word Level Details

Provides a word-level comparison of the selected change.

 Show Character Level Details

Provides a character-level comparison of the selected change.

 Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on macOS))

Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.

**Note:**

When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.

Options Menu

Preferences

Opens the preferences dialog box that includes numerous pages of options that can be configured.

Menu Shortcut Keys

Opens the **Menu Shortcut Keys** option page where you can configure keyboard shortcuts available for menu items.

Reset Global Options

Resets options to their default values. Note that this option appears only when the tool is executed as a stand-alone application.

Import Global Options

Allows you to import an options set that you have previously exported.

Export Global Options

Allows you to export the current options set to a file.

Help Menu

Help (F1)

Opens a **Help** dialog box that displays the User Manual at a section that is appropriate for the context of the current cursor position.

Use Online Help

If this option is selected, when you select Help or press F1 while hovering over any part of the interface, Oxygen JSON Editor attempts to open the help documentation in online mode. If this option is not selected or an internet connection fails, the help documentation is opened in offline mode.

Report problem

Opens a dialog box that allows the user to write the description of a problem that was encountered while using the application. You can change the URL where the reported problem is sent by using the `com.oxygenxml.report.problems.url` system property. The report is sent in XML format through the `report` parameter of the POST HTTP method.

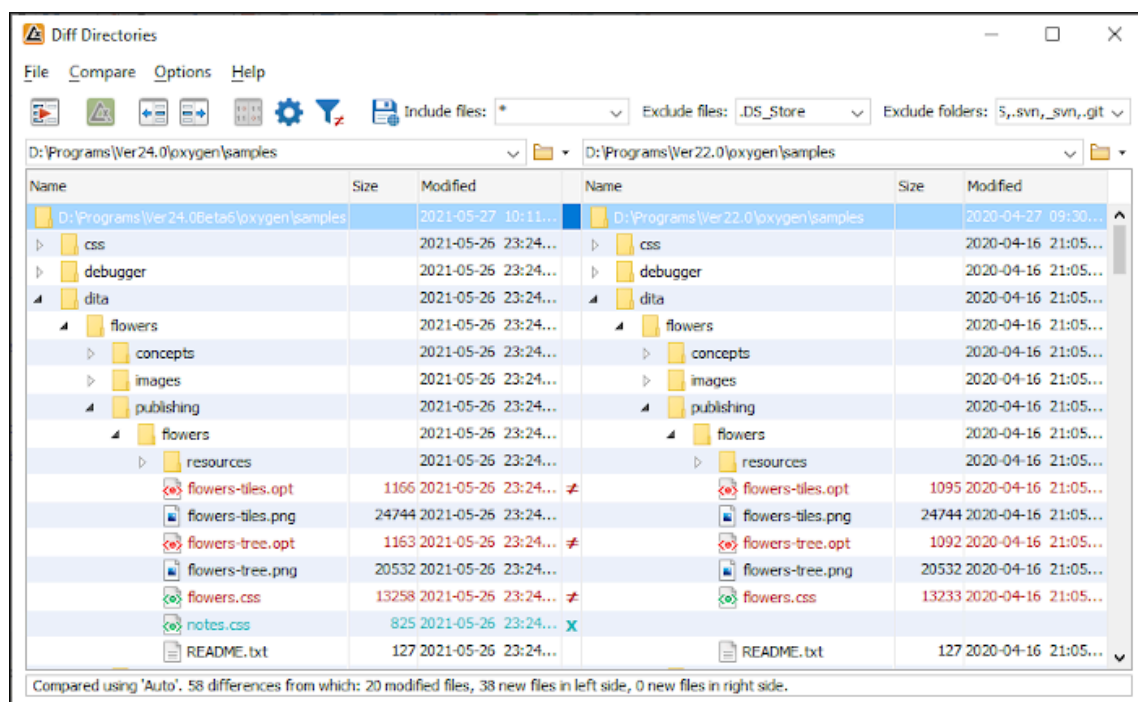
Support Center

Opens the Oxygen JSON Editor Support Center web page in a browser.

Compare Directories Tool

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 JSON Editor installation folder (`diffDirs.exe`).

Figure 61. Diff Directories Dialog Box



Starting the Tool from a Command Line

The directory comparison tool can also be started by using command-line arguments. In the installation folder there is an executable shell (`diffDirs.bat` on Windows, `diffDirs.sh` on macOS and Linux). To specify the directories to compare, you can pass command-line arguments using the following construct:

```
diffDirs.bat/diffDirs.sh [directory path 1] [directory path 2].
```

If you pass only one argument, you are prompted to manually choose the second directory or archive.

Example:

To do a comparison between two directories, the command line would look like this:

Windows

```
diffDirs.bat "c:\documents new" "c:\documents old"
```



Tip:

If there are spaces in the path names, surround the paths with quotes.

Linux

```
diffDirs.sh home/documents1 home/documents2
```

macOS

```
diffDirs.sh documents1 documents2
```

Directory Comparisons

To perform a directory comparison, follow these steps:

1. Select a folder in the left panel and the folder you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the **Browse for local directory** action in the **Browse** drop-down menu.

Step Result: The selected directory structures are opened in the two side-by-side panels.

2. To highlight the differences between the two folders, click the **Perform Directories Differencing** button from the toolbar.
3. You can also use the **Diff Options** button to access the [Directories Comparison preferences page \(on page 169\)](#) 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 JSON Editor treat supported archives as directories, select the **Look in archives** option ([on page 170](#)) 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 ([on page 170](#)). You can double-click lines marked with the **≠** symbol to open a **Compare Files** window, which shows the differences between the two files.

The directories that contain files that differ are expanded automatically so that you can focus directly on the differences. You can merge the contents of the directories by using the copy actions. If you double-click (or press **Enter**) on a line with a pair of files, Oxygen JSON Editor starts a [file comparison \(on page 313\)](#) between the two files, using the **Compare Files** tool.

Related information

[Compare Files Tool \(on page 313\)](#)

Toolbar and Contextual Menu Actions of the Compare Directories Tool

The toolbar of the **Compare Directories** tool contains operations that can be performed on the compared directory structure. Some of the toolbar actions are also available in the contextual menu.

Figure 62. Compare toolbar



Toolbar Actions



Perform Directories Differencing

Looks for differences between the two directories displayed in the left and right side of the application window.



Perform Files Differencing

Opens the [Compare Files tool \(on page 313\)](#) 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 \(on page 169\)](#) where you can configure various options.

**Show Only Modifications**

Displays a more uncluttered file structure by hiding all identical files.

**Save Results as HTML**

Generates an HTML file that contains detailed information about the comparison result.

File and folder filters

Differences can be filtered using three combo boxes: **Include files**, **Exclude files**, and **Exclude folders**. They come with predefined values and are editable to allow custom values. All of them accept multiple comma-separated values and the * and ? wildcards. For example, to filter out all JPEG and GIF image files, edit the **Exclude files** filter box to read ***.jpeg, *.png**. Each filter includes a drop-down menu with the latest 15 filters applied.

Contextual Menu Actions**Perform Files Differencing**

Opens the [Compare Files tool \(on page 313\)](#) that allows you to compare the currently selected files.

**Binary Compare**

Performs a byte-level comparison on the selected files.

**Copy Change from Right to Left**

Copies the selected difference from the file in the right panel to the file in the left panel.

**Copy Change from Left to Right**

Copies the selected difference from the file in the left panel to the file in the right panel.

Open

If the action is invoked on a file, the selected file is opened in Oxygen JSON Editor. If the action is invoked on a directory, the selected directory is opened in the default file browser for your particular operating system.

Open in System Application

Opens the selected file in the system application that is associated with that type of file. The action is available when launching the **Compare Directories** tool from the **Tools** menu in Oxygen JSON Editor.

Show in Explorer

Opens the default file browser for your particular operating system with the selected file highlighted.

Compare Directories Tool Menus

The menus in the **Compare Directories** tool contain some of the same actions that are on the toolbar, as well as some common actions that are identical to the same actions in the Oxygen JSON Editor menus. The menu actions include:

File Menu



Save Results as HTML

Generates an HTML file that contains detailed information about the comparison result.

Close (**Ctrl + W** (**Command + W** on macOS))

Closes the application.

Compare Menu



Perform Directories Differencing

Looks for differences between the two directories displayed in the left and right side of the application window.



Perform Files Differencing

Opens the **Compare Files** tool ([on page 313](#)) that allows you to compare the currently selected files.



Copy Change from Right to Left

Copies the selected change from the right side to the left side (if there is no file/folder in the right side, the left file/folder is deleted).



Copy Change from Left to Right

Copies the selected change from the left side to the right side (if there is no file/folder in the left side, the right file/folder is deleted).

Options Menu

Preferences

Opens the preferences dialog box that includes numerous pages of options that can be configured.

Menu Shortcut Keys

Opens the **Menu Shortcut Keys** option page where you can configure keyboard shortcuts available for menu items.

Reset Global Options

Resets options to their default values. Note that this option appears only when the tool is executed as a stand-alone application.

Import Global Options

Allows you to import an options set that you have previously exported.

Export Global Options

Allows you to export the current options set to a file.

Help Menu

Help (F1)

Opens a **Help** dialog box that displays the User Manual at a section that is appropriate for the context of the current cursor position.

Use Online Help

If this option is selected, when you select Help or press F1 while hovering over any part of the interface, Oxygen JSON Editor attempts to open the help documentation in online mode. If this option is not selected or an internet connection fails, the help documentation is opened in offline mode.

Report problem

Opens a dialog box that allows the user to write the description of a problem that was encountered while using the application. You can change the URL where the reported problem is sent by using the `com.oxygenxml.report.problems.url` system property. The report is sent in XML format through the `report` parameter of the POST HTTP method.

Support Center

Opens the Oxygen JSON Editor Support Center web page in a browser.

Compare Images

You can use the **Compare Directories** tool to compare images. If you double-click a line that contains two different images, the **Compare images** window is displayed. This dialog box presents the images in the left

and right sides, scaled to fit the available view area. You can use the contextual menu actions to scale the images to their original size or scale them down to fit in the view area.

The supported image types are: *GIF*, *JPG*, *JPEG*, *PNG*, and *BMP*.

Compare Directories Against a Base (3-Way) Tool

The **Compare Directories Against a Base (3-way)** tool allows you to perform three-way comparisons on directories to help you identify and merge changes between multiple modifications of the same directory structure. It is especially helpful for teams that have multiple authors contributing documents to the same directory system. It offers information about conflicts and changes, and includes actions to easily merge, accept, overwrite, or ignore changes to the directory system.

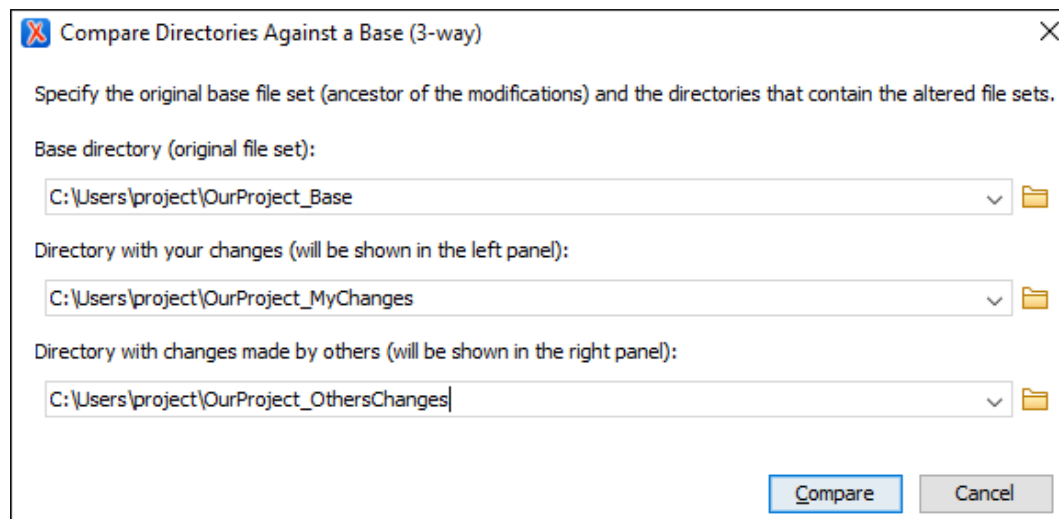
How to Perform 3-Way Directory Comparisons

To perform a 3-way directories comparison, follow these steps:

1. Select **3 Compare Directories Against a Base (3-way)** from the **Tools** menu.

Step Result: This opens a dialog box that allows you to select the 3 file sets that will be used for the comparison.

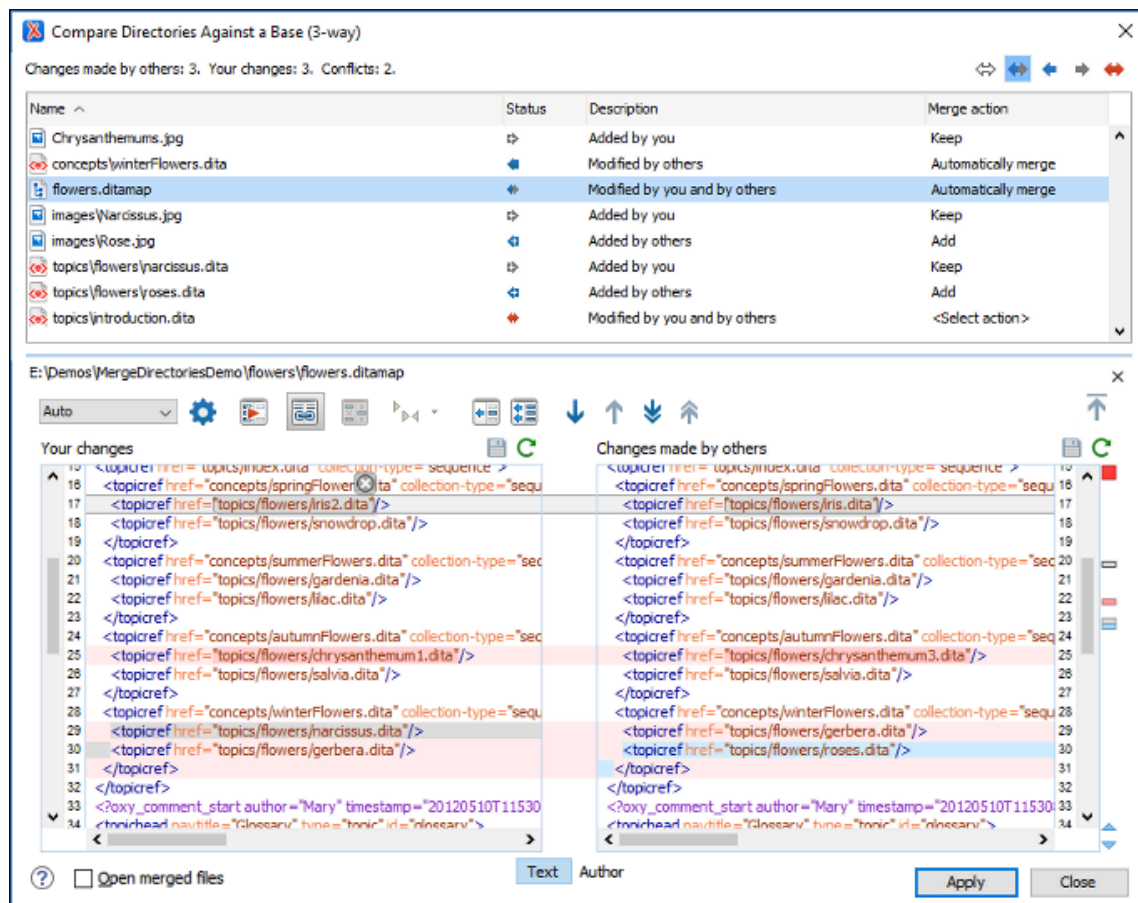
Figure 63. Compare Directories Against a Base File Set Chooser



2. Select the file sets to be compared:
 - **Base directory** - This is the original (base) file set before any modifications were made by you or others.
 - **Directory with your changes** - This is the file set with changes that you have made. This file set will be displayed in the left panel in the comparison tool.
 - **Directory with changes made by others** - This is the file set with changes made by others that you want to merge with your changes. This file set will be displayed in the right panel in the comparison tool.
3. Click the **Compare** button to compare the file sets and open the comparison and merge tool.
4. Use the features and actions described in the next section to identify and merge the changes.

3-Way Directory Comparison and Merge Tool

Figure 64. Comparison and Merge Tool



The 3-way directory comparison and merge tool includes the following information, features, and actions:

Number of Changes and Conflicts

The first thing you see in the top-left corner of the tool is the grand total of all the changes made by others, changes made by you, and the number of conflicts.

Filter Buttons

In the top-right corner you can use the toggle buttons to filter the list of modifications:

Show all files

Use this button to show all modified and unmodified files, as well as conflicts.

Show only files modified by you and others

Filters the list to show all files that have been modified, including conflicts.

Show only files modified by others

Filters the list to only show the files that were modified by others.

Show only files modified by you

Filters the list to only show the files that were modified by you.

Show only conflicting files

Filters the list to only show files that contain conflicts.

List of Files Panel

This panel shows the list of files in the compared file sets based upon the filter button that is selected. This panel includes the following sortable columns:

- **Name** - The file names.
- **Status** - An icon that represents the file status. Red icons indicate some sort of conflict. Gray icons indicate modifications made by you. Blue icons indicate modifications made by others.
- **Description** - A description of the file status.
- **Merge Action** - This column provides a drop-down menu for each file that allows you to choose some merge actions depending upon its status. A default action is always set to **Automatically merge** the changes made by others with your changes. If there is a conflict, the default is **<Select action>** and you are required to make a selection. Click this column to access the drop-down menu where you can make a selection. The same actions are available in the contextual menu.



Tip:

If the solution proposed in the **Merge Action** column for any particular file is not satisfactory, you can change it directly in that column (even if that file is not selected) without automatically re-triggering the comparison (except for in certain cases where re-triggering the comparison is necessary).

You can click a file to open it in the file comparison panel (the file from your file set is shown in the left panel while the file from the file set with changes made by others is shown in the right panel). For image files, the comparison panel shows a preview of the image. For other binary files, a preview is not available and you will just see its status.

File Comparison Panels

If you click a file in the top panel, the file is opened in this file comparison section. The file from your file set is shown in the left panel and the file from the other file set is shown in the right panel.



Note:

If Oxygen JSON Editor does not recognize the file type, a dialog box will be displayed that allows you to select the type of editor you want it to be associated with for this comparison (if you want Oxygen JSON Editor to remember this association, you can select the **Associate file type with editor** option at the bottom of the dialog box).

This panel includes the following information and toolbar actions:

File Path

The first thing you see in this panel is the file path where merge actions will be applied if you make changes.

✕ Close

Closes the file comparison panel.

Algorithm Drop-down Menu

This drop-down menu allows you to select one of the following diff algorithms to be used for file comparisons:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **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 \(on page 166\)](#) where you can configure various options.



Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.



Synchronized scrolling

Toggles synchronized scrolling. When toggled on, a selected difference can be seen in both panels.



Ignore Whitespaces

Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces. This option is not available when the file comparison is in **Author** mode.



Tags Display Mode

Allows you to select the amount of markup to be displayed in the **Author** visual mode. You can choose between: **Full Tags with Attributes**, **Full Tags**,

Block Tags,  **Block Tags without Element Names,**  **Inline Tags,**  **Partial Tags,** or  **No Tags.**



Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.



Copy All Changes from Right to Left

Copies all changes from the file in the right panel to the file in the left panel.



Next Block of Changes (Ctrl + Period (Command + Period on macOS))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.



Note:

A change block groups one or more consecutive lines that contain at least one change.



Previous Block of Changes (Ctrl + Comma (Command + Comma on macOS))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.



Next Change (Ctrl + Shift + Period (Command + Shift + Period on macOS))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.



Previous Change (Ctrl + Shift + Comma (Command + Shift + M on macOS))

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.



First Change (Ctrl + B (Command + B on macOS))

Jumps to the first change.

Left-Side File (Your changes)

Above the panel you can see the file path and the following two buttons:



Save

Saves changes made to the file.



Reload

Reloads the file.

Right-Side File (Changes made by others)

Above the panel you can see the file path and the following two buttons:



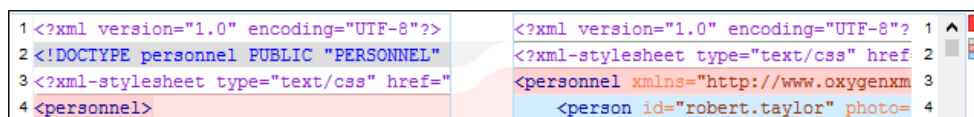
Reload

Reloads the file.

Displaying Changes in the File Comparison Panels

The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes.

Figure 65. File Comparison Panels



The differences are also highlighted in several colors, depending on the type of change, and dynamic lines connect the compared fragments in the middle section between the two panes. The highlighting colors can be customized in the [Files Comparison / Appearance preferences page \(on page 168\)](#), but the default colors and their shades mean the following:

- **Pink** - Identifies modifications on either side.
- **Gray** - Identifies an addition of a node in the left side (your outgoing changes).
- **Blue** - Identifies an addition of a node in the right side (incoming changes).
- **Lighter Shade** - Identifies blocks of changes that can be merged in their entirety.
- **Darker Shade** - Identifies specific changes within the blocks that can be merged more precisely.

Direct Editing Actions in the File Comparison Panels

In addition to selecting merge actions from the drop-down menus in the **Merge Action** column in the top panel, you can also edit the files directly in the left pane (your local changes). The two editors are constantly synchronized and the differences are refreshed when you save the modified document ( **Save** button or **Ctrl+S**) or when you click the  **Perform File Differencing** button.

A variety of actions are available in the contextual menu in both editing panes. The tool also includes some inline actions to help you merge, copy, or remove changes. When you select a change, the following inline action widgets are available, depending on the type of change:



Append right change to left

Copies the content of the selected change from the right side and appends it on the left side.

 Copy change from right to left

Replaces the content of a change in the left side with the content of the change in the right side.

 Remove change

Removes the change from the left side.

Anytime you save manual changes ( **Save** button or **Ctrl+S**), the selection in the **Merge Action** column in the top panel automatically changes to **Use merged** and a copy of the original file is kept so that you can revert to the original file if necessary. To discard your manual changes and revert to your original changes, select a different action in the **Merge Action** drop-down menu.

Open Merged Files

If you select this option, all the files that will be modified by the merge operation will be opened in the editor after the operation is finished.

Applying Changes

When you click the **Apply** button, all the merge actions you have selected and the changes you have made will be processed.

If there are unresolved conflicts (conflicts where no merge action is selected in the **Merge Action** drop-down menu), a dialog box will be displayed that allows you to choose how to solve the conflicts. You can choose between the following:

- **Keep your changes** - If you select this option and then click **Apply**, your local changes will be preserved for the unresolved conflicts.
- **Overwrite your changes** - If you select this option and then click **Apply**, your local changes will be overwritten with the changes made by others, for the unresolved conflicts.
- **Cancel** - You can click the **Cancel** button to go back to the merge tool to resolve the conflicts individually.

Canceling Changes

If you click the **Cancel** button at the bottom of the merge tool, all the changes you made in the tool will be lost.

Related information

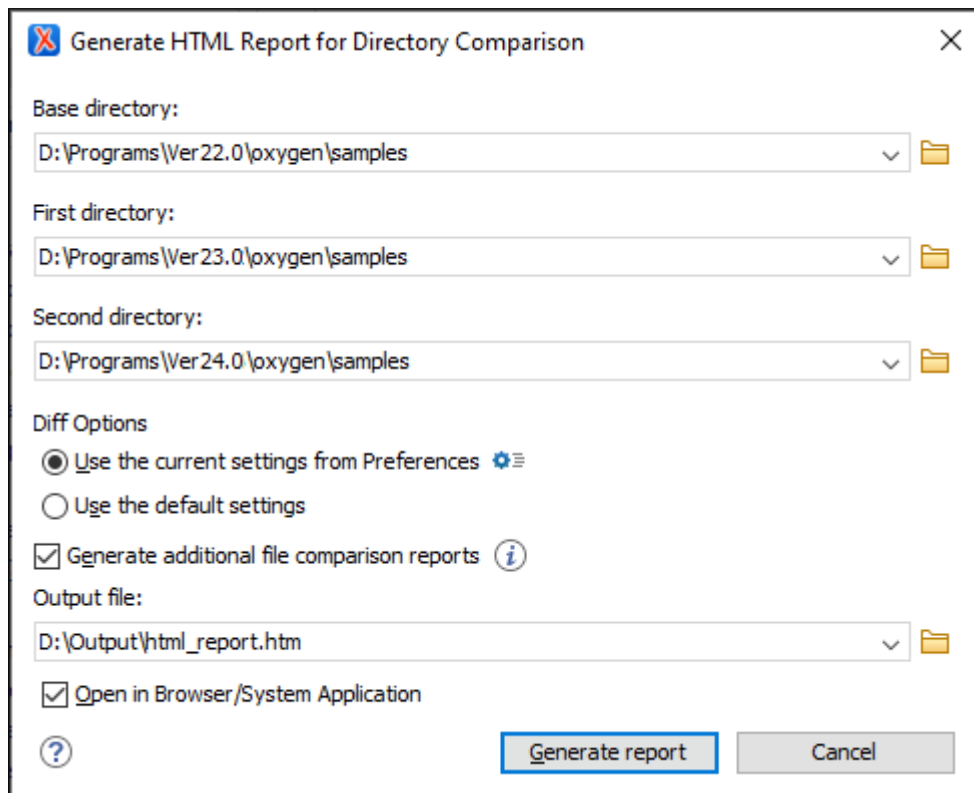
[Compare Directories Tool \(on page 332\)](#)

[Compare Files Tool \(on page 313\)](#)

Generate HTML Report for Directory Comparison

The **Generate HTML report for directory comparison** tool can be used to generate a report in the form of an HTML file that contains the results of a directory comparison (for either 2-way or 3-way comparisons). The **Generate HTML report for directory comparison** action for invoking the tool can be found in the **Tools** menu. It opens a dialog box where you can specify the directories to compare as well as some other options.

Figure 66. Generate HTML Report for Directory Comparison Dialog Box



The **Generate HTML report for directory comparison** dialog box contains the following options:

Base directory

Specifies the path of the base directory that the other two directories will be compared against in a 3-way comparison. This field should be left empty for 2-way comparisons.

First directory

Specifies the path of the first directory to be included in the comparison.

Second directory

Specifies the path of the second directory to be included in the comparison.

Diff options

Specifies which option set to use for generating the comparison report. If you choose **Use the current settings from Preferences**, the options set in the **Directories Comparison** preferences page (*on page 169*) and the **include/exclude filter options** in the **Compare Directories** tool (*on page 335*) are taken into account when generating the comparison result. You can also click the  **Diff options** button to open the **Directories Comparison** preferences page where you can

see or modify the current settings. If you choose **Use the default settings**, the default values for all settings are used.

Generate additional file comparison reports

Generates further comparison reports for all non-binary modified file pairs and provides links to them in the main report (in the middle cells of the results table). [See the example below \(on page 347\)](#). These additional file comparison reports are saved to a directory that will have the same parent directory and the same name as the output file provided, suffixed by **"-OXY-FC-REPORTS"**. The links created in the main report are relative to this directory. If the main HTML report is later copied or moved to another location, to retain full functionality in the browser, the directory with the additional file comparison reports must also be copied/moved to the same location.



Note:

Generating additional file comparison reports could significantly increase the execution time. A progress tracker for the whole operation is available.



Tip:

An XPath expression specified in the **Ignore nodes by XPath** text field within the [Files Comparison preferences page \(on page 166\)](#) is now taken into account if you enable the **Generate additional file comparison reports** option.

Output file

Specifies the path for an output file to save the comparison results file.

Open in Browser/System Application

Opens the comparison results file in the browser or system application that is associated with HTML files.

After clicking the **Generate report** button, a report in the form of an HTML file is generated with details about the comparison results.

Figure 67. HTML Report for Directory Comparison

Differences: 13

Comparison details: ☒ all differences (13) ☐ outgoing (5) ☐ incoming (5) ☐ conflicts (3)

Base folder: D:/Sample1/dita-flowers/flowers-base/

Folder 1: D:/Sample1/dita-flowers/flowers-by-John/

Folder 2: D:/Sample1/dita-flowers/flowers-by-Mary/

File name	Size	Modified		File name	Size	Modified
concepts/autumnFlowers.dita	1151	2021-07-06 01:49:12	* *	concepts/autumnFlowers.dita	1143	2021-07-16 06:52:21
concepts/glossaryGenus.dita	571	2021-07-16 06:54:17	*	concepts/glossaryGenus.dita	577	2021-07-06 01:49:12
concepts/glossaryPanicle.dita	483	2021-07-15 06:53:34	*	concepts/glossaryPanicle.dita	495	2021-07-06 01:49:12
images/Gerbera.jpg	10134	2021-07-06 01:49:12	*	images/Gerbera.jpg	22776	2021-07-16 05:55:25
			+	publishing/flowers/resources/images/flower_logo.png	6178	2021-07-06 01:49:12
			+	publishing/flowers/READ-ME.txt	127	2021-07-06 01:49:12
publishing/flowers/README.txt	127	2021-07-06 01:49:12	-			
tasks/gardenPreparation.dita	2275	2021-07-06 01:49:12	*	tasks/gardenPreparation.dita	2291	2021-07-15 12:21:02
topics/flowers/chrysanthemum.dita	2949	2021-07-15 07:48:25	*	topics/flowers/chrysanthemum.dita	2932	2021-07-06 01:49:12
topics/flowers/gerbera.dita	2432	2021-07-16 06:18:28	* *	topics/flowers/gerbera.dita	2456	2021-07-16 06:25:13
topics/flowers/snowdrop.dita	2883	2021-07-15 13:57:59	*	topics/flowers/snowdrop.dita	2806	2021-07-06 01:49:12
topics/test/		2021-07-15 12:36:56	+			
topics/introduction.dita	770	2021-07-16 06:58:21	* *	topics/introduction.dita	758	2021-07-15 12:12:46

Figure 68. Example of an Additional File Comparison Report

← → ↻ ⓘ File | D:/Output/html_report-OXY-FC-REPORTS/fc-36.html

Differences: 5 difference blocks, 8 differences in total

Comparison details by difference blocks: ☒ all (5) ☐ incoming (3) ☐ outgoing (2)

Base file: D:/Sample1/dita-flowers/flowers-base/topics/flowers/gerbera.dita

File 1: D:/Sample1/dita-flowers/flowers-by-John/ topics/flowers/gerbera.dita		File 2: D:/Sample1/dita-flowers/flowers-by-Mary/ topics/flowers/gerbera.dita
8 is a genus of ornamental plants from the daisy family (Asteraceae). It was named in 9 honor of the German naturalist Traugott Gerber (1710-1743) who travelled extensively in Russia and 10 was a friend of Carl Linnaeus </p>	+ -	8 is a genus of ornamental plants from the sunflower family (Asteraceae). It was named in 9 honor of the German naturalist Traugott Gerber </p>
12 13 <!--Maybe we can add more pictures here....--> 14 15 <p>It has approximately 30 species in the wild, extending to South America, Africa and tropical	+ -	11 <p>It has approximately 30 species in the wild, extending to South America, Africa and tropical
18 also known as Transvaal daisy or Barberton Daisy. 19 Gerbera species bear a large capitulum with striking, two-lipped ray florets in yellow,	- +	14 also known as Transvaal daisy or Barberton Daisy. 15 16 17 Gerbera species bear a large capitulum with striking, two-lipped ray florets in yellow,
25 The domesticated <xref keyref="cultivar" format="dita">cultivars</xref> are mostly a result of a	- +	24 The domesticated <xref format="dita" keyref="cultivar">cultivars</xref> are mostly a result of a

Resources

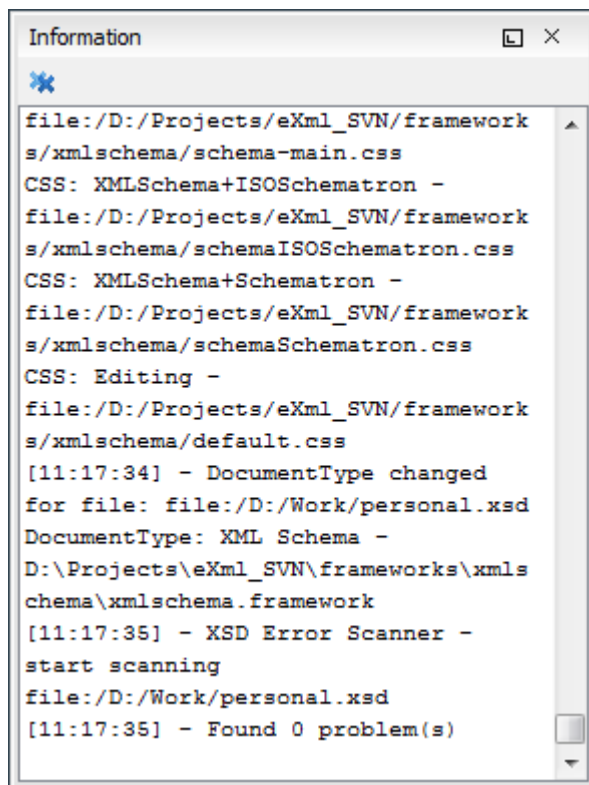
For more information about how to generate HTML comparison reports, watch our video demonstration:

<https://www.youtube.com/embed/6jPccHKUNNk>

Related information[Compare Directories Tool \(on page 332\)](#)[Compare Directories Against a Base \(3-Way\) Tool \(on page 338\)](#)[Compare Files Tool \(on page 313\)](#)

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 69. 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 (on page 185) in the **Views** preference page.

To make the view visible, select **Window > Show View > Information**.

Editor Highlights

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

Highlights are generated in both **Text** and **Author** mode when the following actions generate results:

- **Find/Replace in Files** (on page 277)
- **Find/Replace** (on page 273)
- **Open/Find Resource** (on page 267)
- **Find All**
- **Find All Elements** (on page 283)
- **XPath in Files** (on page 257)

By default, Oxygen JSON Editor uses a different color for each type of highlight (*XPath in Files*, *Find/Replace*, etc.) You can customize these colors and the maximum number of highlights displayed in a document on the **Editor preferences page** (on page 115). 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 editor pane.
- Clicking the **Next** and **Previous** buttons () from the bottom of the range ruler, located at the right side of the editor pane.



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, located at the right side of the editor pane.
- Close the results tab at the bottom of the editor that contains the output of the action that generated the highlights.
- Click the  **Remove all** button on the right side of the **Results** view at the bottom of the editor.



Note:

Use the  **Highlight all results in editor** button (on the right side of the **Results** panel) to either display all the highlights or hide them.

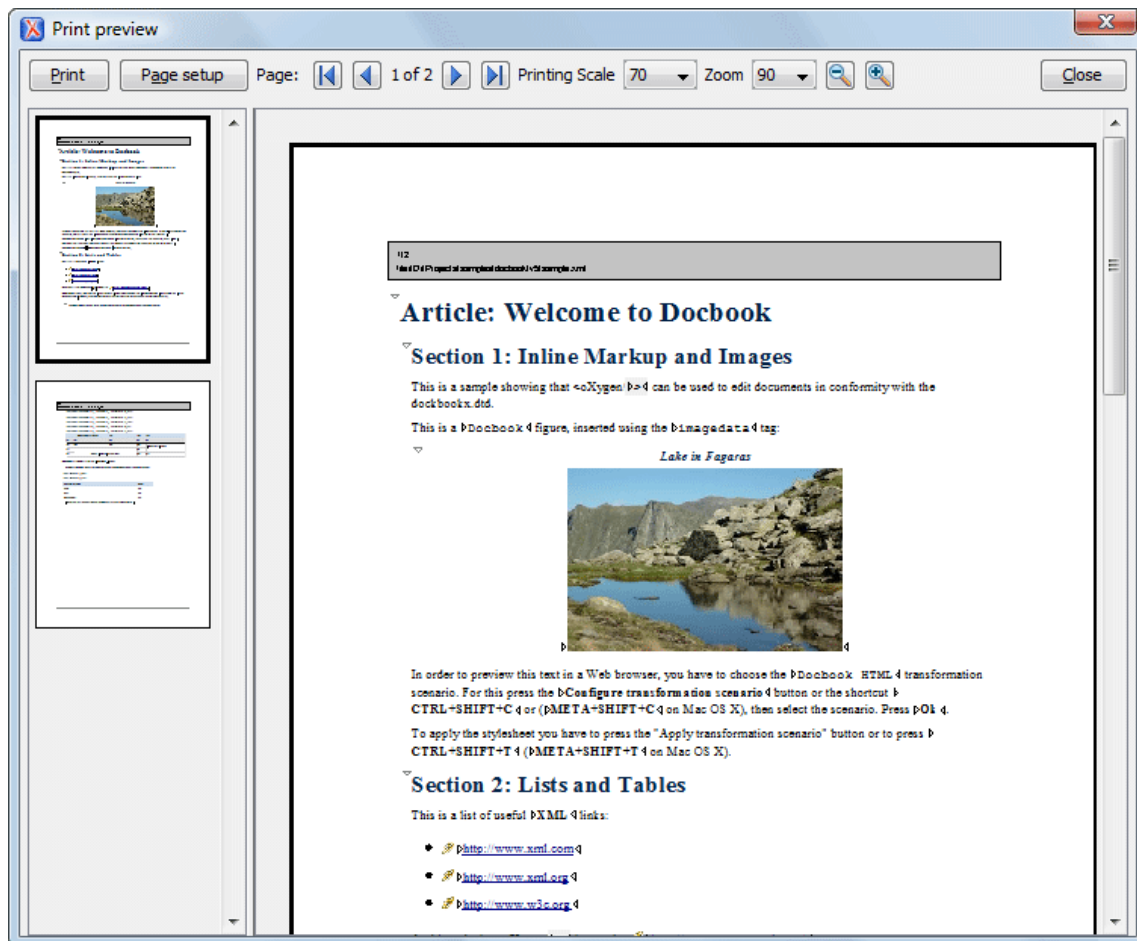
Printing a Document

Printing is supported in **Text** and **Grid** modes.

The **Print** (**Ctrl + P** (**Command + P** on macOS)) action that is available from **File** menu displays a series of dialog boxes that allow you to configure print settings. After defining the settings in each dialog box, click **OK** to continue to the next one.

A **Print Preview** action is also available in the **File** menu. It first opens a **Page Setup** dialog box where you can define some paper, orientation, and margin settings. After you click **OK**, it displays the **Print Preview** dialog box where you can see a preview of how the document will look when it is printed..

Figure 70. 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 (on page 118) is selected), the printed output will include the line numbers.
- If you are printing an XML document that is opened in **Text** mode and the *folded* support is activated (the **Enable folding** option (on page 118) is selected), the printed output will include the current *folded* state. Note that this applies to printing an entire document and not selections within the document.

- If you are printing an XML document that is opened in **Text** mode and a block of content is selected, you have the ability to print only the selection of text rather than the entire document. When you invoke the print action with a block of content selected in **Text** mode, a dialog box will be presented that offers you the choice to print the selection or the entire document.

6.

Editing Supported Document Types

Oxygen JSON Editor supports editing a large variety of document types, including JSON, JSON Schema, YAML, Markdown, CSS, LESS, Java, JavaScript, Python, SQL, Text, Properties, Batch, Shell, PowerShell, Dockerfile, and more. There is also minimal support to edit XML files in **Text** mode, and you have access to the **Outline** view, syntax highlights, and validation and content completion if a schema is associated with the document.

Each type of document has unique features and options and this chapter includes a large amount of information about editing numerous types of documents and various editing features that are provided in Oxygen JSON Editor.

Related information

[Built-in Frameworks \(Document Types\) \(on page 470\)](#)

Editing CSS Stylesheets

Oxygen JSON Editor includes a built-in editor for CSS stylesheets. This section presents the features of the CSS editor and how these features should be used. The features of the CSS editor include:

- **Create new CSS files and templates** - You can use the built-in new file wizards to [create new CSS documents or templates \(on page 224\)](#).
- **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 JSON Editor makes CSS files more readable.
- **Shortcut to open resources** - You can use **Ctrl + Single-Click (Command + Single-Click on macOS)** 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 JSON Editor includes a built-in *CSS Validator*, integrated with general validation support. This makes the usual validation features for presenting errors also available for CSS stylesheets.

When you edit a CSS document, you can access the [CSS validator options \(on page 160\)](#) 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 \(on page 160\)](#). 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 \(on page 352\)](#) when this profile is selected.

Specify Custom CSS Properties

To specify custom CSS properties, follow these steps:

1. Create a file named `CustomProperties.xml` that has the following structure:

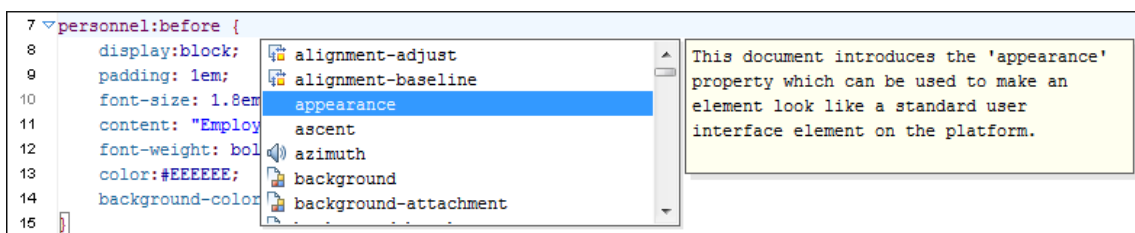
```
<?xml version="1.0" encoding="UTF-8"?>
<css_keywords
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.oxygenxml.com/ns/css
http://www.oxygenxml.com/ns/css/CssProperties.xsd"
  xmlns="http://www.oxygenxml.com/ns/css">
  <property name="custom">
    <summary>Description for custom property.</summary>
    <value name="customValue"/>
    <value name="anotherCustomValue"/>
  </property>
</css_keywords>
```

2. Go to your desktop and create the `builtin/css-validator/` folder structure.
3. Press and hold **Shift** and right-click anywhere on your desktop. From the contextual menu, select **Open Command Window Here**.
4. In the command line, run the `jar cvf custom_props.jar builtin/` command.
The `custom_props.jar` file is created.
5. Go to `[OXYGEN_INSTALL_DIR]/lib` and create the `endorsed` folder. Copy the `custom_props.jar` file to `[OXYGEN_INSTALL_DIR]/lib/endorsed`.

Content Completion in CSS Stylesheets

A [Content Completion Assistant \(on page 638\)](#) offers the CSS properties and the values available for each property. It can be manually activated with the **Ctrl + Space** shortcut and is context-sensitive when invoked for the value of a property. The code templates that are proposed include form controls, actions, and **Author** mode operations.

Figure 71. Content Completion in CSS Stylesheets



The properties and values available are dependent on the CSS Profile selected in the [CSS preferences \(on page 160\)](#). 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 Information:

[Specify Custom CSS Properties \(on page 353\)](#)

Syntax Highlighting in CSS Files

Oxygen JSON Editor supports syntax highlighting in **Text** mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for CSS files, follow these steps:

1. Open the **Preferences** dialog box (**Options > Preferences**) (on page 74).
2. Go to **Editor > Syntax Highlight** (on page 151).
3. Select and expand the **CSS** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

Related Information:

[Syntax Highlight Preferences \(on page 151\)](#)

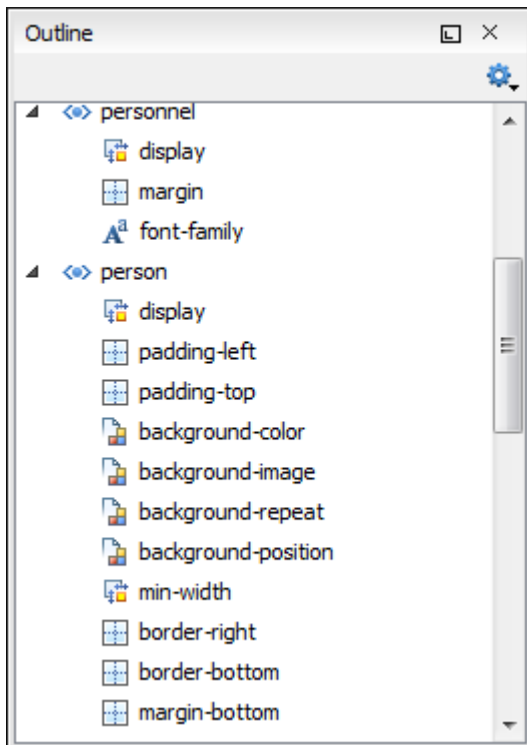
CSS Outline View

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 JSON Editor highlights the corresponding import or selector in the CSS editor.

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

Figure 72. 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 JSON Editor opens a dialog box that allows you to:

- Set the location of the resulting CSS.
- Place each style rule on a new line.

After pressing **OK**, Oxygen JSON Editor performs the following actions:

- All spaces are normalized (all leading and trailing spaces are removed, while sequences of white spaces are replaced with single space characters).
- All comments are removed.



Note:

The CSS minifier relies heavily upon the W3C CSS specification. If the content of the CSS file you are trying to minify does not conform with the specifications, an error dialog box will be displayed, listing all errors encountered during the processing.

The resulting CSS stylesheet gains a lot in terms of execution performance, but loses in terms of readability. The source CSS document is left unaffected.



Note:

To restore the readability of a minified CSS, invoke the **Format and Indent** action from the **Document > Source** menu, the **Source** submenu from the contextual menu, or **Source** toolbar. However, this action will not recover any of the deleted comments.

Editing LESS Stylesheets

Oxygen JSON Editor provides support for stylesheets coded with the LESS dynamic stylesheet language. LESS extends the CSS language by adding features that allow mechanisms such as *variables*, *nesting*, *mixins*, *operators*, and *functions*. Oxygen JSON Editor offers additional LESS-editing 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 \(on page 224\)](#).
- **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 JSON Editor supports syntax highlighting in LESS files, although there may be some limitations in supporting all the LESS constructs.

For more information about LESS go to: <http://lesscss.org/>.

Validating LESS Stylesheets

Oxygen JSON Editor includes a built-in *LESS CSS Validator*, integrated with general validation support. The usual validation features for presenting errors also available for LESS stylesheets.

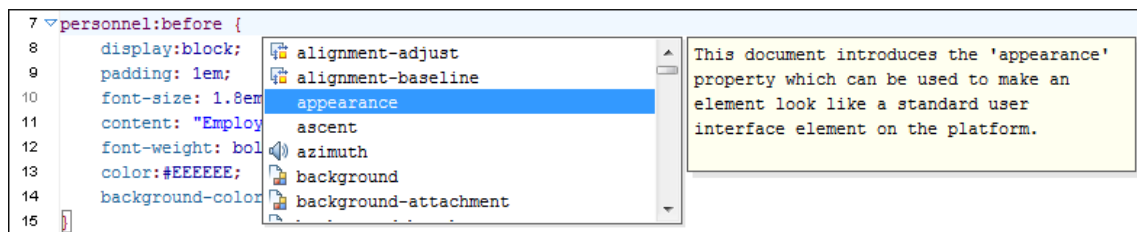
Oxygen JSON Editor provides three validation methods:

- Automatic validation as you type - marks validation errors in the document as you are editing.
- Validation upon request, by pressing the  **Validate** button from the  **Validation** toolbar drop-down menu. An error list is presented in the message panel at the bottom of the editor.
- Validation scenarios, by selecting  **Configure Validation Scenario(s)** from the  **Validation** toolbar drop-down menu. Errors are presented in the message panel at the bottom of the editor. This is useful when you need to validate the current file as part of a larger LESS import hierarchy (for instance, you may change the URL of the file to validate to the root of the hierarchy).

Content Completion in LESS Stylesheets

A *Content Completion Assistant* (on page 638) offers the LESS properties and the values available for each property. It can be manually activated with the **Ctrl + Space** shortcut and is context-sensitive when invoked for the value of a property in a LESS file. The code templates that are proposed include form controls, actions, and **Author** mode operations.

Figure 73. Content Completion in LESS Stylesheets



The properties and values available are dependent on the CSS Profile selected in the [CSS preferences](#) (on page 160).

Syntax Highlighting in LESS Files

Oxygen JSON Editor supports syntax highlighting in **Text** mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for LESS files, follow these steps:

1. Open the **Preferences** dialog box (**Options > Preferences**) (on page 74).
2. Go to **Editor > Syntax Highlight** (on page 151).
3. Select and expand the **LESS** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

Related Information:

[Syntax Highlight Preferences \(on page 151\)](#)

Compiling LESS Stylesheets to CSS

When editing LESS files, you can compile the files into CSS. Oxygen JSON Editor provides both manual and automatic options to compile LESS stylesheets into CSS.

**Important:**

The LESS processor works well only with files having the *UTF-8* encoding. Thus, it is highly recommended that you always use the `utf-8` encoding when working with LESS files or the files they import (other LESS or CSS files). You can use the following directive at the beginning of your files:

```
@charset "utf-8";
```

You have two options for compiling LESS files to CSS:

1. Use the contextual menu in a LESS file and select **Compile to CSS (Ctrl + Shift + C (Command + Shift + C on macOS))**.
2. Select the **Automatically compile LESS to CSS when saving** option ([on page 135](#)) (in the **Save** preferences page). If selected, when you save a LESS file it will automatically be compiled to CSS (this option is deselected by default).

**Important:**

If this option is selected, when you save a LESS file, the CSS file that has the same name as the LESS file is overwritten without warning. Make sure all your changes are made in the LESS file. Do not edit the CSS file directly, as your changes might be lost.

Editing JSON Documents

This section explains the features of the Oxygen JSON Editor and how to use them.

Resources

For more information about the JSON support in Oxygen JSON Editor, see the following resources:

- [Video: JSON Editing](#)
- [Video: JSON Tools in Oxygen](#)
- [Webinar: JSON and JSON Schema Support in Oxygen](#)

JSON Editor

Oxygen JSON Editor includes a specialized JSON editor with various editing features for files that have the `json` file extension. It also includes a document template to help you get started with JSON documents. The template is called **JSON** and it can be found in the **New Document** folder in the **New document wizard** (on [page 225](#)).



Tip:

You can experiment with a sample of a JSON file available at: `[OXYGEN-INSTALL-DIR]/samples/json/personal.json`.

Text Mode Editor

When editing JSON documents in the **Text** editing mode, the usual text editing actions are available, along with other editor-specific actions, including:

- [Search and Find/Replace](#) (on [page 263](#))
- Drag and Drop
- [Validation](#) (on [page 363](#))
- Format and Indent (Pretty Print)

The JSON Text mode editor also offers unique features. For example, the property name is displayed after the ending bracket:

```

        "five.worker"
    ] /subordinates
} /link
}, /person[0]
{
    "id": "one.worker",

```

This default behavior of displaying the property name after the ending bracket can be disabled by deselecting the **Show property name after ending bracket** option (on [page 118](#)) in the **Editor > Edit Modes > Text** preferences page.



Note:

You can run XPath expressions on open 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.

Grid Mode Editor

Oxygen JSON Editor allows you to view and edit the JSON documents in the **Grid** mode. The JSON is represented in **Grid** mode as a compound layout of nested tables and the JSON data and structure can be easily manipulated with table-specific operations or drag and drop operations on the grid components.

Figure 74. JSON Editor Grid Mode

	id	name	family	given	email	link
1	"Big.Boss"	name	"Boss"		"chief@oxyg enxml.com"	link
2	"one.worker"	name	"Worker"	"One"	"one@oxygen xml.com"	link
3	"two.worker"	name	"Worker"	"Two"	"two@oxygen xml.com"	link
4	"three.worker"	name	"Worker"	"Three"	"three@oxyg enxml.com"	link
5	"four.worker"	name	"Worker"	"Four"	"four@oxyge nxml.com"	link
6	"five.worker"	name	"Worker"	"Five"	"five@oxyge nxml.com"	link

You can also use the following JSON-specific contextual actions:

Array

Useful when you want to convert a JSON *value* to *array*.

Insert value before

Inserts a value before the currently selected one.

Insert value after

Inserts a value after the currently selected one.

Append value as child

Appends a value as a child of the currently selected value.

You can [customize the JSON grid appearance \(on page 119\)](#) 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.

Navigating References in JSON Documents

When editing JSON documents (or JSON Schema), you can easily navigate *JSON Pointer* references and hyperlinks by using the **CTRL + Click** shortcut. Holding the **CTRL** key while hovering over a *JSON Pointer* reference or hyperlink will change the reference to a clickable link.

JSON Schema References

Referencing allows you to create modular, maintainable, and reusable schemas. It is particularly useful when you have multiple instances of a similar structure in your data and want to enforce consistency in validation rules.

A schema can reference another schema using the `$ref` (or `$dynamicRef`) keyword. Its value is a URI-reference that is resolved against the schema's [Base URI](#). When evaluating a `$ref`, an implementation uses the resolved identifier to retrieve the referenced schema and applies that schema to the instance. For more details about structuring JSON schemas, see [Understanding JSON Schema: Structuring a complex schema](#).

- **Defining Reusable Schemas** - You can define reusable schema components using the `definitions` (or `$defs` for latest drafts) keyword. These definitions act as named schemas that you can reference using `$ref` (or `$dynamicRef`).

```
{
  "$defs": {
    "person": {
      "type": "object",
      "properties": {
        "name": { "type": "string" },
        "age": { "type": "integer" }
      }
    }
  }
}
(my_schema.json)
```

- **Referencing a Schema** - To reference a schema defined in a definitions block, use the `$ref` keyword followed by the JSON Pointer of the definition.

```
{
  ...
  "type": "object",
  "properties": {
    "person1": { "$ref": "#/$defs/person" },
    "person2": { "$ref": "#/$defs/person" }
  }
}
```

In this example, both `"person1"` and `"person2"` properties refer to the `"person"` schema defined in `$defs`.

- **External References** - You can also reference schemas from external files using their URI. This can be useful when you have multiple schema files.

```
{
  "$ref": "my_schema.json#/$defs/person"
}
```

Here, the schema at the specified URI is being referenced. You can also use `$id` to uniquely identify a schema resource and then reference it from another file using the same mechanism. The following example sets a custom id for the "person" schema:

```
{
  "$defs": {
    "person": {
      "$id": "#person_info",
      "type": "object",
      "properties": {
        "name": { "type": "string" },
        "age": { "type": "integer" }
      }
    }
  }
}
```

You can now reference this definition by its `$id`, not by its JSON Pointer:

```
{
  "$ref": "my_schema.json/#person_info"
}
```



Note:

Oxygen JSON Editor allows `$id` only as an URI fragment, not as a relative pointer.

- **Combining References with Other Keywords** - You can use `$ref` in combination with other keywords. For instance, you might reference a schema within the `items` keyword to define an array of a specific type:

```
{
  ...
  "type": "array",
  "items": { "$ref": "#/$defs/person" }
}
```

Starting with latest drafts, you can use `$ref` in conjunction with other type-specific keywords for stricter validation. For example, the previous schema can be modified to not allow extraneous properties for a "person" object:

```
{
  ...
  "type": "array",
  "items": {
    "$ref": "#/$defs/person",
    "additionalProperties": false
  }
}
```

Validating JSON Documents

Oxygen JSON Editor includes a built-in JSON validator that is used to validate JSON documents against JSON Schemas, as well as a built-in JSON *Well-Formedness* validator (based on the free JAVA source code available at www.json.org). A built-in JSON Schematron Validator engine is also provided to validate JSON documents against a specified Schematron schema.

Resources

For more information, see the following video demonstration:

<https://www.youtube.com/embed/3JEL6nFUozQ>

Checking Well-Formedness in JSON Documents

A *Well-formed* JSON document is a sequence of Unicode code points that strictly conforms to the JSON grammar defined by the [JSON Data Interchange Syntax specification](#). By default, Oxygen JSON Editor automatically checks the document for *Well-formedness* as you type.

Check for Well-Formedness Manually

To manually check documents for *Well-Formedness*:

- Select the  **Check Well-Formedness ()** action from the  ▾ **Validation** drop-down menu on the toolbar or from the menu.
- A selection of files can be checked for well-formedness by selecting the  **Check Well-Formedness** action from the **Validate** submenu when invoking the contextual menu in the [Project view \(on page 251\)](#).

Result: If any errors are found, the result is displayed in the message panel at the bottom of the editor. Each error is displayed as one record in the result list and is accompanied by an error message. Clicking the record will open the document containing the error and highlight its approximate location.

Example: A non *Well-formed* JSON Document

```
{"person": { "name": "John Doe" }
```

This would result in the following error:

```
Expected a ', ' or '}'
```

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, you would need to identify where the missing end bracket needs to be placed.

Validating JSON Documents Against JSON Schema or Schematron

A *valid* JSON document is a *well-formed* document that also conforms to the rules of a JSON Schema that defines the legal syntax of a JSON document. The purpose of the JSON schema is to define the legal properties and values of a JSON document.

This section contains topics that explain the automatic and manual validation possibilities in Oxygen JSON Editor, how validation errors are presented, and information about built-in validation scenarios.

Oxygen JSON Editor also includes a built-in JSON Schematron Validator engine to validate JSON documents against a Schematron schema specified in a custom validation scenario or using the [Validate with action \(on page 365\)](#).



Tip:

Inside the samples folder, there are a few files you can use to see how Schematron validation can be done with JSON files. The path of the folder containing these sample files is:

`[OXYGEN_INSTALL_DIR]/samples/json/schematron/`.

For information about how to associate a schema for the purposes of validation, see [Associating a JSON Schema Through a Validation Scenario \(on page 373\)](#).

Automatic Validation

By default, Oxygen JSON Editor is configured to automatically mark validation errors in the JSON document as you are editing. The [Enable automatic validation option \(on page 153\)](#) in the **Document Checking** preferences [page \(on page 153\)](#) controls whether or not all validation errors and warnings will automatically be highlighted in the editor pane.

The automatic validation starts parsing the document and marking the errors after a [configurable delay \(on page 153\)](#) from the last typed key. Errors are highlighted with underline markers in the main editor pane and small rectangles on the right side ruler. Hovering over a validation error presents a tooltip message with more details about the error.

If the error message is too long to be 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 itself, displays an information dialog box with the full error message. You can use the arrow buttons in this dialog box to navigate through the errors issued by the automatic validation feature.

Related Information:[Manual Validation Actions \(on page 365\)](#)[Presenting Validation Errors in JSON Documents \(on page 365\)](#)

Manual Validation Actions

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

Manual Validation Actions

To manually validate the currently edited document, use one of the following actions:

Validate

Available from the  ▾ **Validation** drop-down menu on the toolbar, the **Document > Validate** menu, or from the **Validate** submenu when invoking the contextual menu on one or more JSON documents in the **Project view** ([on page 251](#)).

Validate with

Available from the  ▾ **Validation** drop-down menu on the toolbar or the **Document > Validate** menu. This action opens a dialog box that allows you to [specify a schema for validating the current document \(on page 374\)](#).

Validate with Schema

Available from the **Validate** submenu when invoking the contextual menu on one or more JSON documents in the **Project view** ([on page 251](#)). This action opens a dialog box that allows you to specify a JSON or Schematron schema to be used for the validation.

Other Validation Options

**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 \(on page 153\)](#).

Related Information:[Automatic Validation \(on page 364\)](#)[Presenting Validation Errors in JSON Documents \(on page 365\)](#)

Presenting Validation Errors in JSON Documents

Validation errors and warnings in JSON documents are presented in various locations within the interface.

Validation Marker Locations

Validation issues are marked in the following locations:

- In the main editing pane, with the issue underlined in a color according to the type of issue.
- In the right-side vertical stripe, with a marker that is colored according to the type of issue.
- In the **Outline** view, with an icon that is colored according to the type of issue.

Validation Marker Colors

The colors for each type of issue are as follows:

- **Validation Errors** [Red] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the **Attributes** view are colored in red.
- **Validation Warnings** [Yellow] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the **Attributes** view are colored in yellow.
- **Validation Info** [Blue] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the **Attributes** view are colored in blue.

You can configure the color for each type in the [Document Checking preferences page \(on page 153\)](#).

Validation Markers in the Right-Side Stripe

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

Upper Part of the Stripe

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

Middle Part of the Stripe

Errors are presented with red markers, warnings with yellow markers, and info message with blue markers. If you want to limit the number of markers that are displayed, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#), go to **Editor > Document checking**, and specify the desired limit in the **Maximum number of validation highlights** option [\(on page 153\)](#).

Clicking a marker will highlight the corresponding text area in the editor. The validation message is also displayed both in a tooltip (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 \(on page 153\)](#).

Bottom Part of the Stripe

Two navigation arrows () can be used to jump to the next or previous issue. The same actions can be triggered from **Document > Automatic validation > Next Error/Highlight (Ctrl + Period (Command + Period on macOS))** and **Document > Automatic validation > Previous Error/Highlight (Ctrl + Comma (Command + Comma on macOS))**. Also, the  **Remove All** button can be used to clear all the validation markers.

Hovering Over Validation Issues

Hovering over a validation issue presents a tooltip message with more details about the problem. Also, when hovering over an issue, pressing **F2** will change the focus to the tooltip.

Details About Validation Issues

- Information about the issue is also displayed in the message area on the bottom of the editor panel (clicking the  **Document checking options** button opens the [Document Checking preferences page \(on page 153\)](#) where you can configure some validation options (such as the colors used to present the validation issues). Some validation messages have an icon () and clicking it opens a dialog box with additional information and a link to specifications.
- If you want to see all the validation messages grouped in the **Results** view, use the  **Validate** action from the toolbar or **Document > Validate** menu. To see more information about a validation message, right-click the item in the **Results** view and select **Show message**. Some validation messages have an icon () in the **Info** column and clicking it opens a dialog box with additional information and a link to specifications.

Creating a JSON Validation Scenario

Validation scenarios can be used to [associate one or more JSON Schemas with a JSON document \(on page 373\)](#). Oxygen JSON Editor also includes a built-in JSON Schematron Validator engine that can be specified in the validation scenario to validate JSON documents against a specified Schematron schema.

Creating a JSON Validation Scenario

To create a validation scenario, follow these steps:

1. Select the  **Configure Validation Scenario(s)** action in one of the following ways:
 - From the  **Validation** toolbar drop-down menu.
 - From the **Document > Validate** menu.
 - From the **Validate** submenu, when invoking the contextual menu on a file in the [Project view \(on page 251\)](#).

Step Result: The **Configure Validation Scenario(s)** dialog box is displayed.

2. Click the **New** button.

Step Result: A validation scenario configuration dialog box is displayed.

Figure 75. Validation Scenario Configuration Dialog Box

New scenario

Name

Storage: ☒ Project Options ☐ Global Options

URL of the file to validate	File type	Validation engine	Automatic validation	Schema
<code>\${currentFileURL}</code>	JSON Document	<Default engine>	☒	<No JSON schem...

This scenario configuration dialog box allows you to configure the following information and options:

Name

The name of the validation scenario.

Storage

You can choose between storing the scenario in the **Project Options** ([on page 640](#)) or **Global Options** ([on page 639](#)).

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](#) ([on page 197](#)).

Figure 76. Insert an Editor Variable

<code>\${_Desktop}</code>	- My Desktop
<code>\${start-dir}</code>	- <u>S</u> tart directory of custom validator
<code>\${standard-params}</code>	- <u>L</u> ist of standard params for command line
<code>\${cfn}</code>	- <u>T</u> he current file name without extension
<code>\${currentFileURL}</code>	- The path of the currently edited file (<u>U</u> RL)
<code>\${cfdu}</code>	- The path of current file directory (<u>U</u> RL)
<code>\${frameworks}</code>	- <u>O</u> xxygen frameworks directory (<u>U</u> RL)
<code>\${pdu}</code>	- <u>P</u> roject directory (<u>U</u> RL)
<code>\${oxygenHome}</code>	- Oxygen installation directory (<u>U</u> RL)
<code>\${home}</code>	- The path to user home directory (<u>U</u> RL)
<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

File type

The type of the document that is validated in the current validation unit. Oxygen JSON Editor automatically selects the file type depending on the value of the **URL of the file to validate** field.

Validation engine

You can choose between the following types of validation engines for validating JSON documents:

- **Default engine** - The built-in JSON Validator will be used. For JSON Schema documents, this type should not be chosen unless the document has a schema version specified.
- **JSON Schema Validator** - This type is for JSON Schema documents only. It will use the version specified in the JSON Schema, or if a version is not specified, the [JSON Schema draft-04](#) will be used.
- **JSON Schematron Validator** - The built-in JSON Schematron Validator will be used to validation JSON documents against a specified Schematron schema.

**Note:**

For proper error localization, the root element of the Schematron schema should include the `@queryBinding` attribute with the value of **xslt2** after the Schematron namespace declaration:

```
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" query
Binding="xslt2">
```

Automatic validation

If this option is selected, the validation operation defined by this row is also applied by the automatic validation feature. If the **Automatic validation** feature is disabled in the [Document Checking preferences page \(on page 153\)](#), then this option is ignored, as the preference setting has a higher priority.

Schema

Displays the specified schema.

Specify Schema

Opens the **Specify Schema** dialog box that allows you to set a schema to be used for validating JSON documents.

Move Up

Moves the selected scenario up one spot in the list.

Move Down

Moves the selected scenario down one spot in the list.

Add

Adds a new validation unit to the list.

Remove

Removes an existing validation unit from the list.

3. Configure any of the existing validation units according to the information above. You can use the buttons at the bottom of the table to add, remove, or move validation units.
4. Click **OK**.

Result: 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 click the **Apply associated** button to run the validation scenario.

Sharing JSON Validation Scenarios

The validation scenarios and their settings can be shared with other users by saving them at [project level \(on page 640\)](#) or by [exporting them to a specialized scenarios file \(on page 197\)](#) that can then be imported. When you create a new validation scenario or edit an existing one, there is a **Storage** option to control whether the scenarios are stored in **Project Options (on page 640)** or **Global Options (on page 639)**.

Storage: ☒ Project Options ☐ Global Options

Selecting **Project Options (on page 640)** stores the scenario in the project file and can be shared with other users that have access to the project. If your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) then your team will have access to the scenarios that you define. When you create a scenario at the project level, the URLs from the scenario become relative to the project URL.

Selecting **Global Options** (on page 639) stores the scenario in the global options that are stored in the user home directory.

You can also change the storage options of existing validation scenarios by using the **Change storage** action from the contextual menu of the list of scenarios in the **Configure Validation Scenario(s)** dialog box.

Resolving References with an XML Catalog

If a reference to a remote JSON schema must be used but a local copy of the schema should actually be preferred for performance reasons, the reference can be resolved to the local copy with an *XML Catalog* (on page 640).

For example, if the JSON schema contains a reference to a remote schema such as:

```
{"$ref": "http://json-schema.org/example/geo.json"}
```

the reference can be resolved to a local copy of the schema by inserting the following catalog entry:

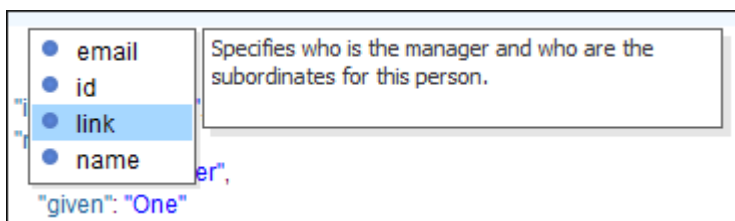
```
<uri name="http://json-schema.org/example/geo.json" uri="schemas/geo.json"/>
```

Content Completion Assistant in JSON

Oxygen JSON Editor includes an intelligent *Content Completion Assistant* (on page 638) that offers proposals for inserting JSON structures that are valid at the current editing location.

The *Content Completion Assistant* is enabled by default. To disable it, open the **Preferences** dialog box (**Options > Preferences**) (on page 74), go to **Editor > Content Completion**, and deselect the **Enable content completion** option (on page 139).

Figure 77. Content Completion Assistant in JSON



Content Completion and the Associated Schema

The *Content Completion Assistant* feature is schema-driven and the list of proposals in the *Content Completion Assistant* (on page 638) depend on the associated JSON Schema. For information about ways to associate a schema to a JSON document, see the *Associating a Schema to JSON Documents* (on page 373) section.

Using the Content Completion Assistant in JSON

The feature is activated in **Text** mode for JSON documents by:

- Typing a quote symbol (") to insert a property or value.
- Pressing **Ctrl + Space** or **Alt + ForwardSlash (Command + Option + ForwardSlash on macOS)**.

You can navigate through the list of proposals by using the **Up** and **Down** keys on your keyboard. In some cases, the *Content Completion Assistant* displays a [documentation window with information about the particular proposal \(on page 372\)](#). You can also change the size of the documentation window by dragging its top, right, and bottom borders.

To insert the selected proposal, press **Enter** or **Tab**.

Types of Proposals Listed in the Content Completion Assistant for JSON

The proposals that populate the *Content Completion Assistant* for JSON documents depend on the structure defined in the associated JSON Schema. The types of structure proposed in the content completion window include:

- JSON properties
- JSON values
- JSON arrays
- JSON objects

The number and type of proposals displayed by the *Content Completion Assistant* is dependent on the cursor's current position in the JSON document and the child items displayed within a given context are defined by the structure of the specified JSON Schema.

Code Templates in the Content Completion

Oxygen JSON Editor includes a set of built-in code templates for JSON documents that can be selected from the *Content Completion Assistant*. The code templates are displayed with a  symbol in the content completion list. You can also define your own code templates and share them with others.

Schema Annotations in JSON Content Completion

A schema annotation is a documentation snippet that appears in the *Content Completion Assistant (on page 638)* offering more information about the current proposal.

This feature is enabled by default, but you can disable it by deselecting the **Show annotations in Content Completion Assistant (on page 143)** option in the **Annotations** preferences page.

Collecting Annotations from the JSON Schema

In a JSON Schema, the annotations are specified in the value of the `title` and `description` properties like this:

```
"idType": {
  "title": "The 'id' property",
  "description": "Specifies a required ID for this person.",
  "type": "string",
```



```
"maxLength": 20
}
```

Associating a Schema to JSON Documents

To provide as-you-type validation and to compute valid proposals for the *Content Completion Assistant* (on page 638), Oxygen JSON Editor requires a schema to be associated with the JSON document. The schema specifies how the internal structure is defined.

Detecting the Schema(s) for Validation and Content Completion

For validation, Oxygen JSON Editor tries to detect the JSON Schema by searching in the following order:

1. The schema [referenced in validation stages from the validation scenario\(s\)](#) (on page 373) associated with the current JSON document.
2. If a schema is not detected, then it falls back to the [schema associated directly in the JSON document](#) (on page 376).



Tip:

To quickly open the schema used for validating the current document, select the **Open Associated Schema** action from the toolbar (or **Document > Schema** menu).

Associating a JSON Schema Through a Validation Scenario

Oxygen JSON Editor uses the rules defined in the detected schema to report errors and warnings during automatic and manual validations that help maintain the structural integrity of your JSON documents. Oxygen JSON Editor includes built-in validation engines for validating JSON documents against a JSON Schema or Schematron schema. There are several methods that can be used to validate JSON document with a schema.

Configure a Validation Scenario and Specify the Schema

You can specify the schema to be used for validation directly in the [JSON validation scenario](#) (on page 367).

To associate a schema to a validation scenario to be used whenever the scenario is invoked, follow these steps:

1. Select the **Configure Validation Scenario(s)** from the **Validation** toolbar drop-down menu, from the **Document > Validate** menu, or from the **Validate** submenu when invoking the contextual menu on a JSON file in the **Project view** (on page 251).
2. Click the **New** button to [create a new validation scenario](#) (on page 367) or the **Edit** button to modify an existing one.
3. Add or configure validation units according to your needs. For details about all of the configuration options, see [Creating a JSON Validation Scenario](#) (on page 367).
4. Click the **Specify Schema** button to select the schema to be associated with the validation unit.
5. Click **OK** on both dialog boxes.

Result: The schema is now associated with that validation scenario whenever it is invoked.

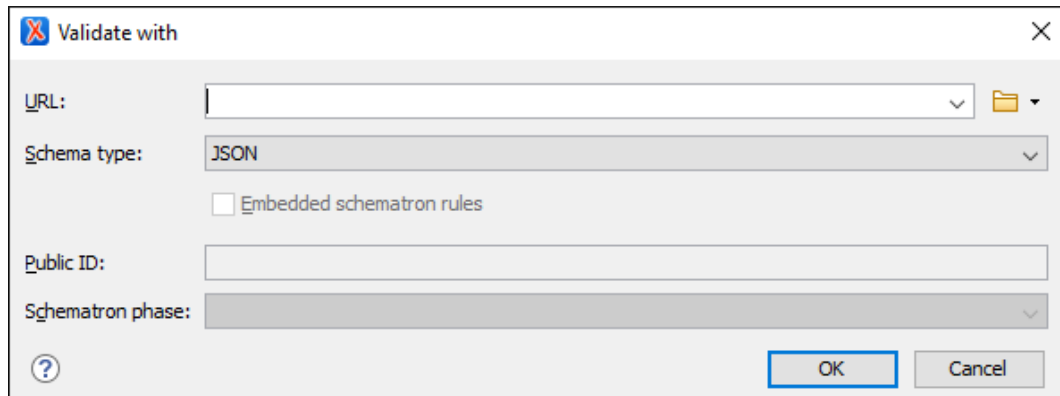
Use the Validate with Action to Specify a Schema for Validating the Current Document

To validate the current document using a specified schema, follow these steps:

1. Select the **Validation with** action from the  **Validation** drop-down menu on the toolbar (or **Document > Validate** menu).

Step Result: The **Validate with** dialog box is displayed:

Figure 78. Validate with Dialog Box



The dialog box titled "Validate with" has a close button (X) in the top right corner. It contains the following fields and controls:

- URL:** A text input field with a history drop-down arrow and a folder icon for browsing.
- Schema type:** A dropdown menu currently set to "JSON".
- Embedded schematron rules:** An unchecked checkbox.
- Public ID:** A text input field.
- Schematron phase:** A dropdown menu.
- Buttons:** "OK" and "Cancel" buttons at the bottom right, with a help icon (?) at the bottom left.

This dialog box contains the following options:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S). You can specify the URL by using the text field, the history drop-down, the  **Insert Editor Variables** (on page 197) button, or the browsing actions in the  **Browse** drop-down list.
- **Schema type** - You can select one of the following two types (other types of schema will not work with JSON documents):
 - **JSON** - Used for validating JSON documents against a specified JSON Schema.
 - **Schematron** - Used for validating JSON documents against a specified Schematron schema. You can also select a **Schematron phase** that you want to use for the validation.



Note:

For proper error localization, the root element of the Schematron schema should include the `@queryBinding` attribute with the value of **xslt2** after the Schematron namespace declaration:

```
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding
="xslt2">
```

2. Select the schema to be associated with the manual validation.
3. Click **OK**.

Result: The current document is validated using the schema you specified.

Use the Validate with Schema Action to Specify a Schema for Validating all Selected JSON Documents

To validate multiple JSON documents using a specified schema, follow these steps:

1. Select all the JSON documents you want to validate in the **Project** view.
2. Invoke the contextual menu (right-click) and select the **Validate with Schema** action from the **Validate** submenu.

Step Result: The **Validate with** dialog box is displayed:

Figure 79. Validate with Dialog Box

This dialog box contains the following options:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S). You can specify the URL by using the text field, the history drop-down, the **Insert Editor Variables** (on page 197) button, or the browsing actions in the **Browse** drop-down list.
- **Schema type** - You can select one of the following two types (other types of schema will not work with JSON documents):
 - **JSON** - Used for validating JSON documents against a specified JSON Schema.
 - **Schematron** - Used for validating JSON documents against a specified Schematron schema. You can also select a **Schematron phase** that you want to use for the validation.



Note:

For proper error localization, the root element of the Schematron schema should include the `@queryBinding` attribute with the value of **xslt2** after the Schematron namespace declaration:

```
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2">
```

3. Select the JSON schema that you want to use to validate all selected JSON documents.
4. Click **OK**.

Result: The selected JSON documents are validated using the JSON schema you specified.

Associating a JSON Schema Directly in JSON Documents

Associate Schema Action

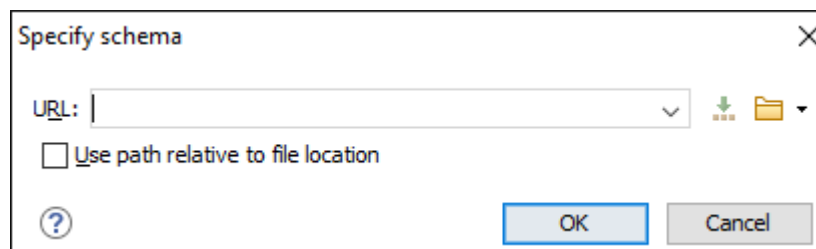
The schema used by the [Content Completion Assistant \(on page 638\)](#) and document validation engine can be associated with the current document by using the  **Associate Schema** action. The association can specify a relative file path or a URL of the schema.

To associate a JSON Schema to the current JSON document, follow these steps:

1. Select the  **Associate Schema** action from the toolbar (or **Document > Schema** menu).

Step Result: The **Associate Schema** dialog box is displayed:

Figure 80. Associate Schema Dialog Box



This dialog box contains the following options for JSON documents:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S).
 - **Use path relative to file location** - Select this option if the JSON instance document and the associated schema contain relative paths. The location of the schema file is inserted in the JSON 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.
2. Select the JSON Schema that will be associated with the JSON document.
 3. Click **OK**.

Result: A `$schema` property is added at the beginning of the document with its value set to the specified URL. If the document already contained a schema association, the old association is replaced with the new one.



Tip:

To quickly open the schema used for validating the current document, select the  **Open Associated Schema** action from the toolbar (or **Document > Schema** menu).

Associating a JSON Schema in a Framework (Document Type) Configuration

The JSON schema used to compute valid proposals in the *Content Completion Assistant* (on page 638) and by the document validation engine to report errors and warnings can be defined in each particular *framework* (on page 639) (document type). This schema will be used only if one is not detected in the current JSON file (on page 376).

To associate a JSON schema in a particular *framework* (document type), follow these steps:

1. Open the **Preferences** dialog box (**Options > Preferences**) (on page 74) and go to **Document Type Association**.
2. Select your particular document type and click the **Edit** (on page 87), **Extend** (on page 88), or **Duplicate** (on page 87) button to modify an existing *framework* (or use the **New** button to create a new one).

Step Result: This opens a **Document type** configuration dialog box (on page 89).

3. Go to the **Schema** tab (on page 92).
4. Select the schema type and its URI.
5. Click **OK**.

Result: The schema is now associated with the particular document type and will be used by the *Content Completion Assistant* and validation engine if a schema is not detected in the current JSON document.

Learn Document Structure When JSON Schema is not Detected

When there is no JSON schema associated with a JSON document, Oxygen JSON Editor can learn the document structure by parsing the document internally to provide an initialization source for content completion and validation.

This feature is controlled by the **Learn on open document** option (on page 139) that is available in the **Editor > Content Completion** preferences page. This feature enabled by default. If you want to disable it, deselect the **Learn on open document** option.

You can choose to create a schema from the learned structure and save it as a file using the **Learn Structure** and **Save Structure** actions.

Creating a JSON Schema from Learned Document Structure

To create a JSON schema from the learned structure of a JSON document, follow these steps:

1. Open the JSON document that will be used to create the schema.
2. Go to **Document > JSON Document > Learn Structure** (**Ctrl + Shift + L** (**Command + Shift + L** on macOS)). The **Learn Structure** action reads the mark-up structure of the current document. A **Learn completed** message is displayed in the application status bar when the action is finished.
3. Go to **Document > JSON Document > Save Structure** and enter the file path for the JSON schema.
4. Click the **Save** button.

Syntax Highlighting in JSON Documents

Oxygen JSON Editor supports syntax highlighting in **Text** mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for JSON files, follow these steps:

1. Open the **Preferences** dialog box (**Options > Preferences**) *(on page 74)*.
2. Go to **Editor > Syntax Highlight** *(on page 151)*.
3. Select and expand the **JSON** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

Related Information:

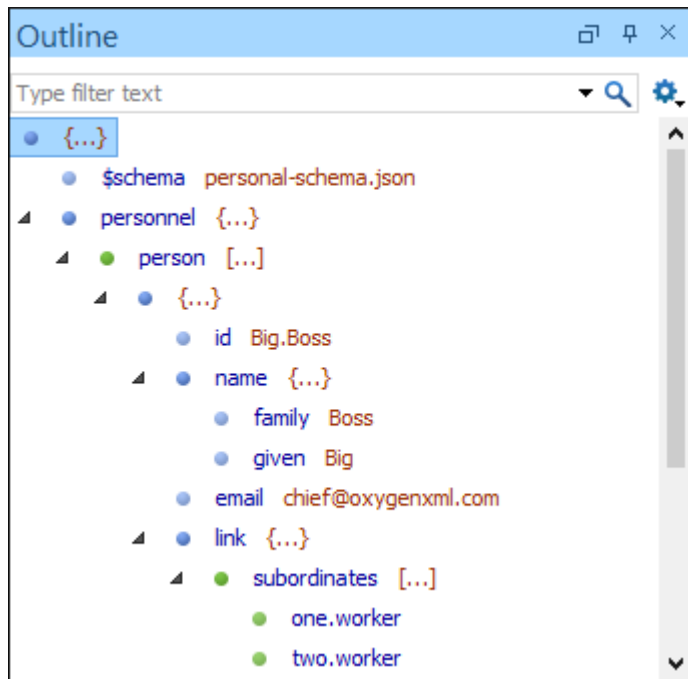
[Syntax Highlight Preferences](#) *(on page 151)*

Folding in JSON

In a large JSON document, the data enclosed in the curly bracket characters `{ }` can be collapsed so that only the needed data remains in focus. The folding features available for XML documents are also available in JSON documents.

JSON Outline View

The **Outline** view for JSON documents displays the list of all the components of the JSON document you are editing. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

Figure 81. JSON Outline View

Outline View Features

The **Outline** view allows you to:

- Quickly navigate through the document by selecting nodes in the **Outline** tree.
- Move elements by dragging them to a new position in the tree structure.
- Highlight elements in the editor area. It is synchronized with the editor area, so when you make a selection in the editor area, the corresponding nodes are highlighted in the **Outline** view, and vice versa.
- View document errors, as they are highlighted in the **Outline** view. A tooltip also provides more information about the nature of the error when you hover over the faulted element.

Outline View Interface

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (such as `*` or `?`) and separate multiple patterns with commas.

It also includes a  **Settings** menu in the top-right corner that presents the following options to help you filter the view even further.

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.

Drag and Drop Actions in the Outline View

Entire JSON properties, objects, and arrays 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 a JSON node within the **Outline** view and drop it on another node, then the dragged node will be moved after the drop target.
- If you hold the mouse pointer over the drop target for a short time before the drop then the drop target node will be expanded first and the dragged node will be moved inside the drop target.
- You can also drop a node before or after another node if you hold the mouse pointer towards the upper or lower part of the target. A marker will indicate whether the drop will be performed before or after the target node.
- If you hold down the **Ctrl (Command on macOS)** key after dragging, a copy operation will be performed instead of a move.

Contextual Menu Actions

The following actions are available in the contextual menu of the JSON **Outline** view:



Cut

Cuts the currently selected component.



Copy

Copies the currently selected component.



Paste

Pastes the copied 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.

JSON to XML Converter

Online JSON to XML Converter



Attention:

For a simple ONLINE tool for converting a single JSON file to XML, or vice versa, go to: https://www.oxygenxml.com/xml_json_converter.html.

Converting JSON to XML in Oxygen

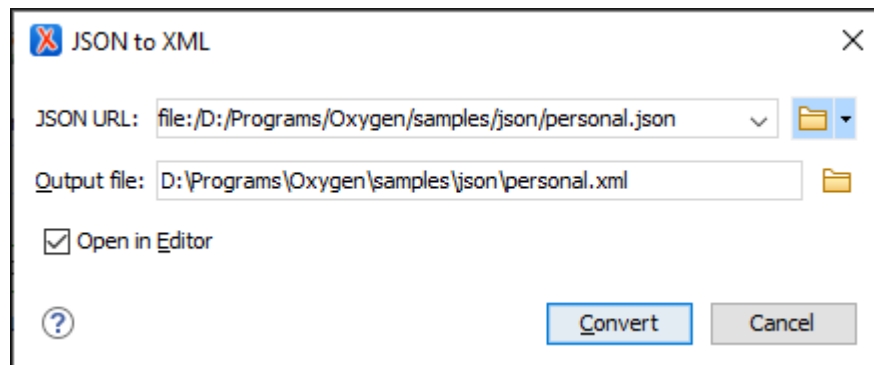
Oxygen JSON Editor includes a useful and simple tool for converting JSON files to XML. The **JSON to XML** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

To convert a JSON document to XML, follow these steps:

1. Select the **JSON to XML** action from the **Tools > JSON Tools** menu.

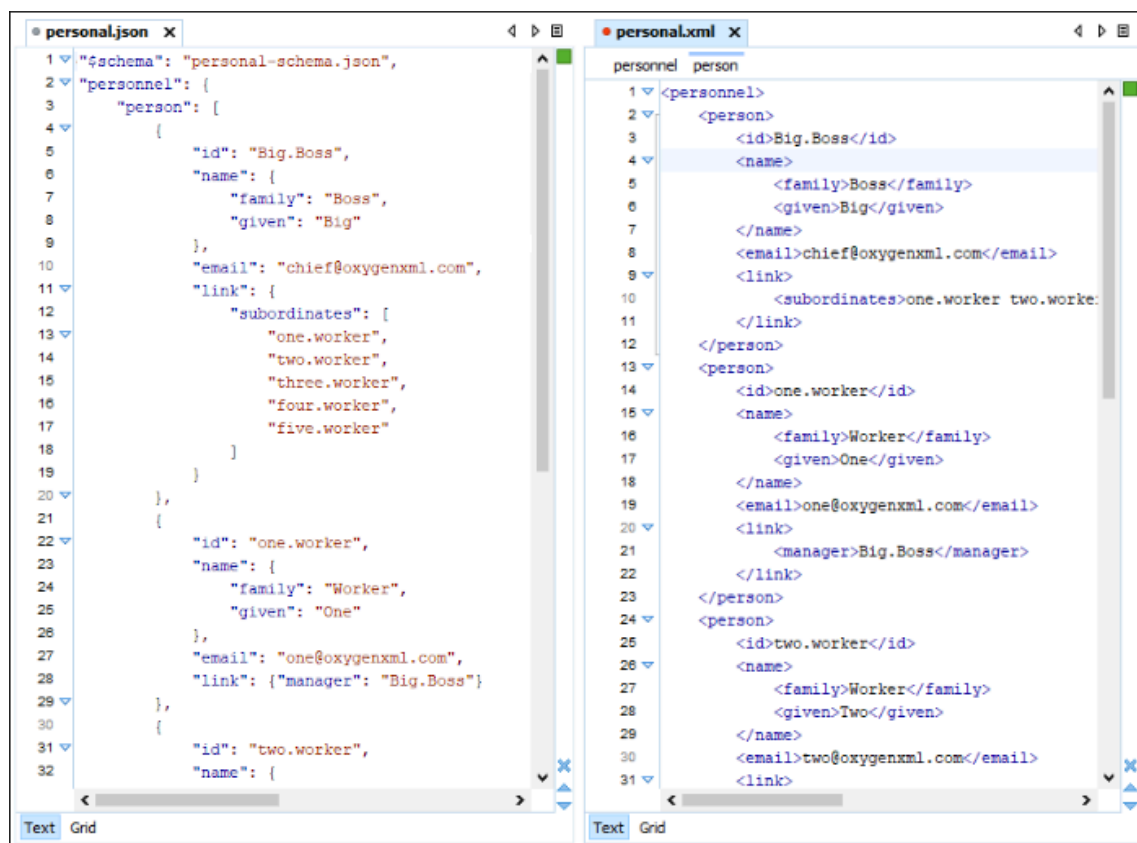
The **JSON to XML** dialog box is displayed:

Figure 82. JSON to XML Dialog Box



2. Choose or enter the **Input URL** of the JSON document.
3. Choose the path of the **Output file** that will contain the resulting XML document.
4. Select the **Open in Editor** option to open the resulting XML document in the main editing pane.
5. Click the **Convert** button.

Result: The original JSON document is now converted to an XML document.

Figure 83. Example: XML to JSON Operation Result

Conversion Details

- If the JSON document has more than one property on the first level, the converted XML document will have an additional root element called `<JSON>`.

For example, the following JSON document:

```
{
  "personnel": {
    "person": [
      {"name": "Boss"},
      {"name": "Worker"}
    ]
  },
  "id": "personnel-id"
}
```

it is converted to:

```
<?xml version="1.0" encoding="UTF-8"?>
<JSON>
  <personnel>
    <person>
      <name>Boss</name>
```

```

    </person>
  <person>
    <name>Worker</name>
  </person>
</personnel>
<id>personnel-id</id>
</JSON>

```

- If the JSON document is an array, the converted XML document will have a root element called `<array>` and for each item within the array, another `<array>` is created.

```

[
  { "name": "Boss" },
  { "name": "Worker" }
]

```

it is converted to:

```

<?xml version="1.0" encoding="UTF-8"?>
<array>
  <array>
    <name>Boss</name>
  </array>
  <array>
    <name>Worker</name>
  </array>
</array>

```

- If the name of a JSON property contains characters that are not valid in XML element names (for example, \$), then the invalid characters will be escaped as its hexadecimal equivalent in the converted XML.

```

{"$id": "personnel-id"}

```

is converted to:

```

<_X24_id>personnel-id</_X24_id>

```

Related Information:

[XML to JSON Converter \(on page 384\)](#)

XML to JSON Converter

Online XML to JSON Converter

**Attention:**

For a simple **ONLINE** tool for converting a single XML file to JSON, or vice versa, go to: https://www.oxygenxml.com/xml_json_converter.html.

Converting XML to JSON in Oxygen

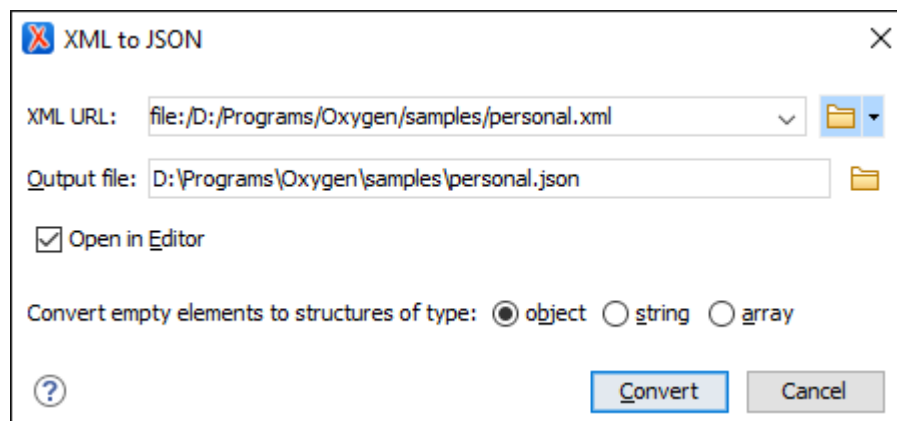
Oxygen JSON Editor includes a useful and simple tool for converting XML files to JSON. The **XML to JSON** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

To convert an XML document to JSON, follow these steps:

1. Select the **XML to JSON** action from the **Tools > JSON Tools** menu.

Step Result: The **XML to JSON** dialog box is displayed:

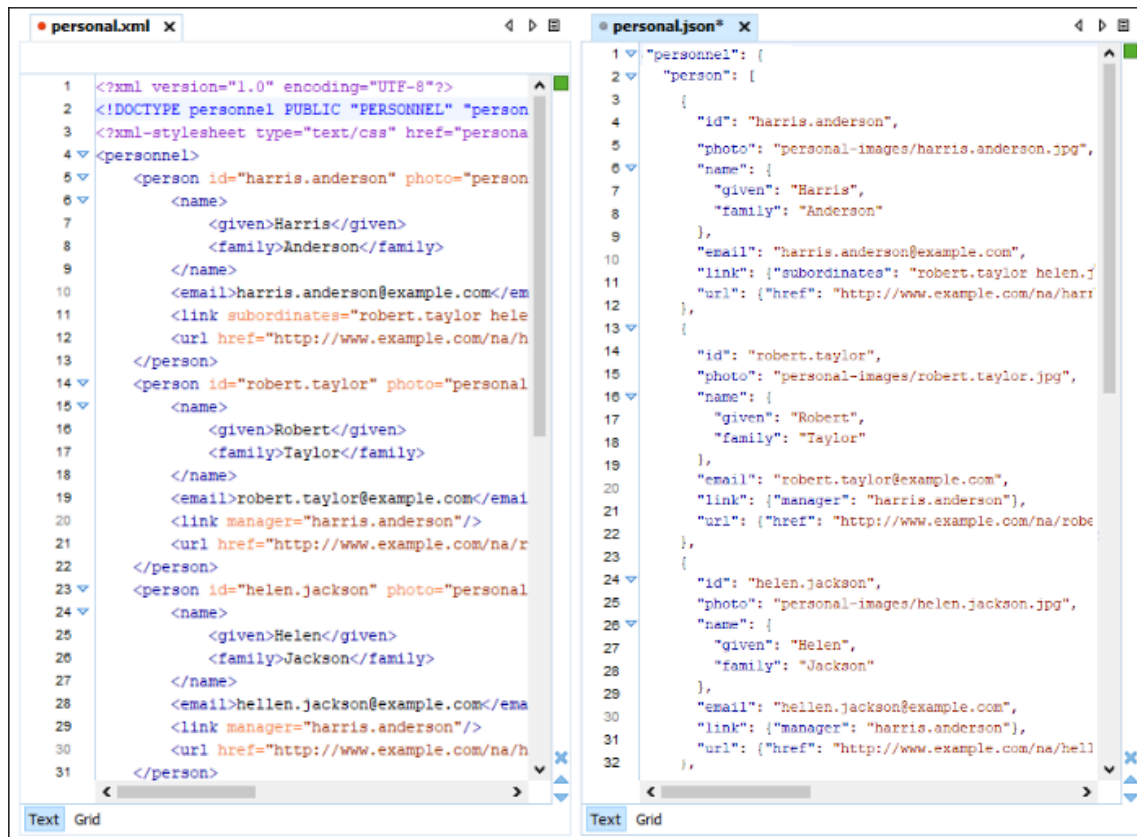
Figure 84. XML to JSON Dialog Box



2. Choose or enter the **Input URL** of the XML document.
3. Choose the path of the **Output file** that will contain the resulting JSON document.
4. Select how you want empty elements to be converted (default is **object**).
5. Select the **Open in Editor** option to open the resulting JSON document in the main editing pane.
6. Click the **Convert** button.

Result: The original XML document is now converted to a JSON document.

Figure 85. Example: XML to JSON Operation Result



Conversion Details

- Some XML components are ignored (e.g. comments and processing instructions).
- If any elements contain attributes in the XML document, the attributes are converted to properties in the converted JSON document. If the XML document contains more than one element with the same name, they will be converted into an array of object in the converted JSON document.

For example, the following XML document:

```
<personnel>
  <person id="person.one">
    <name>Boss</name>
  </person>
  <person id="person.two">
    <name>Worker</name>
  </person>
</personnel>
```

it is converted to:

```
{
  "personnel": {
    "person": [
      {
```

```

    "id": "person.one",
    "name": "Boss"
  },
  {
    "id": "person.two",
    "name": "Worker"
  }
]
}
}

```

- If the XML document contains elements with mixed content (text plus elements), the converted JSON document will contain a `#text` property with its value set as the text content. If there are multiple text nodes, the subsequent `#text` properties will contain a number (e.g. `#text1`, `#text2`). If there are multiple elements with the same name, the first property will have the element name and the subsequent properties will contain a number (e.g. `b`, `b#1`, `b#2`).

```
<p>This <b>is</b> an <b>example</b>!</p>
```

is converted to:

```

{
  "p": {
    "#text": "This ",
    "b": "is",
    "#text1": " an ",
    "b#1": "example",
    "#text2": "!"
  }
}

```

- If the XML document contains element names that contains hexadecimal codes (for example, if they were escaped during a [JSON to XML conversion \(on page 381\)](#)), it will be converted to the normal character value in the converted JSON document.

```
<_X24_id>personnel-id</_X24_id>
```

is converted to:

```
{"$id": "personnel-id"}
```

Related Information:

[JSON to XML Converter \(on page 381\)](#)

JSON to YAML Converter

Converting JSON to YAML in Oxygen

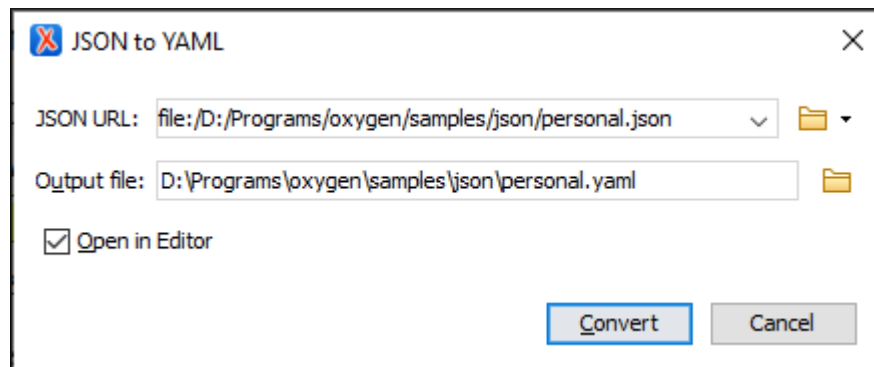
Oxygen JSON Editor includes a useful and simple tool for converting JSON files to YAML. The **JSON to YAML** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

To convert a JSON document to YAML, follow these steps:

1. Select the **JSON to YAML** action from the **Tools > JSON Tools** menu.

The **JSON to YAML** dialog box is displayed:

Figure 86. JSON to YAML Dialog Box



2. Choose or enter the **JSON URL** for the document you want to convert.
3. Choose the path of the **Output file** that will contain the resulting YAML document.
4. **[Optional]** Select the **Open in Editor** option to open the resulting YAML document in the main editing pane.
5. Click the **Convert** button.

Result: The original JSON document is now converted to a YAML document.

Related Information:

[YAML to JSON Converter \(on page 387\)](#)

YAML to JSON Converter

Converting YAML to JSON in Oxygen

Oxygen JSON Editor includes a useful and simple tool for converting YAML files to JSON. It even works on files that consist of multiple YAML documents, each separated by three dashes (---), in which case the conversion creates multiple JSON files with a number in the name.

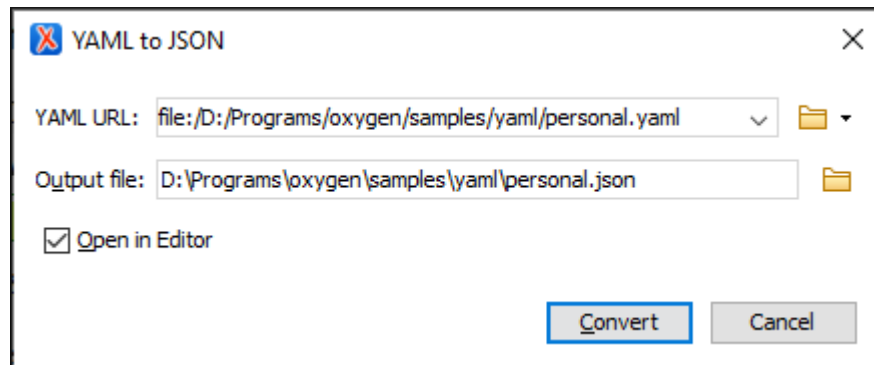
The **YAML to JSON** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

To convert a YAML document to JSON, follow these steps:

1. Select the **YAML to JSON** action from the **Tools > JSON Tools** menu.

The **YAML to JSON** dialog box is displayed:

Figure 87. YAML to JSON Dialog Box



2. Choose or enter the **YAML URL** for the document you want to convert.
3. Choose the path of the **Output file** that will contain the resulting JSON document.
4. **[Optional]** Select the **Open in Editor** option to open the resulting JSON document in the main editing pane.
5. Click the **Convert** button.

Result: The original YAML document is now converted to a JSON document.

Related Information:

[JSON to YAML Converter \(on page 387\)](#)

Contextual Menu Actions in JSON Documents

When editing JSON documents, Oxygen JSON Editor provides the following actions in the contextual menu:

 **Cut**,  **Copy**,  **Paste**

Executes the typical editing actions on the currently selected content.

Copy JSON Pointer

Creates a *JSON Pointer* at the current cursor location and copies the expression that denotes the JSON pointer to the system clipboard.

Copy XPath

Copies the XPath expression of the current property from the current editor to the clipboard.

Toggle Line Wrap (**Ctrl + Shift + Y** (**Command + Shift + Y** on macOS))

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.

Go to Matching Bracket

Moves the cursor to the end bracket that matches the start bracket, or vice versa.

Source submenu

This submenu includes the following actions:

To Lower Case

Converts the content selection to lower case characters. This works with contiguous and multiple selections.

To Upper Case

Converts the selected content to upper case characters. This works with contiguous and multiple selections.

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 ()

Converts a sequence of hexadecimal characters to the corresponding [Unicode character \(on page 303\)](#). 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 and the characters they will be converted to:

- `0x0045` will be converted to `É`
- `0X0125` to `ĥ`
- `265` to `ı`
- `2190` to `☹`



Note:

For more information about finding the hexadecimal value of a character, see [Finding the Decimal, Hexadecimal, or Character Entity Equivalent \(on page 306\)](#).

Base64 Encode/Decode submenu

This submenu include the following actions for encoding or decoding **base 64** 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.

Decode Selection

Replaces a selection of text with the result of decoding that selection.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Base32 Encode/Decode submenu

This submenu include the following actions for encoding or decoding **base32** schemes:

Import File to Encode and Insert

Encodes a file and then inserts the encoded content into the current document at the cursor position.

Decode Selection and Export to File

Decodes a selection of text from the current document and then exports (saves) the result to another file.

Encode Selection

Replaces a selection of text with the result of encoding that selection.

Decode Selection

Replaces a selection of text with the result of decoding that selection.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the

highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Hex Encode/Decode submenu

This submenu include the following actions for encoding or decoding **hex** schemes:

Import File to Encode and Insert

Encodes a file and then inserts the encoded content into the current document at the cursor position.

Decode Selection and Export to File

Decodes a selection of text from the current document and then exports (saves) the result to another file.

Encode Selection

Replaces a selection of text with the result of encoding that selection.

Decode Selection

Replaces a selection of text with the result of decoding that selection.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Join and Normalize Lines (Ctrl + J (Command + J on macOS))

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 + Option + Enter on macOS))

This action has the same result as moving the cursor to the end of the current line and pressing the *ENTER* key.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Go to Definition

Navigates to the definition of the current property.

Flatten Schema

Available only if the JSON document is a JSON schema, it flattens the entire hierarchy of the JSON schema. For more details, see [Flatten JSON Schema \(on page 423\)](#).

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 the 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 \(on page 313\)](#).

Transforming and Querying JSON Documents

Oxygen JSON Editor provides the ability to transform JSON documents to XML or HTML through XSLT or XQuery processing. You also have access to some powerful tools for querying JSON through XPath expressions or XQuery.

Resources

For more information about transforming and querying in JSON, watch our video demonstration:

<https://www.youtube.com/embed/1LHoMhEFagA>

Transforming JSON Documents with XSLT

It is possible to transform JSON documents through XSLT processing. To do so, follow these steps:

1. Create an XSLT 3.0 stylesheet that has the `xsl:initial-template`. You can use one of the following two templates available in the New Document Wizard.
 - **XSLT Stylesheet for JSON** - Processes a JSON document by using a `json-doc()` function and matches the JSON properties from the JSON map.
 - **XSLT Stylesheet for JSON to XML** - Processes a JSON document by using a `json-to-xml()` function and matches the converted XML content.
2. Create a new **XSLT transformation** scenario for your stylesheet.
3. Reference the JSON document that you want to transform using one of these two methods:
 - In the transformation scenario, click the **Parameters** button in the **XSLT** tab and add a parameter that specifies the URL to your JSON document in its value. For example, if you are transforming one of the built-in templates mentioned above, the `input` parameter is added by default and you could specify the URL in its value.
 - Specify the URL to your JSON document in the stylesheet you created. For example, if you use one of the built-in templates mentioned above, you would specify the URL in the value of the `input` parameter (in the `xsl:param` element).
4. Run the transformation.



Tip:

There are some sample files in the `[OXYGEN_INSTALL_DIR]/samples/json/transform` folder that can be used to transform a JSON document to XML or HTML.

Related Information:

[Blog: Transforming JSON](#)

[XSLT Functions on JSON Data](#)

Transforming JSON Documents with XQuery

It is possible to transform JSON documents through XQuery processing. To do so, follow these steps:

1. Create an XQuery file.
2. Create a new **XQuery transformation** scenario for your XQuery file.
3. Reference the JSON document that you want to transform using one of these two methods:
 - In the transformation scenario, click the **Parameters** button in the **XQuery** tab and add a parameter that specifies the URL to your JSON document in its value.
 - Specify the URL to your JSON document in the XQuery file you created.
4. Run the transformation.

**Tip:**

There is a sample XQuery file in the `[OXYGEN_INSTALL_DIR]/samples/json/transform` folder that can be used to transform a JSON document.

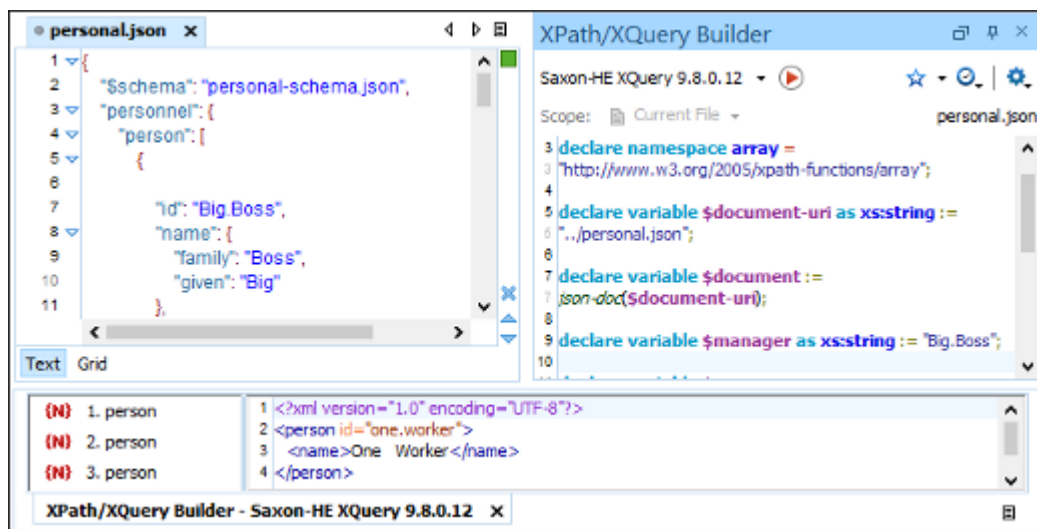
Querying JSON Documents with XPath or XQuery

Oxygen JSON Editor provides an XPath toolbar that makes it easy to quickly query JSON documents using XPath expressions. You can also use the dedicated **XPath/XQuery Builder** view that allows you to compose more complex XPath or XQuery expressions and execute them over JSON documents in **Text** mode.

XPath/XQuery Builder View

You can also use the **XPath/XQuery** view to run XPath and XQuery expressions over a JSON document. For XQuery, you need to reference the JSON document in your XQuery content. For more information about this view, see [XPath Builder View \(on page 474\)](#).

Figure 88. XPath/XQuery Builder View for JSON



Details About Querying JSON Documents Using XPath Expressions

To execute XPath expressions over a JSON document, the document is converted to XML and the XPath is executed over the converted XML document. For this conversion, Oxygen JSON Editor uses the built-in [JSON to XML Converter tool \(on page 381\)](#). The results are mapped back to the original JSON document.

For example, if you have the following JSON document:

```
{
  "personnel": {
    "person": [
      {"name": "Boss"},
      {"name": "Worker"}
    ]
  }
},
```

```
"id": "personnel-id"
}
```

and you want to match the name of the second person, the XPath expression would look like this:

```
/JSON/personnel/person[2]/name
```

The reason why the first element is `JSON` is because if the JSON document contains more than one property on the first level, the converted XML document will have an additional root element called `<JSON>`. For more information, see [JSON to XML Conversion Details \(on page 382\)](#).

The `[2]` in the expression represents the index of the `person` in the array and in this case, it matches the second `person` because the index counting starts with 1.

Editing JSON Schema Documents

Oxygen JSON Editor offers powerful tools that allow you to design, develop, and edit JSON Schemas. These tools include:

- **Text editing mode** ([on page 396](#)) (packed full of editing helpers)
- **Schema Design mode** ([on page 396](#)) (a visual, intuitive schema diagram editor)
- **Grid mode** ([on page 396](#)) (a compact layout of nested tables)
- JSON Schema documentation tool ([on page](#)) for producing high quality output
- **JSON Schema instance generator** ([on page 417](#)) for producing JSON Schema documents from a JSON file
- XSD to JSON Schema converter tool ([on page](#)) for producing a JSON Schema from an XML schema

This section describes the features included in Oxygen JSON Editor for editing JSON Schema documents and how to use them.

Resources

For more information about the JSON support in Oxygen JSON Editor, see the following resources:

- Video: [Introducing JSON Schema Design Mode](#)
- Video: [JSON Editing](#)
- Video: [JSON Tools in Oxygen](#)
- Video: [JSON Schema Search and Refactoring Actions in Oxygen](#)
- Video: [JSON Schema Version 2020-12 Support in Oxygen](#)
- Webinar: [JSON and JSON Schema Support in Oxygen](#)
- Webinar: [Creating and Designing JSON Schemas](#)

JSON Schema Editor

Oxygen JSON Editor includes a specialized JSON Schema editor with various editing features for files that have the `jsonschema` file extension, or for files that have `json` file extension and includes a [meta-schema URL](#)

([on page 422](#)) in the "\$schema" key . The purpose of the JSON schema is to define the legal properties and values of a JSON document to keep it valid and well-formed.

New Document Templates

Oxygen JSON Editor includes a new document template to help you get started creating a JSON Schema document. The template is called **JSON Schema** and it can be found in the **New Document** folder in the **New document wizard** ([on page 225](#)). You can also [customize your own JSON Schema templates](#) ([on page 230](#)) and specify other versions (Draft 04, 06, 07, 2019-09, or 2020-12).



Tip:

You can experiment with a sample of a JSON Schema file available at: [\[OXYGEN-INSTALL-DIR\]/samples/json/personal-schema.json](#).

Text Mode Editor

When editing JSON Schema documents in **Text** editing mode, the usual text editing actions are available, along with various other actions, including:

- [JSON Outline View](#) ([on page 378](#))
- [JSON-specific Syntax Highlighting](#) ([on page 423](#))
- [Search and Find/Replace](#) ([on page 263](#))
- Drag and Drop
- [Validation](#) ([on page 363](#))
- Format and Indent (Pretty Print)

Grid Mode Editor

Oxygen JSON Editor allows you to view and edit the JSON Schema documents in the **Grid** mode. The JSON Schema is represented in **Grid** mode as a compound layout of nested tables and the JSON data and structure can be easily manipulated with table-specific operations or drag and drop operations on the grid components. For more details, see [JSON Grid Mode Editor](#) ([on page 360](#)).

Design Mode Editor

Oxygen JSON Editor provides a powerful, expressive visual schema diagram editor (**Design** mode) for editing JSON Schemas. It is helpful for both content authors who want to visualize or understand a schema and schema designers who develop complex schemas. For all the details, see [JSON Schema Design Mode](#) ([on page 396](#)).

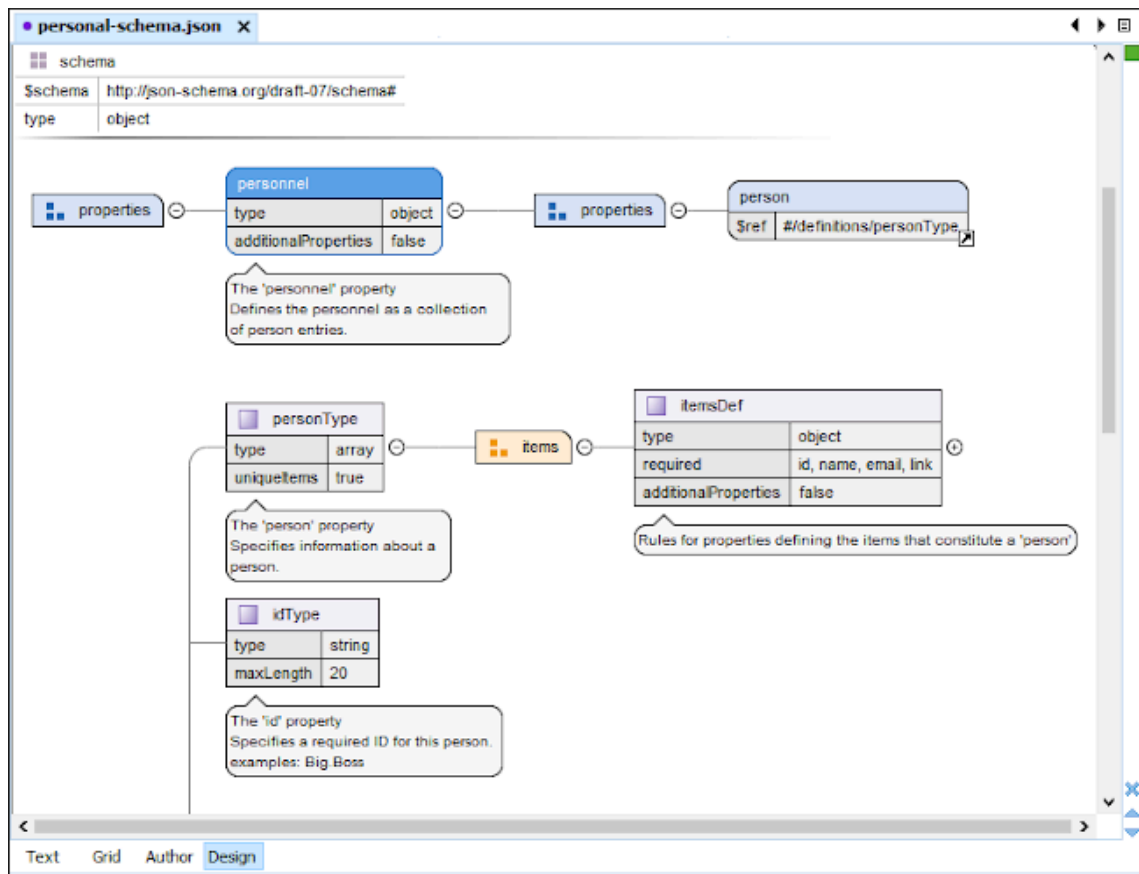
JSON Schema Design Mode (JSON Schema Diagram Editor)

Oxygen JSON Editor provides a powerful, expressive visual schema diagram editor (**Design** mode) for editing JSON Schemas. The structure of the diagram editor is designed to be intuitive and easy to use. The **Design**

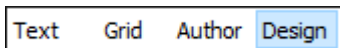
mode was created to help both content authors who want to visualize or understand a schema and schema designers who develop complex schemas.

The JSON Schema **Design** mode includes various [navigation features \(on page 398\)](#), [contextual menu actions \(on page 401\)](#), automatic [validation \(on page 416\)](#), and you can move and edit components directly within the diagram [\(on page 400\)](#).

Figure 89. JSON Schema Design Mode



To switch to the JSON Schema diagram editing mode, select **Design** at the bottom of the editing area.



Resources

For more information about the JSON Schema Design mode, see the following resources:

- Video: [Introducing JSON Schema Design Mode](#)
- Video: [JSON Schema Search and Refactoring Actions in Oxygen](#)
- Video: [JSON Schema Version 2020-12 Support in Oxygen](#)
- Webinar: [The New JSON Schema Diagram Editor](#)
- Webinar: [Create JSON Schema in Design Mode](#)
- Webinar: [Creating and Designing JSON Schemas](#)

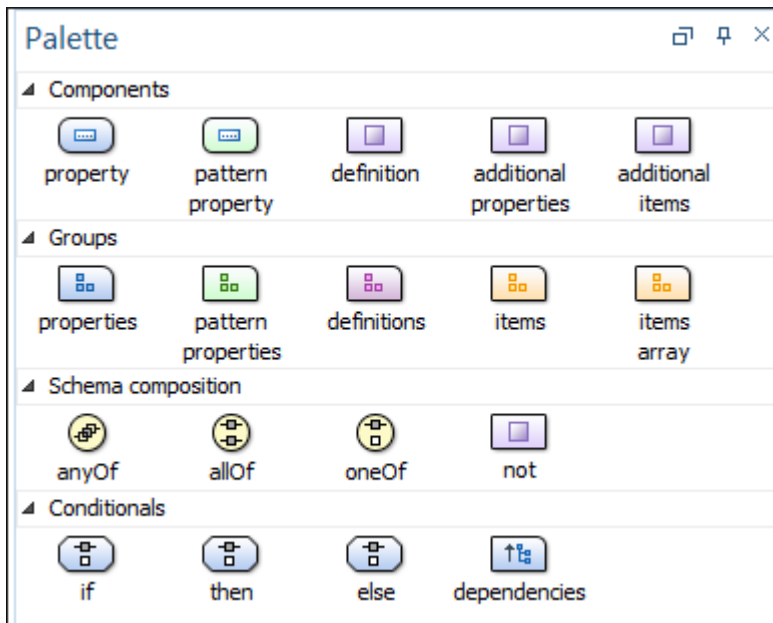
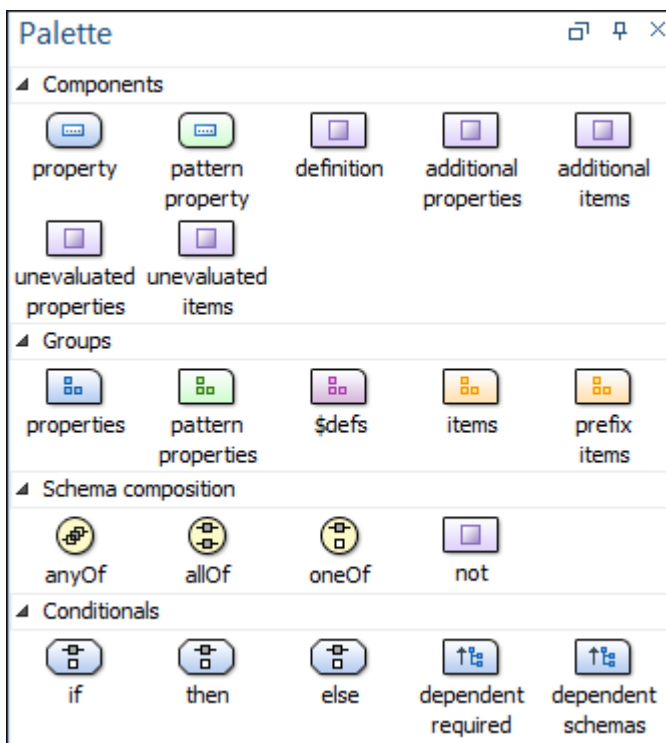
Navigation in the JSON Schema Design Mode

The following editing and navigation features work for all types of schema components in the JSON Schema **Design** mode:

- 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 macOS)** key. To deselect one of the components, use **Ctrl + Single-Click (Command + Single-Click on macOS)**.
- Use the arrow keys to navigate the diagram vertically and horizontally.
- To expand a component, click the  widget to the right of the component.
- Use *Home/End* keys to jump to the first/last component from the same level. Use **Ctrl + Home (Command + Home on macOS)** key combination to go to the diagram root and **Ctrl + End (Command + End on macOS)** 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 property from a properties parent and you press the left arrow key to jump to the properties parent, when you press the right arrow key, then the selection will be moved to the second property.
- Go back and forward between components viewed or edited in the diagram by using the following toolbar actions:
 -  **Back** (go to previous schema component).
 -  **Forward** (go to next schema component).
 -  **Go to Last Modification** (go to last modified schema component).
- Go to the definition of a property by clicking on the **Go to Definition** widget (). You can then use the  **Back** or  **Forward** toolbar buttons to go back and forth between the properties.
- The dedicated **Outline** view displays all of the properties, pattern properties, and definitions found in the document and you can click on any property/definition to navigate to it within the document.
- Search in the diagram using the [Find/Replace dialog box \(on page 273\)](#) or the [Quick find toolbar \(on page 286\)](#). You can find components only in the current file scope.

JSON Schema Palette View (Available in Design Mode)

The **Palette** view is designed to offer quick access to JSON schema components and to improve the usability of the JSON schema diagram builder. You can use the **Palette** to drag and drop components into the **Design** mode. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

Figure 90. JSON Palette View (for schema versions draft-04, draft-06, draft-07)**Figure 91. JSON Palette View (for schema versions draft 2019-09, draft 2020-12)**

Components are organized functionally into 4 collapsible categories:

- **Components:** *property*, *patternProperty*, *definition*, *additionalProperties*, *additionalItems*, *unevaluatedProperties* (for draft 2019-09 or draft 2020-12 schemas), *unevaluatedItems* (for draft 2019-09 or draft 2020-12 schemas).
- **Groups:** *properties*, *patternProperties*, *definitions/\$defs* (for draft 2019-09 or draft 2020-12 schemas), *items*, *itemsArray*, *prefixItems* (for draft 2019-09 or draft 2020-12 schemas).

**Note:**

Two of the group palette components presented above (*itemsArray* and *patternProperty*), have no match in JSON schema keywords. They were introduced only as a design convenience and their use does not generate code that would affect the validity of the edited schema.

- **Schema composition:** *anyOf*, *allOf*, *oneOf*, *not*.
- **Conditionals:** *if*, *then*, *else*, *dependencies* (for draft-04, draft-06, or draft-07 schemas), *dependentRequired* (for draft 2019-09 or draft 2020-12 schemas), *dependentSchemas* (for draft 2019-09 or draft 2020-12 schemas).

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 (*on page 153*) (default color is red).

**Tip:**

When dragging and dropping a property/definition or pattern property on a component, if it does not have the corresponding group component (*properties*, *definitions*, *patternProperties*), it will automatically be created.

Resources

For more information about the Schema palette, watch our video demonstration:

<https://www.youtube.com/embed/r5SLk3XL0Us>

Editing Actions in JSON Schema Design Mode

The JSON Schema **Design** mode includes various editing features, including:

- You can edit a JSON Schema using various [contextual menu actions \(on page 401\)](#).
- You can copy/paste or drag/drop to move existing components to other locations in an JSON Schema. You can also use the Move Up/Down actions to change the order of the components in a parent.
- You can edit 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 JSON Schema Properties preferences page \(on page 132\)](#).

- When switching between editing modes, the cursor position is synchronized. For example, if you switch from **Design** to **Text** mode, the cursor will be in the same position within the document in **Text** mode as it was in **Design** mode, and vice versa.
- The content completion assistant can be used for in-place editing within the JSON schema **Design** mode. For example, when editing properties, you can use **Ctrl+Space** to invoke the content completion window and the proposals are offered based on the defined JSON schema, according to the version used.
- You can drag properties/definitions from the **Outline** view and drop them into appropriate location in the diagram editor.

Contextual Menu Actions in the JSON Schema Design Mode

The contextual menu of the **Design** mode includes the following actions:



Go to Definition **[Ctrl + Shift + Enter]**

Navigates to the referenced schema component. This action is also available by clicking the arrow displayed in its bottom right corner.

Edit Properties

Allows you to edit the properties of the selected component in a in-place editor.

Properties that have set values are rendered in bold while unset properties are rendered with a gray foreground. You can edit any property (set or unset) by double-clicking or by pressing **Ctrl + Enter** (**Command + Enter** on macOS).

You can delete any property already set by pressing **Delete**. This operation does not mean the selected property is deleted from the table. It means the property is *unset* (rendered with gray foreground). The **Edit** and **Remove** actions are also available on the contextual menu in the table.

If the `type` property changes as a result of an editing/removal action, then the list of properties presented in the table is updated according to the new schema type.



Note:

When filling in string values they should not be enclosed in quotation marks, these are added automatically.



Note:

For array values simply fill in the items that will constitute the array, separated by commas.

Edit Annotations

Allows you to edit the annotations for the selected schema component in the **Edit Annotations** dialog box. Annotations are not required, but they are encouraged as a good practice and can make the schema “self-documenting”.

The **title** and **description** must be strings. A **title** is preferably short, whereas a **description** provides a more lengthy explanation about the data described by the schema. The **default** keyword specifies a default schema value that can be anything. The **examples** keyword is meant to provide an array of examples that validate against the schema. Its items must be separated by a comma.

Annotations that have set values are rendered in bold while unset annotations are rendered with a gray foreground. You can unset an annotation by using the **Delete** key or the **Remove** action that is available in the contextual menu of the table.

By default, annotations are rendered under the graphical representation of the component. To edit the annotations, use the **Edit Annotations** action from the contextual menu or simply double-click the annotations area (if any).

Edit Dependencies

Available for `dependencies` and `dependentRequired` components, this action allows you to add, rename, delete, and edit the values for dependencies.

Make Required

Marks the selected property as being required in the parent object. By default, the defined properties are not required in the JSON schema. You can set a list of required properties in the `required` keyword. By invoking the action, the name of the property is added in the parent object's `required` keyword.

Make Optional

Marks the selected property as being optional in the parent object. By default, the defined properties are optional in the JSON schema. You can set a list of required properties in the `required` keyword. By invoking the action, the name of the property is deleted from the parent object's `required` keyword.

Refactoring > Extract definition in another file

Extracts a definition to a new file. If the file does not already exist, the action will create a new file and a document preset will be used to match the current schema specification. If the file does exist, the action will find a corresponding group (or create one) to append the extracted definition. This action can also be used on a selection of multiple definitions.

Refactoring > Extract definition in current file

Extracts a definition as a global definition and references it. It can be used on a property to extract its definition (in case you want to reuse it) or on a local definition to extract it as global one. This action can also be used on a selection of multiple definitions. Note that this action is not available for global definitions.

Refactoring > Convert type to 'any' type

Converts the `type` for the selected property, definition, or conditional into an `any` `type` with the value `true` or `false`. You can set `true` value to represent a schema that matches anything, or `false` for a schema that matches nothing.

Refactoring > Convert 'any' type to standard type

Converts the `any` `type` for the selected property, definition, or conditional into a standard `type`.

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.

 Undo [Ctrl + Z (Command + Z on macOS)]

Reverses the last editing action.

 Redo [Ctrl + Y (Command + Shift + Z on macOS, Ctrl + Shift + Z on Linux/Unix)]

Recreates the last editing action that was reversed by the **Undo** function.

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. These files are identified by searching the references of the selected component in the scope provided.

 Cut [Ctrl + X (Command + X on macOS)]

Cuts the selected component(s).

 Copy [Ctrl + C (Command + C on macOS)]

Copies the selected component(s) to the clipboard.

 Paste [Ctrl + V (Command + V on macOS)]

Pastes the component(s) from the clipboard as children of the selected component.

Remove [Delete key]

Removes the selected component(s).

Move Up [Alt + UpArrow (Option + UpArrow on macOS)]

Moves a component up in its parent.

Move Down [Alt + DownArrow (Option + DownArrow on macOS)]

Moves a component down in its parent.

Search > Search References

Available on components that have a *Definition* type in the diagram, it searches all references of the selected definition in a scope determined by the schemas referenced in the file and the schemas declared in the validation scenarios associated with them. You can also use it on the *Schema* type component (the root of the schema diagram) to search for all references to the schema name.

Search > Search References in

Available on components that have a *Definition* type in the diagram, it is an extension of the **Search References** action, where the search for references is additionally done in the file(s) specified when defining a scope in the resulting dialog box. You can also use it on the *Schema* type component (the root of the schema diagram) to search for references to the schema name.

Search > Search Occurrences in File [Ctrl + Shift + U (Command + Shift + U on macOS)]

Available on all components that have a *Definition* type in the diagram, it searches all occurrences of the currently selected definition in the current file. You can also use it on the *Schema* type component (the root of the schema diagram) to search for all occurrences of the schema name in the current file.

Flatten Schema

Flattens the entire hierarchy of JSON schemas. For more details, see [Flatten JSON Schema \(on page 423\)](#).

**Expand All**

Recursively expands all sub-components of the selected component.

**Collapse All**

Recursively collapses all sub-components of the selected component.

Print Selection

Prints the selected component diagram.

Save as Image

Saves the selected component diagram as image, in JPEG, BMP, SVG or PNG format.

**Options**

Opens the [JSON Schema preferences page \(on page 132\)](#) where you can control which properties to display for JSON Schema components in the JSON Schema **Design** mode.

JSON Schema Design Mode Components and Properties

A schema diagram contains a series of interconnected components. In the JSON Schema **Design** mode, each component is displayed with distinguishable graphics and the thickness of the lines that connect the components identify whether the connected component is required or optional (a thick line denotes a required component while a thin line denotes an optional component).

Figure 92. Example: Required Component

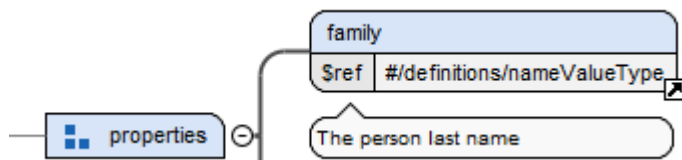
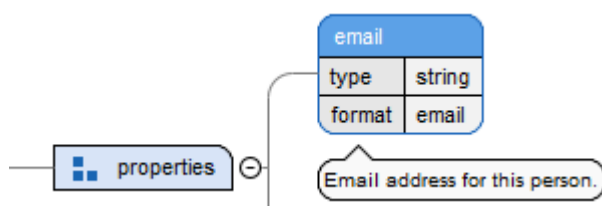


Figure 93. Example: Optional Component

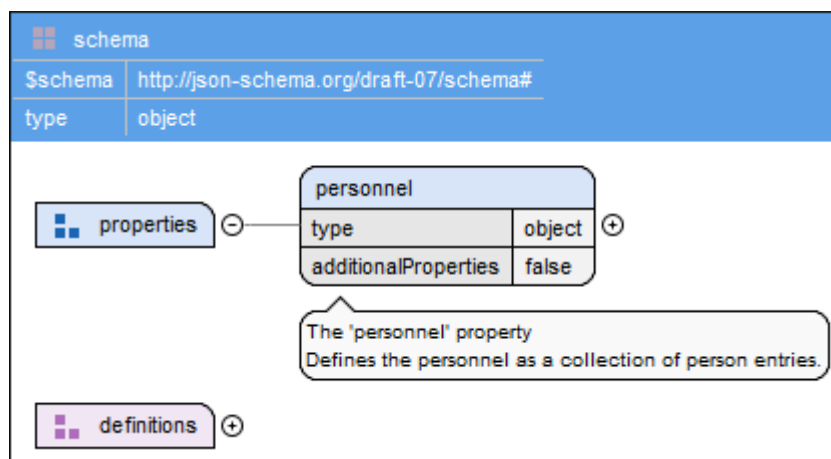


This section provide details about the available components and their corresponding graphics, along with details about component properties.

JSON Schema Components

Schema

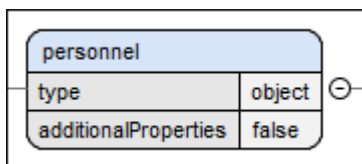
Figure 94. The schema Component



Description: Defines the root element of a JSON schema. A JSON schema document contains all the steps that are necessary to be performed to validate JSON documents and it contains a collection of JSON schema components.

Property

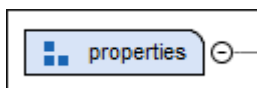
Figure 95. Example of a property Component



Description: Each `key:value` pair is known as a **property** of the object. The value can be any JSON data type.

Properties

Figure 96. The properties Component



Description: An object is valid against the **properties** keyword if every property that is present in both the object and the value of this keyword validates against the corresponding schema. The value must be an object, where properties must contain valid JSON schemas (object or boolean). Only the property names that are present in both the object and the keyword value are checked.

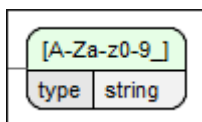


Note:

Properties with a boolean value are presented in diagram as components. You can change the boolean value by modifying the `any type` property value. You can also convert a boolean `any type` property into a standard type property using the **Convert property to standard type** contextual menu action.

Pattern Property

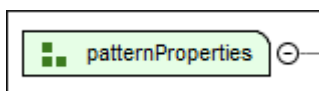
Figure 97. Example of a patternProperty Component



Description: Maps regular expressions to schemas. If a property name matches the given regular expression, the property value must validate against the corresponding schema.

Pattern Properties

Figure 98. The patternProperties Component

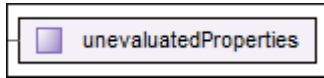


Description: An object is valid against the **patternProperties** keyword if every property where a property name (key) matches a regular expression from the value of this keyword is also valid against the corresponding

schema. The value must be an object where the keys must be valid regular expressions and the corresponding values must be valid JSON schemas (object or boolean).

Unevaluated Properties (for draft 2019-09 or draft 2020-12 schemas)

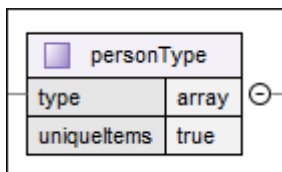
Figure 99. The `unevaluatedProperties` Component



Description: An object is valid against the **unevaluatedProperties** keyword if every unevaluated property is valid against the schema defined by the value of this keyword. Unevaluated properties are the properties that were not evaluated anywhere in the current schema. This keyword can see through adjacent keywords, such as **allOf**.

Definition

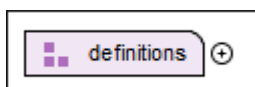
Figure 100. Example of a definition Component



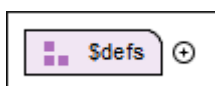
Description: The definition of a component of a schema. It can be referenced from a property to define its specification.

Definitions

Figure 101. The definitions Component



or



Description: The optional **definitions** keyword (or **\$defs** for *draft 2019-09* or *draft 2020-12* schemas) does not directly validate data, but it contains a map of validation schemas. The value can be anything.

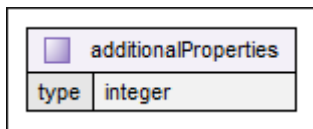


Note:

Definitions with a boolean value are presented in diagram as components. You can change the boolean value by modifying the definition's `any type` property value. You can also convert a definition's boolean `any type` property into a standard `type` property using the **Convert definition to standard type** contextual menu action.

Additional Properties

Figure 102. The `additionalProperties` Component



Description:

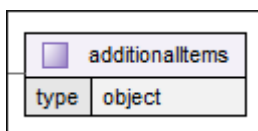
An object is valid against the **additionalProperties** keyword if all *unchecked* properties are valid against the schema defined by the value of this keyword. *Unchecked* properties are the properties not checked by the **properties** and **patternProperties** keywords (if a property name is not present in the **properties** keyword and does not match any regular expression defined by the **patternProperties** keyword, then it is considered *unchecked*). The value must be a valid JSON schema (object or boolean).

To be more concise, if there are *unchecked* properties:

- If the value of the **additionalProperties** keyword is *true*, it is always valid.
- If the value is *false*, it is never valid.
- If the value contains an object (schema), every property must be valid against that schema.

Additional Items

Figure 103. The `additionalItems` Component



Description: An array is valid against the **additionalItems** keyword if all *unchecked* items are valid against the schema defined by the keyword value. An item is considered *unchecked* if the **items** keyword or **prefixItems** keyword (starting with draft 2020-12) contains an array of schemas and does not have a corresponding position (index). The value must be a valid JSON schema (object or boolean).

Items

Figure 104. The `items` Component

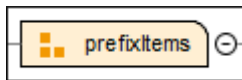


Description: An array is valid against the **items** keyword if the items are valid against the corresponding schemas provided by the keyword value. The value can be:

- A valid JSON schema (object or boolean). Every item must be valid against this schema.
- An array of valid JSON schemas. Each item must be valid against the schema defined at the same position (index). Items that do not have a corresponding position (e.g. an array contains 5 items but this keyword only has 3) are considered valid unless the **additionalItems** keyword is present, which will decide the validity.

Prefix Items (for draft 2020-12 schemas)

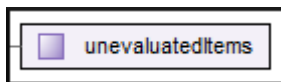
Figure 105. The prefixItems Component



Description: An array is valid against the **prefixItems** keyword if items are valid against the corresponding schemas provided by the keyword value. The value of this keyword must be an array of valid JSON schemas and each item must be valid against the schema defined at the same position (index). Items that do not have a corresponding position (an array contains 5 items and this keyword only has 3) will be considered valid, unless the **items** keyword is present (which will decide the validity).

Unevaluated Items (for draft 2019-09 or draft 2020-12 schemas)

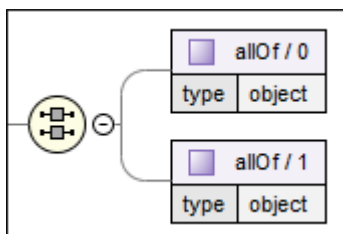
Figure 106. The unevaluatedItems Component



Description: An object is valid against the **unevaluatedItems** keyword if every unevaluated item is valid against the schema defined by the value of this keyword. Unevaluated items are the items that were not evaluated anywhere in the current schema. This keyword can see through adjacent keywords, such as **allOf**.

Allof

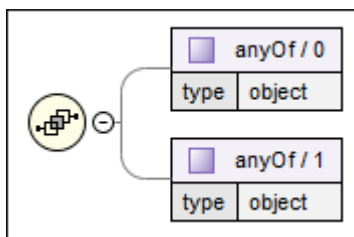
Figure 107. Example of an allOf Component



Description: An instance is valid against the **allOf** keyword if it is valid against all schemas defined by the value of this keyword. The value of this keyword must be an array of valid JSON schemas (objects or boolean).

AnyOf

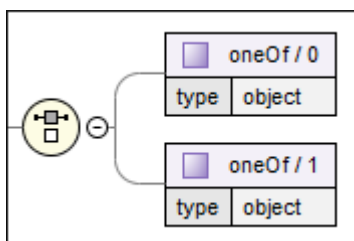
Figure 108. Example of an anyOf Component



Description: An instance is valid against the **anyOf** keyword if it is valid against at least one schema defined by the value of this keyword. The value must be an array of valid JSON schemas (objects or boolean).

OneOf

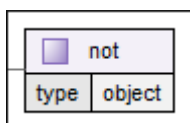
Figure 109. Example of an oneOf Component



Description: An instance is valid against the **oneOf** keyword if it is valid against exactly one schema defined by the value of this keyword. The value must be an array of valid JSON schemas (object or boolean).

Not

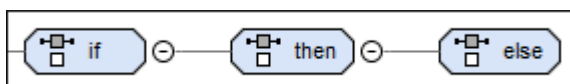
Figure 110. Example of a not Component



Description: An instance is valid against the **not** keyword if it is not valid against the schema defined by the value of this keyword. The value must be a valid JSON schema (object or boolean).

If/Then/Else

Figure 111. Examples of an if, then, and else Components

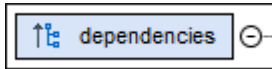


Description: A conditional structure that contains three keywords: **if**, **then**, and **else**. Every keyword value must be a valid JSON schema (object or boolean). When the **if** keyword is not present, the **then** and **else** keywords are ignored. When the **if** keyword is present, at least one of the **then** or **else** keywords should also be present (or both). The instance is valid against this keyword in one of the following cases:

- The **if** keyword validates the instance and the **then** keyword also validates it.
- The **if** keyword does not validate the instance but the **else** keyword validates it.

Dependencies

Figure 112. The dependencies Component



Description: An object is valid against the **dependencies** keyword if it meets all dependencies specified by this keyword value. Only property names (from this keyword value) that are also present in the object are checked. The value of this keyword must be an object, where property values can be:

- Objects representing valid JSON schemas and the whole object must match the entire schema.



Important:

For *draft 2019-09* and *draft 2020-12* schemas, you should use the **dependentSchemas** keyword ([on page 411](#)) instead.

- Arrays of strings representing property names, then the object must contain all property names.

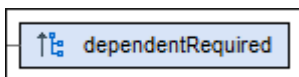


Important:

For *draft 2019-09* and *draft 2020-12* schemas, you should use the **dependentRequired** keyword ([on page 411](#)) instead.

Dependent Required (for draft 2019-09 or draft 2020-12 schemas)

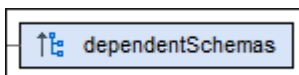
Figure 113. The dependentRequired Component



Description: An object is valid against the **dependentRequired** keyword if it meets all dependencies specified by this keyword value. The value of this keyword must be an object where property values must be arrays of strings representing property names, and the object must contain all property names. Only property names (from this keyword value) that are also present in the object are checked.

Dependent Schemas (for draft 2019-09 or draft 2020-12 schemas)

Figure 114. The dependentSchemas Component



Description: An object is valid against the **dependentSchemas** keyword if it meets all dependencies specified by this keyword value. The value of this keyword must be an object where property values must be objects

representing valid JSON schemas, and the whole object must match the entire schema. Only property names (from this keyword value) that are also present in the object are checked.

Related information

[JSON Schema Component Properties \(on page 412\)](#)

JSON Schema Component Properties

Table 2. JSON Schema Diagram Component Properties

Prop- er- ties Group	Prop- erty Name	Description
Common Prop- er- ties	type	Specifies the type of data that the schema is expecting to validate. This keyword is not mandatory and the value must be a string representing a valid data type, or an array of strings representing a valid list of data types. When specifying multiple types, their order is irrelevant to the validation process, but make sure that a data type is specified only once.
	\$ref	An instance is valid against this keyword if it is valid against the schema that points to the location indicated in its value. The value must be a string representing a URI, URI reference, URI template, or JSON pointer.  Note: A <i>draft 2019-09</i> or <i>draft 2020-12</i> schema that contains a <code>\$ref</code> in conjunction with other type-specific keywords (such as <i>properties</i> or <i>items</i>) is processed as a combined schema, under the <i>allOf</i> criterion. This means that for an instance to be valid, it has to validate against both the referenced schema and the schema defined in-place by those keywords.
	\$id	Specifies a unique ID for a document or a document sub-schema. The value must be a string representing a URI. All sub-schema IDs are resolved relative to the document ID. It is not a required keyword, but it is considered best practice to use it.
	enum	An instance validates against this keyword if its value can be found in the items defined by its value. The value must be an array that contains anything (an empty array is not allowed).
	const	An instance validates against this keyword if its value equals the value of this keyword. The value can be anything.
	\$comment	Contains an observation about the schema. The value must be a string.

Table 2. JSON Schema Diagram Component Properties (continued)

Prop- er- ties Group	Property Name	Description
	readOnly	Used to mark specific properties as <i>read-only</i> .
	writeOnly	Used to mark specific properties as <i>write-only</i> .
	deprecated	Used to indicate that the instance value is deprecated and should not be used since it might be removed in the future.
Ob- ject Prop- er- ties	additional- Properties	An object is valid against this keyword if all <i>unchecked</i> properties are valid against the schema defined by its value. <i>Unchecked</i> properties are the properties not checked by the properties and patternProperties keywords (if a property name is not present in the properties keyword and does not match any regular expression defined by the pattern-Properties keyword, then it is considered <i>unchecked</i>). The value must be a valid JSON schema (object or boolean). To be more concise, if we have <i>unchecked</i> properties: • If the value of this keyword is <i>true</i>, it is always valid. • If the value is <i>false</i>, it is never valid. • If the value contains an object (schema), every property must be valid against that schema.
	unevaluat- edProp- erties	Similar to the additionalProperties keyword except that this one can recognize properties that declared in subschemas.
	maxProp- erties	An object is valid against this keyword if the number of properties it contains is lower than or equal to its value. The value must be a non-negative integer. Using 0 as a value means that the object must be empty (no properties).
	minProp- erties	An object is valid against this keyword if the number of properties it contains is greater than or equal to its value. The value of this keyword must be a non-negative integer. Using 0 as a value has no effect.
	pattern- Properties	An object is valid against this keyword if every property where a property name (key) matches a regular expression from its value and is also valid against the corresponding schema. The value must be an object where the keys must be valid regular expressions and the corresponding values must be valid JSON schemas (object or boolean).

Table 2. JSON Schema Diagram Component Properties (continued)

Prop- er- ties Group	Prop- erty Name	Description
	property-Names	An object is valid against this keyword if every property name (key) is valid against its value. The value must be a valid JSON schema (an object or a boolean).
	required	An object is valid against this keyword if it contains all property names (keys) specified by its value. The value must be a non-empty array of strings that represent property names.
	dependencies	An object is valid against this keyword if it meets all dependencies specified by its value.  Note: For <i>draft 2019-09</i> and <i>draft 2020-12</i> schemas, you should use the dependent-Required and dependentSchemas keywords instead.
	dependent-Required	An object is valid against this keyword if it meets all dependencies specified by its value. The value must be an object where property values must be arrays of strings representing property names, and the object must contain all property names.
	dependent-Schemas	An object is valid against this keyword if it meets all dependencies specified by its value. The value must be an object where property values must be objects representing valid JSON schemas, and the whole object must match the entire schema.
Ar- ray Prop- er- ties	additional-Items	An array is valid against this keyword if all <i>unchecked</i> items are valid against the schema defined by its value.
	unevaluatedItems	An array is valid against this keyword if the value is a valid JSON schema that will be applied to all array items that were not evaluated by other keywords for items (<code>prefix-Items</code> , <code>items</code> , <code>contains</code>).
	contains	An array is valid against this keyword if at least one item is valid against the schema defined by its value. The value must be a valid JSON schema (object or boolean).
	maxContains	An array is valid against this keyword if the number of <i>contains</i> is lower than or equal to its value. The value must be a non-negative integer.
	minContains	An array is valid against this keyword if the number of <i>contains</i> is greater than or equal to its value. The value must be a non-negative integer.

Table 2. JSON Schema Diagram Component Properties (continued)

Prop- er- ties Group	Prop- erty Name	Description
	items	An array is valid against this keyword if the items are valid against the corresponding schemas provided by its value. The value of this keyword can be: • A valid JSON schema (object or boolean). Every item must be valid against this schema. • An array of valid JSON schemas. Each item must be valid against the schema defined at the same position (index). Items that do not have a corresponding position (e.g. an array contains 5 items but this keyword only has 3) will be considered valid unless the additionalItems keyword is present, which will decide the validity.
	maxItems	An array is valid against this keyword if the number of items it contains is lower than or equal to its value. The value must be a non-negative integer.
	minItems	An array is valid against this keyword if the number of items it contains is greater than or equal to its value. The value must be a non-negative integer.
	prefixItems	An array is valid against this keyword if each item is a schema that corresponds to each index of the document's array (where the first element validates the first element of the input array, the second element validates the second element of the input array, and so on).
	uniqueItems	An array is valid against this keyword if an item cannot be found more than once in the array. The value must be boolean. If set to <i>false</i> , the keyword validation will be ignored.
Num- ber/In- te- ger Prop- er- ties	exclusive- Maximum	A number is valid against this keyword if it is strictly lower than its value. The value must be a number (integer or float) or boolean.
	exclusiveM- inimum	A number is valid against this keyword if it is strictly greater than its value. The value must be a number (integer or float) or boolean.
	maximum	A number is valid against this keyword if it is lower than or equal to its value. The value must be a number (integer or float).

Table 2. JSON Schema Diagram Component Properties (continued)

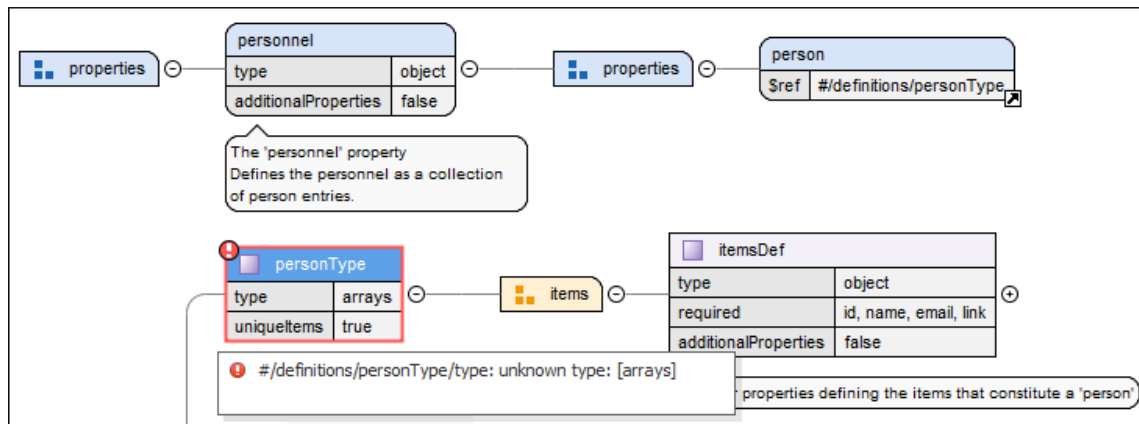
Prop- er- ties Group	Property Name	Description
	minimum	A number is valid against this keyword if it is greater than or equal to its value. The value must be a number (integer or float).
	multipleOf	A number is valid against this keyword if the division between the number and its value results in an integer. The value must be a strictly positive number (zero is not allowed).
String Prop- er- ties	contentEn- coding	A string is valid against this keyword if it is encoded using the method indicated by its value. The value must be a string.
	content- MediaType	A string is valid against this keyword if its content has the media type (MIME type) indicated by its value. If the contentEncoding keyword is also specified, the decoded content must have the indicated media type. The value must be a string.
	format	Performs a semantic validation on data. The value must be a string that represents a format. The keyword behavior depends on the data type, meaning that the same format name for a string behaves differently on a number, or is missing, because not all data types must implement a format and usually differing data types have different formats.
	maxLength	A string is valid against this keyword if its length is lower than or equal to its value. The value must be a non-negative integer.
	minLength	A string is valid against this keyword if its length is greater than or equal to its value. The value must be a non-negative integer.
	pattern	A string is valid against this keyword if it matches the regular expression specified by its value. The value must be a string that represents a valid regular expression.

Related information

[JSON Schema Components \(on page 405\)](#)

JSON Schema Design Mode Validation

Validation for the **Design** mode is seamlessly integrated in the Oxygen JSON Editor [JSON Schema validation support \(on page 421\)](#). You can ensure that the JSON Schemas you develop comply with JSON standards by using the built-in validation engine. You can also configure a validation scenario to use an external JSON Schema validation engine. A validation scenario can also be configured to define a main module of a complex JSON Schema to validate modules in the context of the larger schema structure.

Figure 115. JSON Schema Design Mode Validation

Visual Error Markers

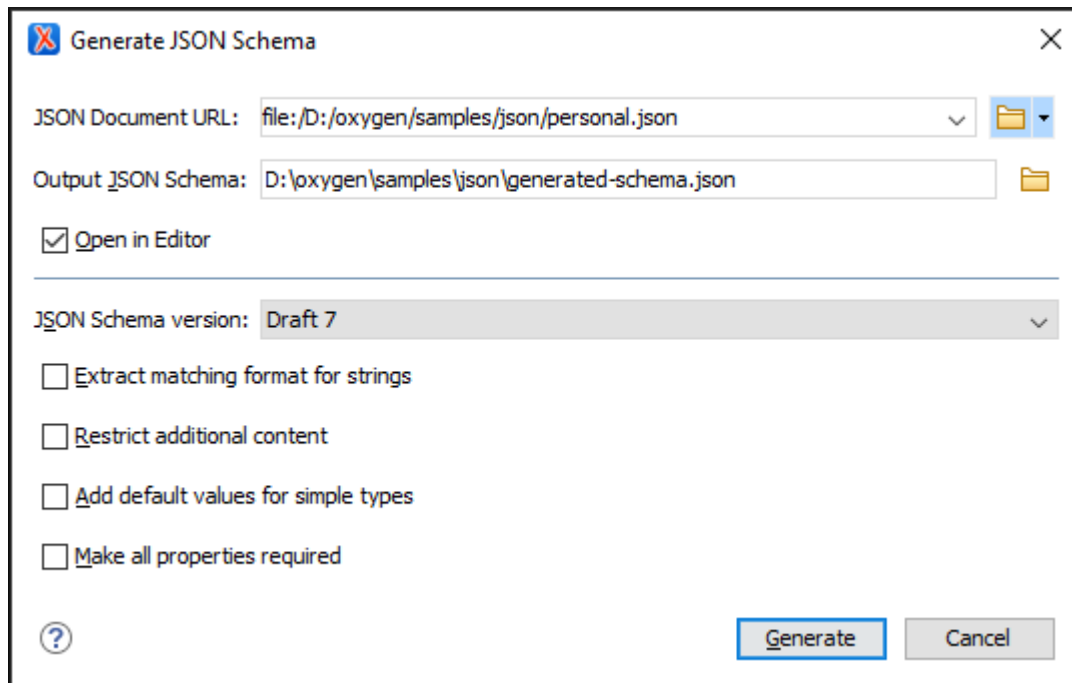
A schema validation error is presented by highlighting the invalid component in the following locations:

- The component is surrounded with a red or yellow border within the diagram (you can customize these colors in the [Document Validation preferences page \(on page 153\)](#)).
- A tooltip contains information about the error when hovering over the error within the diagram.
- The vertical stripe at the right side of the editor displays error markers.
- In the message area at the bottom of the editor (denoted with a red icon .

If you validate the entire schema using the  **Validate** action from the  **Validation** toolbar drop-down menu, all validation errors will be presented in the **Results** pane at the bottom of the application. To resolve an error, just click it and the corresponding schema component will be displayed as the diagram root so that you can easily correct the error.

Generating JSON Schema from a JSON File

Oxygen JSON Editor includes a tool for generating a sample JSON Schema from a JSON file. To generate a sample JSON Schema, select **Generate JSON Schema** from the **Tools > JSON Tools** menu. The action opens a dialog box where you can configure some options for generating the JSON Schema.

Figure 116. Generate JSON Schema Dialog Box

The **Generate JSON Schema** dialog box includes the following fields and options:

JSON Document URL

The URL of the JSON file. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the ▾ **Browse** drop-down list.

Output JSON Schema

The path to the folder where the generated JSON Schema will be saved.

Open in Editor

If selected, the generated JSON Schema is opened in the editor.

JSON Schema version

The version of the resulting JSON schema. The possible choices are: **Draft 4**, **Draft 6**, **Draft 7**, **Draft 2019-09**, and **Draft 2020-12**.

Extract matching format for strings

If selected, the generator will attempt to find a format that matches the string values from the JSON Document.

Restrict additional content

If selected, *additionalProperties* (for objects) and *additionalItems* (for arrays) will be set to *false* in the resulting schema. By default, these keys are not in the schema, meaning that providing additional content (according to the schema) is allowed.

Add default values for simple types

If selected, the *default* values (*0* for number, *""* for string, *false* for boolean) and *examples* for strings will be added.

Make all properties required

If selected, the generator will mark all the properties as required in the resulting schema.

You can click **Generate** at any point to generate the JSON Schema.

Generating Sample JSON Files from a JSON Schema

Oxygen JSON Editor includes a tool for generating sample JSON files. To generate sample JSON files from a JSON Schema, select **Generate Sample JSON Files** from the **Tools > JSON Tools** menu. The action opens a dialog box where you can configure a variety of options for generating the files.

Figure 117. Generate Sample JSON Files Dialog Box

The **Generate Sample JSON Files** dialog box includes the following fields and options:

Schema URL

The URL of the Schema location. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the **Browse** drop-down list. The tool supports schemas with versions *Draft 04*, *06*, *07*, *2019-09*, and *2020-12*.

Associate schema in the document

If enabled, the specified schema will be associated with the generated files.

Output folder

Path to the folder where the generated JSON instances will be saved.

File name

The name of the instance(s) that will be generated. By default, `instance.json` is used.

Number of instances

The desired number of JSON instances to be generated. When more than one instance is generated, the index of the instance will be added to its file name.

Property value

You can specify the way the values of the properties are generated. The following options are available:

- *None* - Assigns empty values for properties (a template file will be generated). This is the default value.
- *Default* - Assigns the name of the property as the value (for strings) or assigns the specified minimum value (for numbers).
- *Random* - Assigns random values according to schema restrictions.

Generate optional properties

If selected, the JSON instance will be generated with optional properties that are defined in the JSON schema. Otherwise, only the required properties will be generated.

Generate additional content

If selected, the JSON instance will be generated with additional properties that are defined in the JSON schema as `additionalProperties` and additional items that are defined as `additionalItems` (in the case of an Array).

Choice strategy

You can specify the way an instance will be generated from a schema that contains a `CombinedSchema` (with either *oneOf* or *anyOf*). The following options are available:

- *First* - The first defined schema in *oneOf* or *anyOf* will be used.
- *Random* - A random schema defined in *oneOf* or *anyOf* will be used.

Recursion level

This option controls the maximum allowed depth (must be a number), in case the selected schema contains recursive calls of `$ref` schemas referencing one another. By default, it is set to 1, meaning that the generation for the recursive calls will stop after the first iteration.

Open first instance in editor

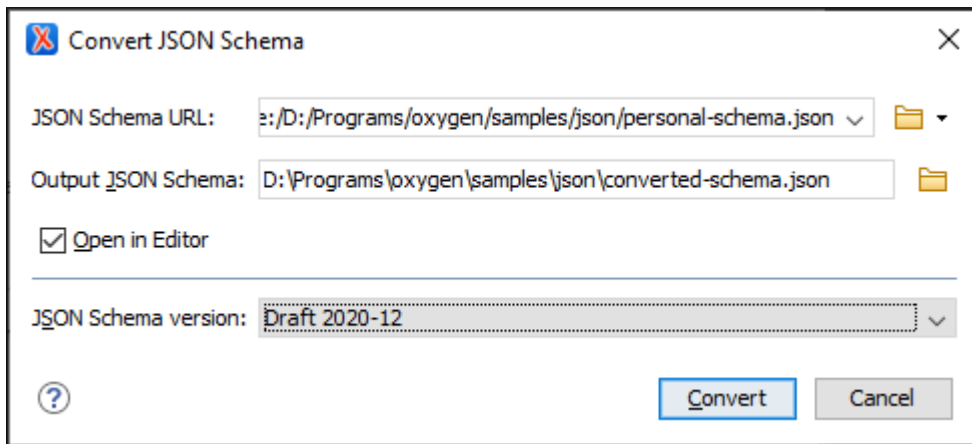
If selected, the first generated instance is opened in the editor.

You can click **OK** at any point to generate the sample JSON files.

JSON Schema Converter

Oxygen JSON Editor includes a tool for converting an older version of a JSON schema (Draft 4, 6, or 7) to the latest versions (Draft 2019-09 or Draft 2020-12).

To convert a JSON schema, select **Convert JSON Schema** from the **Tools > JSON Tools** menu. The action opens a dialog box where you can configure some options for converting the JSON Schema.

Figure 118. Convert JSON Schema Dialog Box

The **Convert JSON Schema** dialog box includes the following fields and options:

JSON Schema URL

The URL of the JSON schema file. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the **Browse** drop-down list.

Output JSON Schema

The path to the folder where the converted JSON schema will be saved.

Open in Editor

If selected, the converted JSON schema is opened in the editor.

JSON Schema version

The version of the resulting JSON schema. The possible choices are: **Draft 2019-09** or **Draft 2020-12**.

You can click **Convert** at any point to generate the JSON Schema.

Conversion Notes

- The `$schema` declaration is changed according to the selected JSON schema version.
- The `definitions` keyword is converted to `$defs` and all the references are updated.
- The `dependencies` keyword is split into `dependentRequired` and `dependentSchemas`.
- The `items` keyword (tuple array) is converted to `prefixItems` (Draft 2020-12).
- The `additionalItems` keyword is converted to `items` (Draft 2020-12, only if `prefixItems` is present).
- The `exclusiveMinimum` and `exclusiveMaximum` keywords with boolean values (Draft 4) are removed.
- The `id` keyword (Draft 4) is converted to `$id`.
- The `$ref` keyword wrapped into 1-item `allOf` is unwrapped because the latest versions allow processing `$ref` along with other keywords.

Validating JSON Schema Documents

A *valid* JSON Schema document is a *well-formed* document that also conforms to the JSON meta-schema rules that defines the legal syntax of a JSON Schema document.

If a JSON document includes a meta-schema URL in the document root with the **"\$schema"** key, the file will be validated as a JSON Schema against the specified meta-schema.

Quick Reference

- If there is a **"\$schema"**: **"http://json-schema.org/draft-04/schema"** property in the schema root, then [Draft 4](#) will be used.
- If there is a **"\$schema"**: **"http://json-schema.org/draft-06/schema"** property in the schema root, then [Draft 6](#) will be used.
- If there is a **"\$schema"**: **"http://json-schema.org/draft-07/schema"** property in the schema root, then [Draft 7](#) will be used.
- If there is a **"\$schema"**: **"http://json-schema.org/draft/2019-09/schema"** property in the schema root, then [Draft 2019-09](#) will be used.
- If there is a **"\$schema"**: **"http://json-schema.org/draft/2020-12/schema"** property in the schema root, then [Draft 2020-12](#) will be used.
- If there is a **"\$schema"** property in the schema root, but with a different draft value, then an error will be displayed (*"could not determine version"*).
- If none of these are found, then it is validated as a simple JSON instance.
- You could also select the **JSON Schema Validator** in a [JSON validation scenario \(on page 367\)](#) and it will use the version specified in the JSON Schema, or if a version is not specified, the [JSON Schema draft-04](#) will be used.

For information about how to associate a JSON Schema for the purposes of validation, see [Associating a JSON Schema Through a Validation Scenario \(on page 373\)](#).

For information about using a JSON Schema to validate documents, see [Validating JSON Documents Against JSON Schema or Schematron \(on page 364\)](#).

Draft 2019-09 and Draft 2020-12 Validator Limitations

The JSON Schema Validator handles all the newly introduced keywords in Draft 2019-09 and Draft 2020-12 specifications. However, there are still some limitations:

1. The keywords **"\$recursiveRef"** and **"\$recursiveAnchor"** (Draft 2019-09) are not supported. This is also indicated by a validation warning.
2. The keyword **"\$dynamicRef"** has the same functionality as **"\$ref"**, and **"\$dynamicAnchor"** (Draft 2020-12) is not supported. This is also indicated by a validation warning.
3. The keywords **"unevaluatedProperties"** / **"unevaluatedItems"** can "see through" the subschemas of adjacent keywords **"if"**, **"then"**, **"else"**, but do not know about successfully validated *properties* / *items*. This means that all the *properties* / *items* defined by those subschemas are considered evaluated.
4. The keywords **"unevaluatedProperties"** / **"unevaluatedItems"** can "see through" the nested subschemas of adjacent keywords **"oneOf"**, **"anyOf"**, **"allOf"**, but all the *properties* / *items* defined by those subschemas are considered evaluated, regardless of other nested restrictions.

Syntax Highlighting in JSON Schema Documents

Oxygen JSON Editor supports syntax highlighting in **Text** mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for JSON Schema files, follow these steps:

1. Open the **Preferences** dialog box (**Options > Preferences**) (on page 74).
2. Go to **Editor > Syntax Highlight** (on page 151).
3. Select and expand the **JSON Schema** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

Related Information:

[Syntax Highlight Preferences \(on page 151\)](#)

Flatten JSON Schema

Oxygen JSON Editor includes a **Flatten Schema** action that is available in the contextual menu when editing in **JSON Schema Design mode** (on page 396) and in the Text mode. It allows you to flatten an entire hierarchy of JSON schemas. Starting with the main JSON schema, Oxygen JSON Editor calculates its hierarchy by processing the `$ref` keys and determining all the other referenced schema files. This means that the flattened JSON schema is obtained by recursively adding the components of all referenced schemas into the main one, and by updating all the `$ref` keys to point to the components from the resulting schema.

Resolving schema references is done through [XML Catalogs](#). That means that the sub-schemas referenced through URIs and not mapped accordingly cannot not be parsed for integration into the resulting flattened schema, affecting its logic.

The **Flatten Schema** action opens a small dialog box that allows you to configure the operation by choosing the resulting schema name and the directory to save it, and opting whether to open the resulting schema in the editing area after the operation completes.

Editing JSON Lines Documents

Oxygen JSON Editor includes some basic support for working with **JSON Lines** documents. *JSON Lines* is a convenient format for storing structured data that may be processed one record at a time.

Editing JSON Lines Documents

You can edit JSON Lines documents in the **Text** mode editor in Oxygen JSON Editor and you have access to its various features and actions that are common for basic text files.

Validation

Validation support is available for JSON Lines documents if you have associated a schema through a configured validation scenario.

Content Completion

Content completion support is also available for JSON Lines documents if you have associated a schema through a configured validation scenario.

Editing YAML Documents

This section discusses the various features that are included in the Oxygen JSON Editor YAML Editor and how to use them.

Resources

For more information about the YAML (and JSON) support in Oxygen JSON Editor, see the following resources:

- [Video: YAML Support in Oxygen](#)
- [Webinar: JSON and JSON Schema Support in Oxygen](#)

YAML Editor

Oxygen JSON Editor includes a specialized YAML editor with various editing features for files that have the `yaml/ymL` file extension. It also includes a document template to help you get started with YAML documents. The template is called **YAML** and it can be found in the **New Document** folder in the **New document wizard** ([on page 225](#)).



Tip:

You can experiment with a sample of a YAML file available at: `[OXYGEN-INSTALL-DIR]/samples/yaml/personal.yaml`.

Text Mode Editor

When editing YAML documents in the **Text** editing mode, the usual text editing actions are available, along with other YAML editor-specific actions, including:

- [Syntax highlighting](#) ([on page 430](#))
- [Automatic validation](#) ([on page 425](#))
- [Formatting and indenting](#) ([on page 430](#))

Validating YAML Documents

Automatic Validation

Oxygen JSON Editor includes built-in validation engine for YAML documents to help you keep them well-formed. YAML documents are validated automatically as you type and the validation engine detects syntax errors (such as improper indentation or duplicate keys), based on the [YAML specification](#). The built-in validation also works on files that consist of multiple YAML documents.

Manual Validation Actions

To manually validate the currently edited YAML document, use one of the following actions:

Validate

Available from the  ▾ **Validation** drop-down menu on the toolbar, the **Document > Validate** menu, or from the **Validate** submenu when invoking the contextual menu on one or more YAML documents in the **Project view** (*on page 251*). The validation is done based on the [YAML specification](#).

Validate with

Available from the  ▾ **Validation** drop-down menu on the toolbar or the **Document > Validate** menu. This action opens a dialog box that allows you to specify a JSON Schema for validating the current YAML document (it also works on files that consist of multiple YAML documents).

Check Well-Formedness ()

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** (*on page 251*). This action checks the document for syntax errors to make sure it is well-formed.

Validate with Schema

Available from the **Validate** submenu when invoking the contextual menu on one or more YAML documents in the **Project view** (*on page 251*). This action opens a dialog box that allows you to specify a JSON Schema to be used for the validation.

Batch Validation

The built-in validation engine can also be used to batch validation multiple YAML files at once by selecting multiple files in the **Project view**, right-click, and select **Validate >  Check Well-Formedness** or **Validate >  Validate**. This automatically validates the selected files using the built-in YAML validation engine, based on the [YAML specification](#).

Creating a YAML Validation Scenario

The built-in *YAML Validator* engine that can be specified in a validation scenario to validate YAML documents. In this case, the validation is done against a specified JSON Schema.

Creating a YAML Validation Scenario

To create a validation scenario, follow these steps:

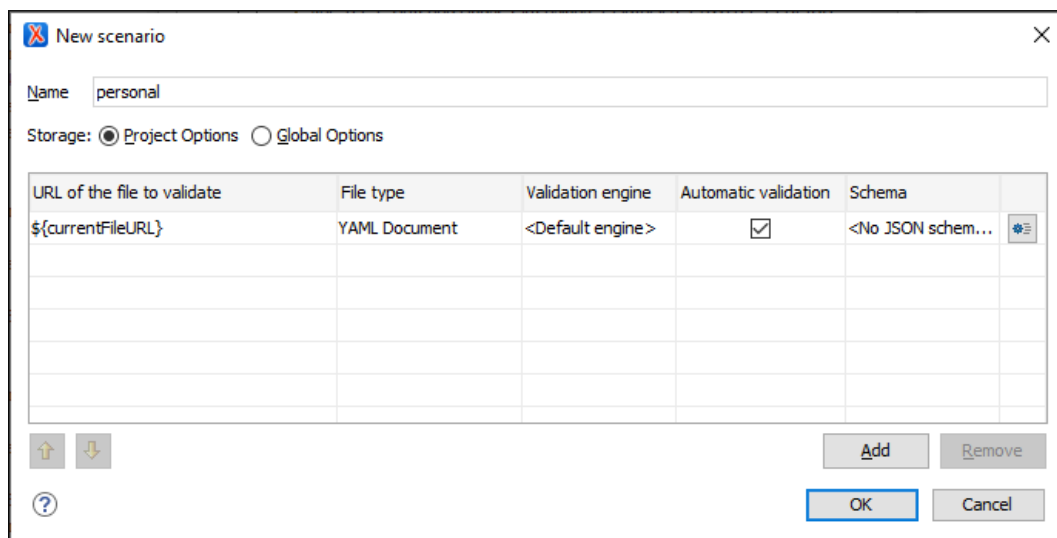
1. Select the  **Configure Validation Scenario(s)** action in one of the following ways:
 - From the  **Validation** toolbar drop-down menu.
 - From the **Document > Validate** menu.
 - From the **Validate** submenu, when invoking the contextual menu on a file in the **Project view** (*on page 251*).

Step Result: The **Configure Validation Scenario(s)** dialog box is displayed.

2. Click the **New** button.

Step Result: A validation scenario configuration dialog box is displayed.

Figure 119. Validation Scenario Configuration Dialog Box



New scenario

Name:

Storage: ☒ Project Options ☐ Global Options

URL of the file to validate	File type	Validation engine	Automatic validation	Schema
\${currentFileURL}	YAML Document	<Default engine>	☒	<No JSON schem... 

  Add Remove

 OK Cancel

This scenario configuration dialog box allows you to configure the following information and options:

Name

The name of the validation scenario.

Storage

You can choose between storing the scenario in the **Project Options** (*on page 640*) or **Global Options** (*on page 639*).

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 \(on page 197\)](#).

Figure 120. Insert an 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

File type

The type of the document that is validated in the current validation unit. Oxygen JSON Editor automatically selects the file type depending on the value of the **URL of the file to validate** field.

Validation engine

For YAML documents, the built-in *YAML Validator* engine (**Default engine**) is used.

Automatic validation

If this option is selected, the validation operation defined by this row is also applied by the automatic validation feature. If the **Automatic validation** feature is disabled in the [Document Checking preferences page \(on page 153\)](#), then this option is ignored, as the preference setting has a higher priority.

Schema

Displays the specified schema.

Specify Schema

Opens the **Specify Schema** dialog box that allows you to set a schema to be used for validating YAML documents.

↑ Move Up

Moves the selected scenario up one spot in the list.

↓ Move Down

Moves the selected scenario down one spot in the list.

Add

Adds a new validation unit to the list.

Remove

Removes an existing validation unit from the list.

3. Configure any of the existing validation units according to the information above. You can use the buttons at the bottom of the table to add, remove, or move validation units.
4. Click **OK**.

Result: 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 click the **Apply associated** button to run the validation scenario.

Related Information:

[Associating a JSON Schema Directly in YAML Documents \(on page 428\)](#)

Associating a JSON Schema Directly in YAML Documents

Associate Schema Action

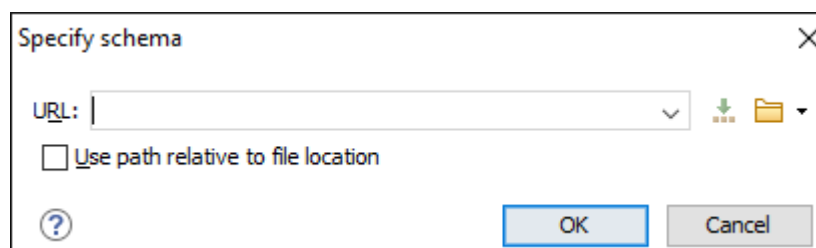
The schema used by the validation engine can be associated with the current document by using the  **Associate Schema** action. The association can specify a relative file path or a URL of the schema.

To associate a JSON Schema to the current YAML document, follow these steps:

1. Select the  **Associate Schema** action from the toolbar (or **Document > Schema** menu).

Step Result: The **Associate Schema** dialog box is displayed:

Figure 121. Associate Schema Dialog Box



This dialog box contains the following options for YAML documents:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S).
 - **Use path relative to file location** - Select this option if the YAML instance document and the associated schema contain relative paths. The location of the schema file is inserted in the YAML 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.
2. Select the JSON Schema that will be associated with the YAML document.
 3. Click **OK**.

Result: A `$schema` property is added at the beginning of the document with its value set to the specified URL. If the document already contained a schema association, the old association will be replaced with the new one.



Tip:

To quickly open the schema used for validating the current document, select the **Open Associated Schema** action from the toolbar (or **Document > Schema** menu).

Related Information:

[Creating a YAML Validation Scenario \(on page 426\)](#)

Content Completion Assistant in YAML

Oxygen JSON Editor includes an intelligent [Content Completion Assistant \(on page 638\)](#) that offers proposals for inserting YAML structures that are valid at the current editing location.

The *Content Completion Assistant* is enabled by default. To disable it, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#), go to **Editor > Content Completion**, and deselect the **Enable content completion** option [\(on page 139\)](#).

Content Completion and the Associated Schema

The *Content Completion Assistant* feature is schema-driven and the list of proposals in the [Content Completion Assistant \(on page 638\)](#) depend on the associated JSON schema. For information about ways to associate a schema to a YAML document, see [Associating a JSON Schema Directly in YAML Documents \(on page 428\)](#).

Using the Content Completion Assistant in YAML

The feature is activated in **Text** mode for YAML documents by pressing **Ctrl + Space** or **Alt + ForwardSlash** (**Command + Option + ForwardSlash** on macOS).

You can navigate through the list of proposals by using the **Up** and **Down** keys on your keyboard. You can also change the size of the documentation window by dragging its top, right, and bottom borders.

To insert the selected proposal, press **Enter** or **Tab**.

Content Completion Options for YAML

The content that is inserted by the content completion mechanism in YAML is generated automatically from the associated schema. You can control the inserted content with options available in the [Options > Preferences > Editor > Content Completion > YAML preferences page \(on page 143\)](#). You can specify whether or not required content, optional content, and additional content should be generated.

Syntax Highlighting in YAML Documents

Oxygen JSON Editor supports syntax highlighting in **Text** mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for YAML files, follow these steps:

1. Open the **Preferences** dialog box ([Options > Preferences](#)) (on page 74).
2. Go to **Editor > Syntax Highlight** (on page 151).
3. Select and expand the **YAML** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

Related Information:

[Syntax Highlight Preferences \(on page 151\)](#)

Folding in YAML Documents

In a large YAML document, the data corresponding to complex keys can be collapsed so that only the needed data remains in focus. The usual *folding features available for XML documents* are also available in YAML documents.

Formatting/Indenting YAML Documents

Oxygen JSON Editor includes support for formatting and indenting YAML documents. You can trigger a format and indent operation for your YAML document using the  **Format and Indent** toolbar button.

Some of the formatting actions that are performed include:

- Indents the document with the indent size specified in the [Editor > Format preferences page \(on page 135\)](#).
- Removes empty lines and extra spaces between keys and values.
- Compacts the string values (for example, *description*) and limits it to 80 characters on a row.

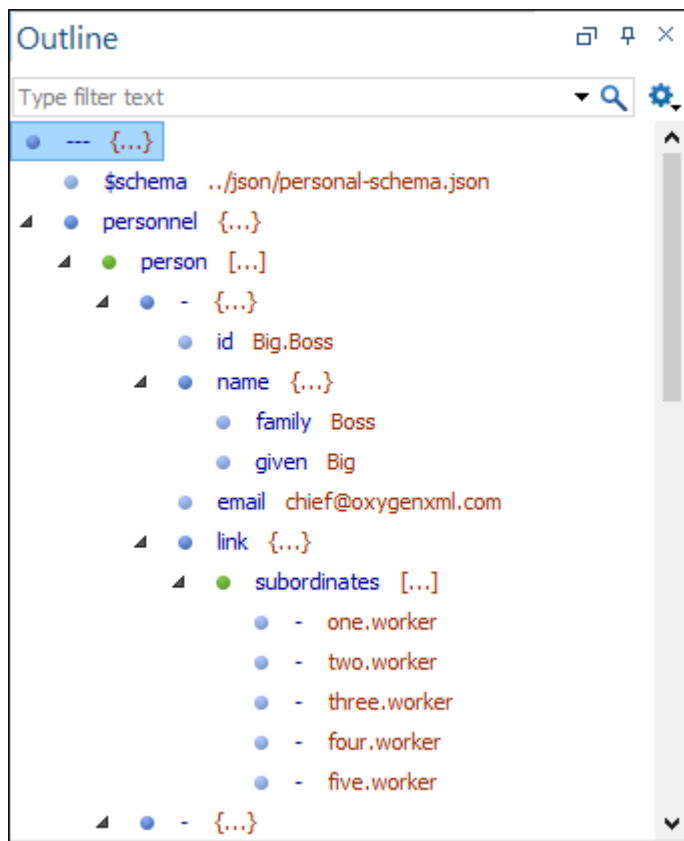
To batch format/indent multiple YAML files, select and right-click the files in the **Project** view, then select

 **Format and Indent Files**.

YAML Outline View

The **Outline** view for YAML documents displays the list of all the components of the document you are editing in a hierarchical (tree-like) representation. It is synchronized with the main editor so you can use the view to navigate to specific parts of the document and move or delete components. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

Figure 122. YAML Outline View



Outline View Features

The **Outline** view allows you to:

- Quickly navigate through the document by selecting nodes in the **Outline** tree.
- 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.

Outline View Interface

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (such as `*` or `?`) and separate multiple patterns with commas.

It also includes a  **Settings** menu in the top-right corner that presents the following options to help you filter the view even further.

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.

Contextual Menu Actions

The following actions are available in the contextual menu of the YAML **Outline** view:



Cut

Cuts the currently selected component.



Copy

Copies the currently selected component.



Paste

Pastes the copied 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.

YAML to JSON Converter

Converting YAML to JSON in Oxygen

Oxygen JSON Editor includes a useful and simple tool for converting YAML files to JSON. It even works on files that consist of multiple YAML documents, each separated by three dashes (---), in which case the conversion creates multiple JSON files with a number in the name.

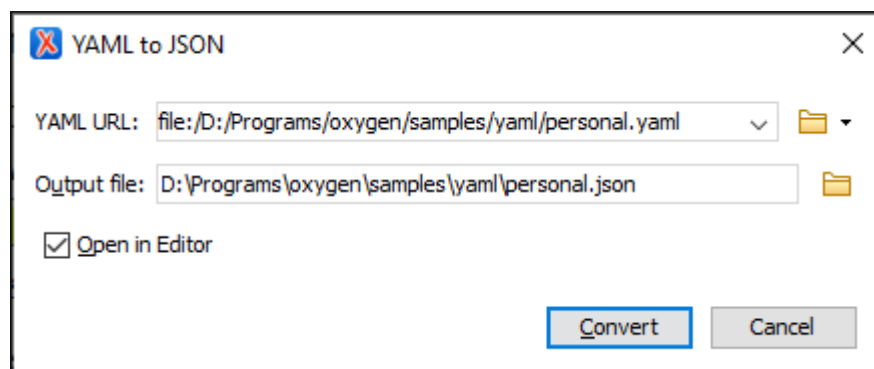
The **YAML to JSON** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

To convert a YAML document to JSON, follow these steps:

1. Select the **YAML to JSON** action from the **Tools > JSON Tools** menu.

The **YAML to JSON** dialog box is displayed:

Figure 123. YAML to JSON Dialog Box



2. Choose or enter the **YAML URL** for the document you want to convert.
3. Choose the path of the **Output file** that will contain the resulting JSON document.
4. **[Optional]** Select the **Open in Editor** option to open the resulting JSON document in the main editing pane.
5. Click the **Convert** button.

Result: The original YAML document is now converted to a JSON document.

Related Information:

[JSON to YAML Converter \(on page 387\)](#)

JSON to YAML Converter

Converting JSON to YAML in Oxygen

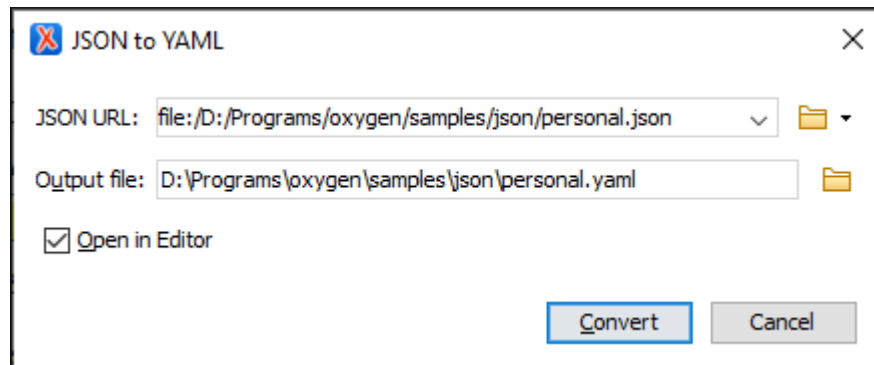
Oxygen JSON Editor includes a useful and simple tool for converting JSON files to YAML. The **JSON to YAML** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

To convert a JSON document to YAML, follow these steps:

1. Select the **JSON to YAML** action from the **Tools > JSON Tools** menu.

The **JSON to YAML** dialog box is displayed:

Figure 124. JSON to YAML Dialog Box



2. Choose or enter the **JSON URL** for the document you want to convert.
3. Choose the path of the **Output file** that will contain the resulting YAML document.
4. **[Optional]** Select the **Open in Editor** option to open the resulting YAML document in the main editing pane.
5. Click the **Convert** button.

Result: The original JSON document is now converted to a YAML document.

Related Information:

[YAML to JSON Converter \(on page 387\)](#)

Contextual Menu Actions in YAML Documents

In addition to the usual text editing actions being available in the YAML editor, it also includes some unique editor-specific actions available in the contextual menu:

 **Cut**,  **Copy**,  **Paste**

Executes the typical editing actions on the currently selected content.

Copy JSON Pointer

Creates a *JSON Pointer* at the current cursor location and copies the expression that denotes the JSON pointer to the system clipboard.

Copy XPath

Copies the XPath expression of the current property from the current editor to the clipboard.

Toggle Line Wrap (**Ctrl + Shift + Y** (**Command + Shift + Y** on macOS))

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.

Toggle Line Comment (**Ctrl + ForwardSlash** (**Command + ForwardSlash** on macOS))

Allows you to comment (or uncomment) the current selection (or current line if no content is selected) in the YAML document you are editing.

Source submenu

This submenu includes the following actions:

To Lower Case

Converts the content selection to lower case characters. This works with contiguous and multiple selections.

To Upper Case

Converts the selected content to upper case characters. This works with contiguous and multiple selections.

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 ()

Converts a sequence of hexadecimal characters to the corresponding [Unicode character \(on page 303\)](#). 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 and the characters they will be converted to:

- `0x0045` will be converted to `É`
- `0x0125` to `ĥ`
- `265` to `ı`
- `2190` to `←`



Note:

For more information about finding the hexadecimal value of a character, see [Finding the Decimal, Hexadecimal, or Character Entity Equivalent \(on page 306\)](#).

Base64 Encode/Decode submenu

This submenu include the following actions for encoding or decoding **base 64** 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.

Decode Selection

Replaces a selection of text with the result of decoding that selection.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Base32 Encode/Decode submenu

This submenu include the following actions for encoding or decoding **base32** schemes:

Import File to Encode and Insert

Encodes a file and then inserts the encoded content into the current document at the cursor position.

Decode Selection and Export to File

Decodes a selection of text from the current document and then exports (saves) the result to another file.

Encode Selection

Replaces a selection of text with the result of encoding that selection.

Decode Selection

Replaces a selection of text with the result of decoding that selection.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Hex Encode/Decode submenu

This submenu include the following actions for encoding or decoding **hex** schemes:

Import File to Encode and Insert

Encodes a file and then inserts the encoded content into the current document at the cursor position.

Decode Selection and Export to File

Decodes a selection of text from the current document and then exports (saves) the result to another file.

Encode Selection

Replaces a selection of text with the result of encoding that selection.

Decode Selection

Replaces a selection of text with the result of decoding that selection.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Join and Normalize Lines (Ctrl + J (Command + J on macOS))

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 + Option + Enter on macOS))

This action has the same result as moving the cursor to the end of the current line and pressing the *ENTER* key.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Go to Definition

Navigates to the definition of the current key (if a JSON schema is associated with the YAML document and that schema contains a definition for the key).

Associating a JSON schema can be done using either of the following methods:

- By [creating a YAML validation scenario \(on page 426\)](#).
- By directly [associating a JSON schema in the YAML document \(on page 428\)](#).



Tip:

Holding down **CTRL** (**Command** on macOS) while hovering over a YAML *key-value* pair reveals a hyperlink that navigates to the key definition in the associated JSON schema (if a definition for that key is found).

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 the 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 \(on page 313\)](#).

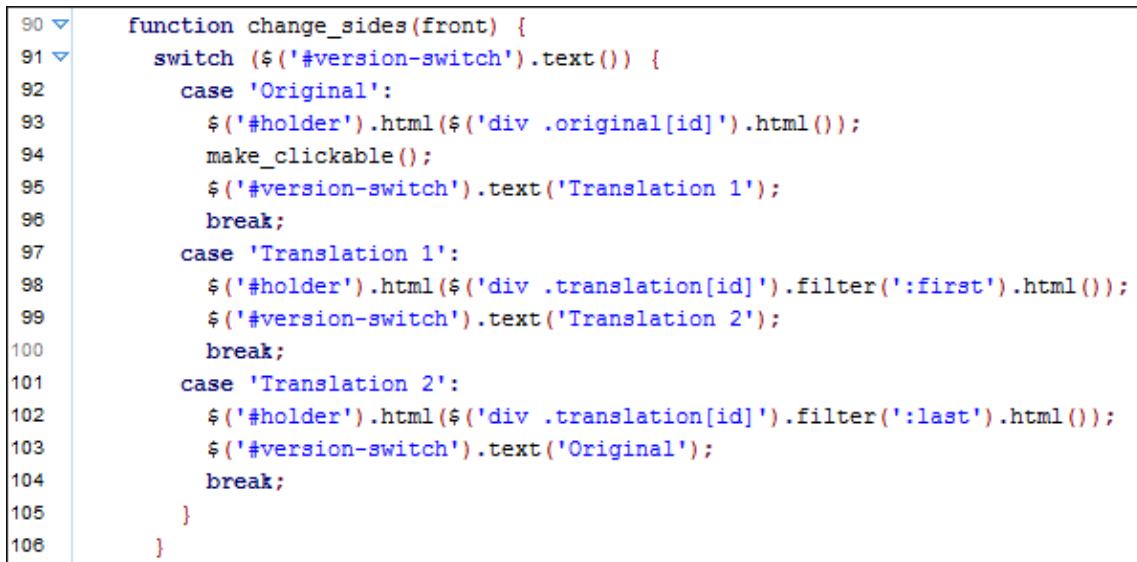
Editing JavaScript Documents

This section explains the features of the Oxygen JSON Editor JavaScript Editor and how you can use them.

JavaScript Editing Actions

Oxygen JSON Editor allows you to create and edit JavaScript files and assists you with useful features such as syntax highlight, content completion, and outline view. To enhance your editing experience, you can select entire blocks (parts of text delimited by brackets) by double-clicking somewhere inside the brackets.

Figure 125. JavaScript Editor Text Mode



```

90  function change_sides(front) {
91      switch ($('#version-switch').text()) {
92          case 'Original':
93              $('#holder').html($('#div .original[id]').html());
94              make_clickable();
95              $('#version-switch').text('Translation 1');
96              break;
97          case 'Translation 1':
98              $('#holder').html($('#div .translation[id]').filter(':first').html());
99              $('#version-switch').text('Translation 2');
100             break;
101          case 'Translation 2':
102              $('#holder').html($('#div .translation[id]').filter(':last').html());
103              $('#version-switch').text('Original');
104              break;
105          }
106      }

```

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 JSON Editor moves the cursor to the matching bracket, highlights its row, and decorates the initial bracket with a rectangle.

**Note:**

A rectangle decorates the opening or closing bracket that matches the current one, at all times.

Source

Allows you to select one of the following actions:

To Lower Case

Converts the selection content to lower case characters.

To Upper Case

Converts the selection content to upper case characters.

Capitalize Lines

Converts to upper case the first character of every selected line.

Join and Normalize Lines

Joins all the rows you select to one row and normalizes the content.

Insert new line after

Inserts a new line after the line at the cursor position.

Modify all matches

Use this option to modify (in-place) all the occurrences of the selected content. When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Open

Allows you to select one of the following actions:

- **Open File at Cursor** - select this action to open the source of the file located at the cursor position
- **Open File at Cursor in System Application** - select this action to open the source of the file located at the cursor position with the application that the system associates with the file

Compare

Select this option to open the **Compare Files** tool to compare the file you are editing with a file you choose in the dialog box.

Folding

When you invoke the contextual menu from the *folding (on page 638)* triangles in the stripe on the left side of the editor, the following actions are available:



Collapse Other Folds

Folds all the elements except the current element.



Collapse Child Folds

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

Unfolds all elements in the current document.

Validating JavaScript Files

You have the possibility to validate the JavaScript document you are editing. Oxygen JSON Editor uses the Mozilla Rhino library for validation. For more information about this library, go to <https://github.com/mozilla/rhino>. 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 JSON Editor can validate a JavaScript document both on-request and automatically.

Content Completion in JavaScript Documents

When you edit a JavaScript document, the *Content Completion Assistant (on page 638)* presents you a list of the elements you can insert at the cursor position. It can be manually activated with the **Ctrl + Space** shortcut.

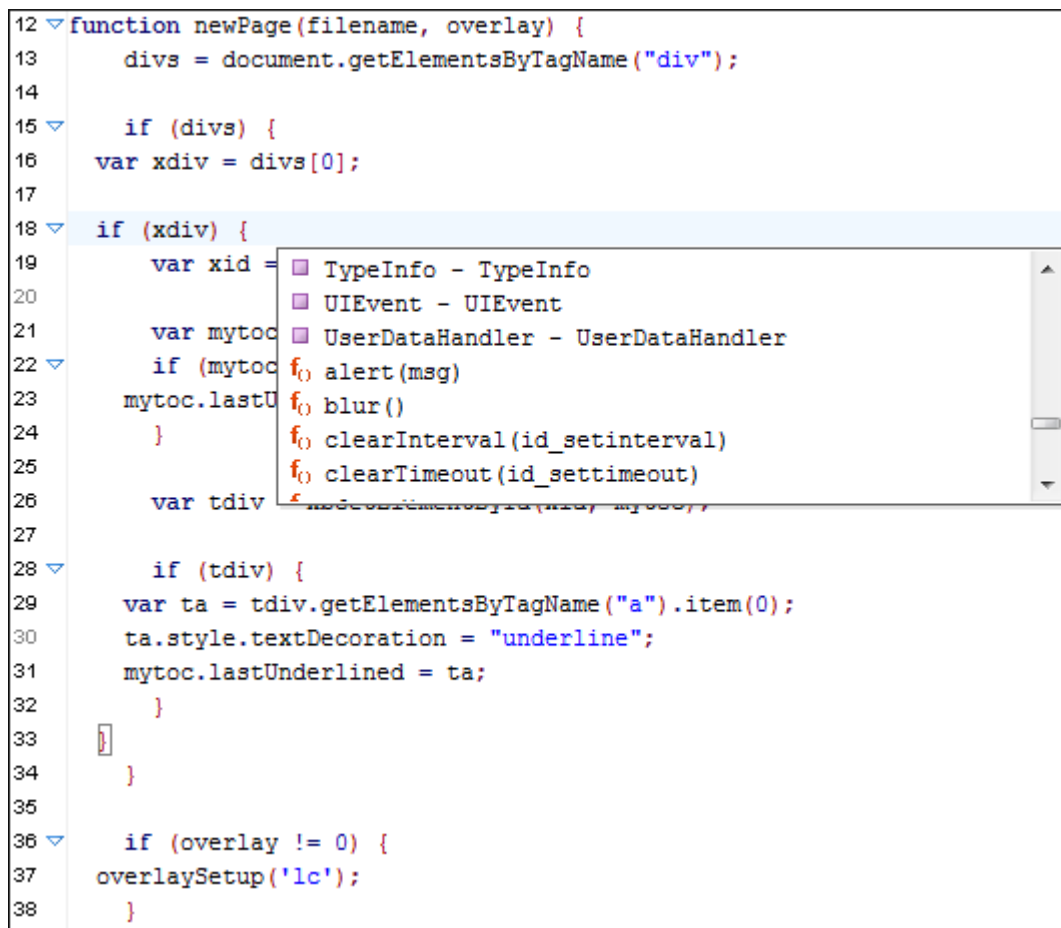
For an enhanced assistance, JQuery methods are also presented. The following icons decorate the elements in the content completion list of proposals depending on their type:

-  - function
-  - variable
-  - object
-  - property
-  - method



Note:

These icons decorate both the elements from the content completion list of proposals and from the **Outline view (on page 443)**.

Figure 126. JavaScript Content Completion Assistant

The *Content Completion Assistant* collects:

- Method names from the current file and from the library files.
- Functions and variables defined in the current file.

If you edit the content of a function, the content completion list of proposals contains all the local variables defined in the current function, or in the functions that contain the current one.

Syntax Highlighting in JavaScript Documents

Oxygen JSON Editor supports syntax highlighting in **Text** mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for JavaScript files, follow these steps:

1. Open the **Preferences** dialog box (**Options > Preferences**) (on page 74).
2. Go to **Editor > Syntax Highlight** (on page 151).
3. Select and expand the **JavaScript** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

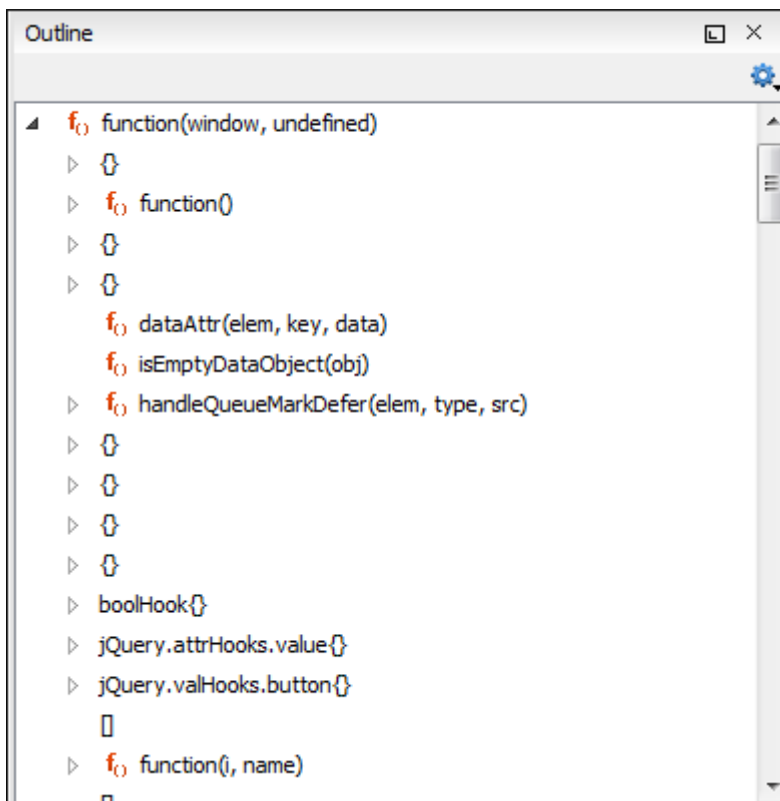
Related Information:

[Syntax Highlight Preferences \(on page 151\)](#)

JavaScript Outline View

Oxygen JSON Editor present a list of all the components of the JavaScript document you are editing in the **Outline** view. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

Figure 127. JavaScript Outline View



The following icons decorate the elements in the **Outline** view depending on their type:

-  - function
-  - variable
-  - object
-  - property
-  - method

The contextual menu of the JavaScript **Outline** view contains the usual  **Cut**,  **Copy**,  **Paste**, and  **Delete** actions. From the  **Settings** menu, you can select the **Update selection on cursor move** option to synchronize the **Outline** view with the editing area.

Editing HTML Documents

Oxygen JSON Editor provides a special framework for editing HTML files (*html* or *htm* file extensions) with a variety of specialized editing features, including validation, content completion, syntax highlighting, HTML-specific actions, and more.

Resources

For more information about the HTML support in Oxygen JSON Editor, see the following resources:

- Webinar: [HTML5 Support in Oxygen](#).
- Video: [HTML Support in Oxygen](#).

HTML Editor

Oxygen JSON Editor includes a specialized HTML editor and various editing features for files that have the `html` or `htm` file extensions. The encoding is detected automatically based on the value specified in the `@charset` attribute of the `<meta>` element.

**Note:**

If an HTML document has an XHTML namespace, or there is an XSD schema declared, or there is a PUBLIC ID specified in a DOCTYPE, or there is a SYSTEM ID with a value other than *"about:legacy-compat"*, then the document will be opened as an XHTML file.

New Document Template

Oxygen JSON Editor includes a new document template to help you get started creating HTML content. It is available when creating [new documents from templates \(on page 225\)](#) and can be found in the **New Document** folder or by typing *html* in the search field.

Text Mode Editor

You can edit HTML files in the **Text** editing mode using all of its useful features. It includes content completion based on a special HTML schema, [syntax highlighting \(on page 447\)](#), a specialized [Outline view \(on page 448\)](#) that presents the structure, [folding support \(on page 447\)](#), and more.

HTML documents support formatting and indenting single or multiple documents to make them more readable. The formatting works even if the document is not XML well-formed and it also works on embedded CSS or JavaScript code. The HTML formatting details are similar to those for XML documents.

HTML-Specific Contextual Menu Actions

There are some specialized actions (available in the contextual menu when you right-click anywhere in the current HTML document) that invoke features unique to HTML documents. These contextual menu actions include:



View in Browser/System Application

Opens the HTML document in your default browser.

Minify HTML

Compresses the HTML document by removing unnecessary white spaces, without affecting the functionality of the document, but significantly reducing the loading time in Web browsers.

HTML to XML Well-formed

Converts the currently edited HTML document to be XML well-formed. This means that unclosed tags will be properly closed, unquoted attribute values will be quoted, and more. This is helpful if, for example, you use XSLT stylesheets while applying transformations on HTML documents (since the transformation will require the HTML document to be XML well-formed).

HTML Validation

Oxygen JSON Editor includes a built-in default validator used for validating HTML documents. It is based upon the W3C HTML Validator and the HTML documents are validated against the [W3C HTML5 specification](#). The validator in Oxygen JSON Editor only supports HTML5 structure. It presents the errors in the editor similar to XML documents. It also checks the embedded CSS content and the warnings and errors are presented similar to the [CSS editor \(on page 352\)](#).

Validating HTML Against a Schematron

It is also possible to validate HTML documents against a Schematron schema. Besides the default HTML validator, Oxygen JSON Editor also includes a built-in *HTML Schematron Validator* engine. There are several ways to validate an HTML document against a Schematron:

- **Configure a Validation Scenario** - You can create or edit a validation scenario, change the **File type** column to *HTML Document*, change the **Validation engine** column to *HTML Schematron Validator*, and specify the Schematron document in the **Schema** column.
- **Manually Validate a Single Document** - You can use the **Validate with** action from the  ▾ **Validation** drop-down menu on the toolbar. This opens a dialog box where you can specify the Schematron document to validate the current document against.
- **Batch Validate Multiple Documents** - You can select multiple HTML documents in the **Project** view, right-click, and use the **Validate with schema** action from the **Validate** submenu. This opens a dialog box where you can specify the Schematron document to validate the selected documents against.



Notes:

- The Schematron must use the HTML5 namespace to reference the elements from the instance.
- Implicit HTML elements (i.e. `<html>`, `<body>`, `<tbody>`) must be included in an XPath expression in the Schematron document, even if they are missing from the HTML document.

**Tip:**

The Oxygen JSON Editor installation directory includes a **samples** folder that contains numerous sample files to help you learn about features, certain file types, and XML technologies. For example, inside the **samples** folder, there is an **html** folder with a **schematron** subfolder where you can find some samples that illustrate HTML validation against a Schematron schema.

Validating HTML Against Other Types of Schema

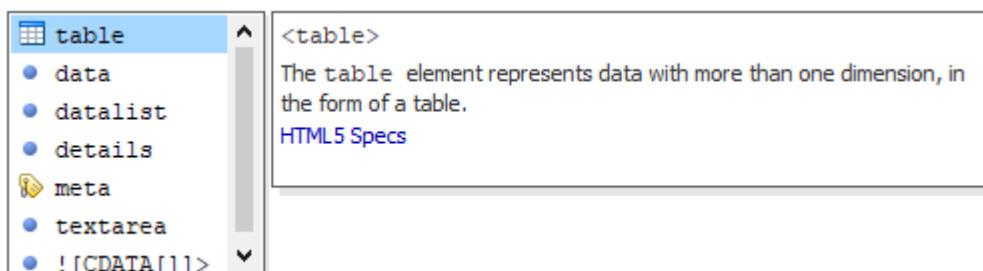
If your HTML document is XML well-formed, you could also configure a validation scenario to validate it as an XML document against other types of schemas. You would create or edit a validation scenario, make sure the **File type** column is set to *XML Document*, select the appropriate **Validation engine**, and specify the schema document in the **Schema** column.

HTML Content Completion Assistant

Oxygen JSON Editor includes an intelligent *Content Completion Assistant* (on page 638) that offers proposals for inserting HTML structures that are valid at the current editing location. Content completion is even available for CSS and JavaScript code that is embedded in an HTML document.

The *Content Completion Assistant* is enabled by default. To disable it, open the **Preferences** dialog box (**Options > Preferences**) (on page 74), go to **Editor > Content Completion**, and deselect the **Enable content completion** option (on page 139).

Figure 128. Content Completion Assistant in HTML



Using the Content Completion in HTML

For HTML documents, the *Content Completion Assistant* uses a built-in schema and the list of proposals depend on the RELAX NG schema specified in the HTML framework. Using the content completion feature is the same as with any other XML document.

Code Templates in the Content Completion

Oxygen JSON Editor includes a set of built-in code templates for HTML documents that can be selected from the *Content Completion Assistant*. The code templates are displayed with a  symbol in the content completion list. You can also define your own code templates and share them with others.

Content Completion for XPath Expressions

When entering XPath expressions in the **XPath** toolbar or **XPath Builder** view, the *Content Completion Assistant* is available as you type to help you compose query patterns.

Syntax Highlighting in HTML Documents

Oxygen JSON Editor supports syntax highlighting in **Text** mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

For HTML documents, it handles attributes without quotes, unclosed or void elements, and it also offers highlighting for embedded CSS or JavaScript content.

To customize the colors or styles used for the syntax highlighting colors for HTML files, follow these steps:

1. Open the **Preferences** dialog box (**Options > Preferences**) ([on page 74](#)).
2. Go to **Editor > Syntax Highlight** ([on page 151](#)).
3. Select and expand the **XHTML** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

Related Information:

[Syntax Highlight Preferences \(on page 151\)](#)

Folding in HTML

In a large HTML document, elements can be collapsed so that only the needed data remains in focus.

Minifying HTML Documents

Minification (or compression) of an HTML document is the practice of removing unnecessary white spaces, without affecting the functionality of the document, but significantly reducing the loading time in Web browsers. While a minified HTML document gains in terms of execution performance, it is more difficult to read.

To minify an HTML document, right-click anywhere in the editor for an HTML document that is open in **Text** mode (or right-click an HTML document in the **Project** view and select the **Minify HTML** action. This opens a dialog box with the following options:

Output file

Use this option to set the name and location of the resulting compressed/minified HTML document.

Remove comments

If selected (default), all the HTML comments and also the comments from embedded CSS or JavaScript code blocks will be removed from the resulting output file.

Compress on a single line

If selected (default), the resulting output file will consist of a single line, as all the 'new line' characters from the source document are removed.

Open output file in editor

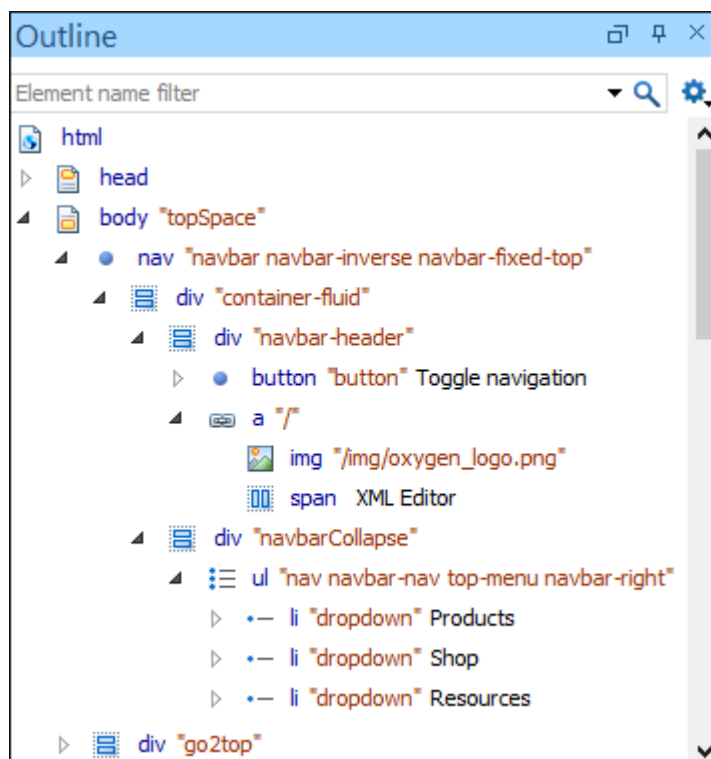
If selected (default), the resulting output file will be opened in Oxygen JSON Editor.

When you click **OK**, the resulting HTML document is a compressed version of the original file for the purpose of enhanced performance, while losing some readability. The source HTML document is not affected.

HTML Outline View

The **Outline** view for HTML documents displays the structure of the HTML document you are editing. By default, it is displayed on the left side of the editor. In addition to the normal features available in the **Outline** view for XML documents, the HTML **Outline** view also handles void elements, elements that are not closed, or attributes without quotes, and presents the tree structure of the HTML document correctly.

Figure 129. HTML Outline View



Querying HTML Documents with XPath

Oxygen JSON Editor provides an **XPath** toolbar that makes it easy to quickly query HTML documents using XPath expressions. You can also use the dedicated **XPath Builder** view ([on page 474](#)) that allows you to compose more complex XPath expressions and execute them over HTML documents (even if they are not

well-formed according to XML standards). Both the **XPath** toolbar and **XPath Builder** view offer content completion as you type to help you compose expressions.

Editing Markdown Documents

Markdown was created as a lightweight markup language with plain text formatting syntax designed to provide syntax that is very easy to read and write, and to convert it to HTML and other formats. It is often used by content contributors who want a quick and easy way to write content without having to take their fingers off the keyboard and without having to learn numerous codes or shortcuts, and it can easily be shared interchangeably between virtually any types of contributor and system.

Oxygen JSON Editor provides a built-in Markdown editor that allows you to convert Markdown syntax to HTML or DITA and it includes a preview panel to help you visualize the final output. Aside from the plain text syntax that is common among most Markdown applications, the editor in Oxygen JSON Editor also integrates many other powerful features that content authors are accustomed to using for other types of documents. Some of these additional unique features include:

- Additional toolbar and contextual menu actions.
- Automatic validation to help keep the syntax valid.
- Dedicated syntax highlighting to make Markdown documents even easier to read and write.
- Specialized syntax rules to combine popular syntax features from several specifications.

Resources

For more information about editing Markdown documents in Oxygen JSON Editor, see our webinar: [Oxygen Markdown Support](#).

Markdown Editor

Oxygen JSON Editor provides an intuitive, dynamic, and easy-to-use Markdown editor.

Markdown Text Editor Pane (Left Side)

The Markdown editor is a simple text editor that is refined to accept Markdown syntax. At the same time, you still have many of the actions, options, and features that you are used to when editing any other type of document in Oxygen JSON Editor.

The features of this special editor that are unique for Markdown documents include:

- **Unique Markdown Syntax Rules** - The Markdown editor in Oxygen JSON Editor uses [an integration of rules \(on page 457\)](#) that combine rules from common default Markdown syntax along with many of the rules used in the GitHub Flavored Markdown syntax.
- **Syntax Highlighting** - The Oxygen JSON Editor syntax highlighting feature is integrated into the Markdown text editor to make it easier to read and write Markdown syntax. You can even [customize the colors and styles for the syntax highlighting \(on page 456\)](#).

- **Automatic Spell Checking** - The Markdown editor supports the Oxygen JSON Editor [automatic spell checking feature \(on page 297\)](#) that reports possible misspelled words as you type. You simply need to select the **Automatic spell check** option in the **Spell Check** preferences page ([on page 156](#)), then click the **Select editors** button and select **Markdown Editor**.
- **Helpful Toolbar and Contextual Menu Actions** - A [variety of unique actions \(on page 450\)](#) are available from the toolbar to help you insert proper Markdown syntax.

Related information

[Markdown Editor Syntax Rules and Specifications \(on page 457\)](#)

[Actions Available in the Markdown Editor \(on page 450\)](#)

[Creating New Markdown Documents \(on page 450\)](#)

Creating New Markdown Documents

To create a new Markdown document in Oxygen JSON Editor, follow these steps:

1. Click the  **New** button on the toolbar or select **File > New**.
2. Select the  **Markdown** document template (in the **New Document** folder).
3. Click the **Create** button.

Result: A new Markdown document is created and it is opened in the specialized [Markdown Editor \(on page 449\)](#).

Related Information:

[Markdown Editor \(on page 449\)](#)

Actions Available in the Markdown Editor

Aside from the actions that are available in Oxygen JSON Editor for any type of document (such as the actions in the various menus and the common sections of the toolbar), a variety of unique actions are also available in the Markdown editor, from the toolbar and contextual menu.

Toolbar Actions

The following default actions are available on the Markdown toolbar when editing Markdown documents:

H₁ Header (1st Level)

Inserts an [atx-style first-level header \(on page 457\)](#) at the cursor position.

H₂ Header (2st Level)

Inserts an [atx-style second-level header \(on page 457\)](#) at the cursor position.

H₃ Header (3rd Level)

Inserts an [atx-style third-level header \(on page 457\)](#) at the cursor position.

— Horizontal Rule

Inserts a [horizontal rule \(on page 458\)](#) at the cursor position.

B Bold (Strong)

Marks the selected text with [bold \(on page 459\)](#).

I Italic (Emphasis)

Marks the selected text with [italics \(on page 458\)](#).

~~Strikethrough~~

Marks the selected text with a [strikethrough \(on page 459\)](#).

`{ }` Code Block

Inserts (or surrounds selected text in) a [codeblock \(on page 463\)](#).

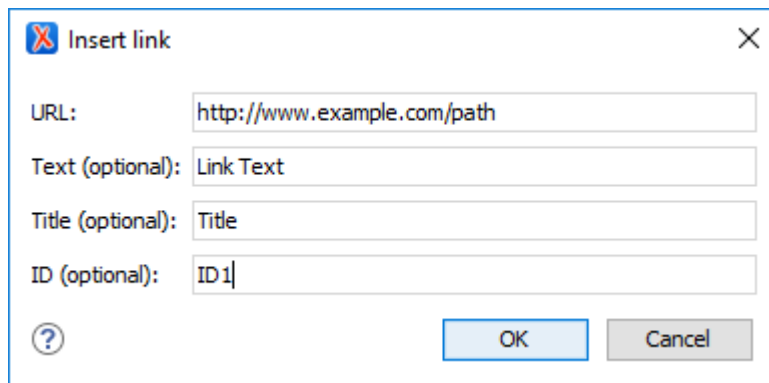
”” Blockquote

Inserts a [blockquote \(on page 462\)](#) at the cursor position.

Insert Link

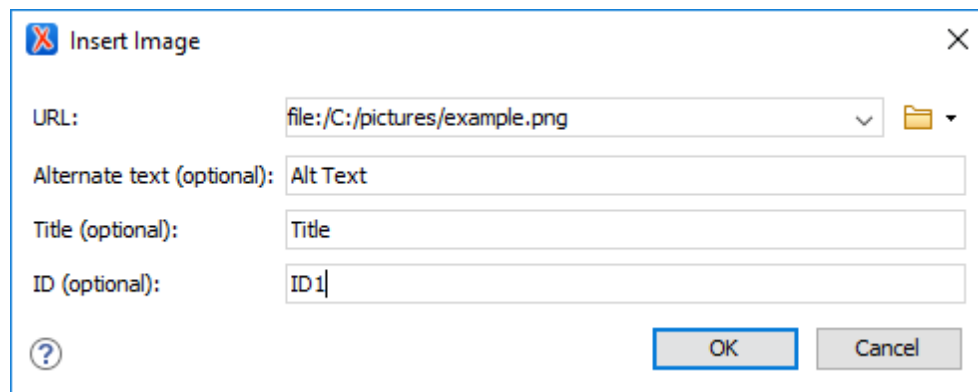
Opens the **Insert Link** dialog box that allows you to define a [link \(on page 459\)](#) to insert at the cursor position.

Figure 130. Insert Link Dialog Box



Insert Image

Opens the **Insert Image** dialog box that allows you to define an [image \(on page 461\)](#) to insert at the cursor position. You can type the URL of the image you want to insert or use browsing actions in the  ▾ **Browse** drop-down menu.

Figure 131. Insert Link Dialog Box

Insert Ordered List

Inserts an [ordered list \(on page 465\)](#) at the cursor position. Three child list items are also automatically inserted by default. You can also use this action to convert selected content to an ordered list.

Insert Unordered List

Inserts an [unordered list \(on page 464\)](#) at the cursor position. Three child list items are also automatically inserted by default. You can also use this action to convert selected content to an unordered list.

Insert Task List

Inserts a [task list \(on page 466\)](#) at the cursor position. Three child list items are also automatically inserted by default. You can also use this action to convert selected content to a task list.

Insert Table

Inserts a [table \(on page 467\)](#) at the cursor position.

Contextual Menu Actions

The following default actions are available in the contextual menu when editing Markdown documents:

Cut, **Copy**, **Paste**

Use these actions to execute the typical editing actions on the currently selected content.

Source submenu

This submenu includes the following actions:

To Upper Case

Converts the content selection to upper case characters.

To Lower Case

Converts the content selection to lower case characters.

Capitalize Lines

It capitalizes the first letter found on every new line that is selected. Only the first letter is affected, the rest of the line remains the same. If the first character on the new line is not a letter then no changes are made.

Convert Hexadecimal Sequence to Character ()

Converts a sequence of hexadecimal characters to the corresponding [Unicode character \(on page 303\)](#). 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 and the characters they will be converted to:

- `0x0045` will be converted to `E`
- `0X0125` to `ĥ`
- `265` to `ı`
- `2190` to `←`



Note:

For more information about finding the hexadecimal value of a character, see [Finding the Decimal, Hexadecimal, or Character Entity Equivalent \(on page 306\)](#).

Base64 Encode/Decode submenu

This submenu includes the following actions for encoding or decoding **base 64** 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.

Decode Selection

Replaces a selection of text with the result of decoding that selection.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Base32 Encode/Decode submenu

This submenu includes the following actions for encoding or decoding **base32** schemes:

Import File to Encode and Insert

Encodes a file and then inserts the encoded content into the current document at the cursor position.

Decode Selection and Export to File

Decodes a selection of text from the current document and then exports (saves) the result to another file.

Encode Selection

Replaces a selection of text with the result of encoding that selection.

Decode Selection

Replaces a selection of text with the result of decoding that selection.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Hex Encode/Decode submenu

This submenu includes the following actions for encoding or decoding **hex** schemes:

Import File to Encode and Insert

Encodes a file and then inserts the encoded content into the current document at the cursor position.

Decode Selection and Export to File

Decodes a selection of text from the current document and then exports (saves) the result to another file.

Encode Selection

Replaces a selection of text with the result of encoding that selection.

Decode Selection

Replaces a selection of text with the result of decoding that selection.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Join and Normalize Lines (Ctrl + J (Command + J on macOS))

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 + Option + Enter on macOS))

This action has the same result as moving the cursor to the end of the current line and pressing the *ENTER* key.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Open submenu

The following actions are available in this submenu:

Open File at Cursor

Opens the file at the cursor position in a new panel. If the file path represents a directory path, it will be opened in system file browser. If the file at the specified location does not exist, an error dialog box is displayed and it includes a **Create new file** button that starts the **New document** wizard. This allows you to choose the type or the template for the file. If the action succeeds, the file is created with the referenced location and name and is opened in a new editor panel.

Open File at Cursor in System Application

Opens the file (identified by its link) or web page (identified by a web link) found at the 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 \(on page 313\)](#).

Show/Hide Preview

A toggle action that shows or hides the *Preview* pane.

Export as DITA Topic

Converts the current Markdown document into a DITA topic.

Export as XDITA Topic

Converts the current Markdown document into a Lightweight DITA XML topic.

Export as HTML

Converts the current Markdown document into an XHTML document.

Print (Available in the *Preview* pane)

Opens a page setup dialog box that allows you to configure printing options for the current document.

Related information

[Markdown Editor \(on page 449\)](#)

[Markdown Editor Syntax Rules and Specifications \(on page 457\)](#)

Syntax Highlighting in the Markdown Editor

Oxygen JSON Editor supports syntax highlighting in the Markdown editor to make it easier to read the semantics of the structured content by displaying each type of XML code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for Markdown documents, follow these steps:

1. Open the **Preferences** dialog box (**Options > Preferences**) [\(on page 74\)](#).
2. Go to **Editor > Syntax Highlight** [\(on page 151\)](#).

3. Select and expand the **Markdown** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

Related Information:

[Syntax Highlight Preferences \(on page 151\)](#)

[Markdown Editor \(on page 449\)](#)

Markdown Editor Syntax Rules and Specifications

The Markdown editor in Oxygen JSON Editor uses rules that were integrated from the most common set of [default Markdown syntax rules](#) along with many of the [GitHub Flavored Markdown rules](#).

This topic lists the Oxygen JSON Editor implementation of the most commonly used syntax rules.

Headers

The Markdown editor supports two styles of headers, *Setext* and *Atx*.

• Setext Style

Setext-style headers are underlined using equal signs (for first-level headers) and dashes (for second-level headers). Any number of equal signs or dashes will result in the same output.

Example: Setext Style Headers

```
First-Level Header (H1)
=====

Second-Level Header (H2)
-----
```

• Atx Style

Atx-style headers use 1-6 hash characters at the start of the line, corresponding to header levels 1-6. Optionally, you may close atx-style headers. This is purely cosmetic and the closing hashes do not need to match the number of hashes used to open the header. It is the number of opening hashes that determines the header level.

Example: Atx Style Headers

```
# H1 text #
## H2 text
### H3 text #####
#### H4 text
```

```
##### H5 text ###
##### H6 text
```

Horizontal Rules (for HTML output only)

You can produce a horizontal rule tag (`<hr>`) by placing three or more hyphens, asterisks, or underscores on a line by themselves (they also need to be preceded and followed by a blank line). Optionally, they can be separated by spaces.

Example: Horizontal Rules

```
* * *

*****

-----

- - - -
```

Paragraphs and Line Breaks

A paragraph is simply one or more consecutive lines of text, separated by one or more blank lines. The text at the beginning of a paragraph should not be indented with spaces or tabs. To create a new paragraph, simply insert a blank line in between them.



Important:

When converting to HTML, if you break a paragraph on multiple lines (without a blank line in between them), it will create a break tag (`<br>`). When converting to DITA, the text is kept in a single paragraph in this case and a blank line is required to break a paragraph. This behavior differs slightly from the default Markdown rules.

Example: Paragraphs

```
This is a paragraph that contains
two lines of text. (In HTML, a break tag is created in between the two lines)

This is a new paragraph.
```

Styling Text

The Markdown editor supports some syntax rules for styling text (such as bold, italic, or strikethrough).

• **Italic (Emphasis)**

Text wrapped with one asterisk or underscore produces an italic (emphasis) tag.

```
*italic*
_italic_
```

- **Bold (Strong)**

Text wrapped with two asterisks or underscores produces a bold (strong) tag.

```
**bold**
__bold__
```

- **Strikethrough**

In HTML only, text wrapped with two tildes (~~) produces a strikethrough tag.

```
~~strikethrough~~
```

- **Underline**

Text wrapped with two plus signs (++) produces an underline tag.

```
++underline++
```



Tip:

You can also combine these styling rules. For example, `**BoldText _ItalicText_ BoldText**` would produce italicized text within bold text. Also, if you surround an asterisk or underscore with spaces, it will be treated as a literal asterisk or underscore. To produce a literal asterisk or underscore at a position where it would otherwise be used as a styling delimiter, you can escape it with a backslash (for example, `\*literal asterisks\*`).

Links

The Markdown editor supports two types of links, *inline* and *reference*. In both cases, it begins with link text that is delimited by [square brackets].

- **Inline Links**

To create an inline link, use a set of regular parentheses immediately after the closing square bracket for the link text. Inside the parentheses, put the URL where you want the link to point, and optionally a title surrounded in quotes. Also, if you reference a local resource on the same server, you can use relative paths.

Examples: Inline Link

With a title:

```
Text with [example link text](http://www.example.com/path "Title") inline link and title.
```

Without a title:

```
Text with [example link text](http://www.example.com/path) inline link without a title.
```

Relative path:

```
Text with [example link text](/relative_path/) inline link with relative path.
```

• Reference Links

Reference-type links use a second set of square brackets that include a label (link identifier) to reference the target for the link (link identifier may consist of letters, numbers, spaces, and punctuation and it is not case-sensitive). You can optionally use a space to separate the sets of brackets. The labels (link identifiers) are only used for creating the links and do not appear in the output.

```
Text with [link text1][id 1] a reference-type link and [link text2][id_2] another one.
```

Then, somewhere in the document, you need to define your link label on a line by itself. The link identifier must be within square brackets followed by a colon, then after one or more spaces the URL for the link. Optionally this can be followed by a title enclosed in single quotes, double quotes, or parentheses. Also, the link may optionally be enclosed in angle brackets (< >).

```
[id 1]: http://example1.com/ "Optional Title"
[id_2]: <http://example2.com/> "Optional Title2"
```

Other notes about Reference Links:

- You can put the title on a second line and use extra spaces or tabs for padding. This is useful for aesthetics when the URL is long.

```
[id]: http://example.com/long/path/to/resource/here
      "Optional Title Here"
```

- The label (link identifier) can be missing, in which case the link text (in square brackets) is used as the name.

```
[My Link][]
```

and then defined as:

```
[My Link]: http://example.com/
```

Automatic Links

The Markdown editor supports a shortcut style for creating automatic links for URLs and email addresses. You simply surround the URL or email address with angle brackets.



Note:

These automatic links only work properly in HTML conversions. The *Preview* pane may display them properly in the DITA tab, but the DITA output will not properly recognize the format.

• URLs

By surrounding a URL with angle brackets, you can show the actual text of the URL while also making it clickable in the output.

```
<http://example.com/>
```

For example, in HTML it is converted to:

```
<a href="http://example.com/">http://example.com/</a>
```

• Email Addresses

Automatic links for email addresses work similarly, except that Markdown will also perform a bit of randomized decimal and hex entity-encoding to help obscure your address from address-harvesting *spambots*.

```
<address@example.com>
```

In HTML, it is converted to something like:

```
<a href="&#x6D;&#x61;i&#x6C;&#x74;&#x6F;:&#x61;&#x64;&#x64;&#x72;&#x65;&#x61;&#x64;&#101;&#120;&#x61;&#109;&#x70;&#x6C;&#x2E;&#99;&#111;&#109;">&#x61;&#x64;&#x64;&#x72;&#x65;&#115;&#115;&#64;&#101;&#120;&#x61;&#109;&#x70;&#x6C;&#x2E;&#99;&#111;&#109;</a>
```

Images

The Markdown editor uses an image syntax that is intended to resemble the syntax for two types of links (*inline* and *reference*). In both cases, the syntax for images begins with an exclamation mark, followed by `Alt` attribute text surrounded by square brackets, and then followed by a set of parentheses that contain the URL or path to the image.

• Inline Images

For inline images, use a set of regular parentheses immediately after the closing square bracket for the `Alt` attribute text. Inside the parentheses, put the URL or path of the image, and optionally a title surrounded in quotes.

Examples: Inline Images

With a title:

```
Text with ![Alt text](/path/to/img.jpg "Optional title") inline image and a title.
```

Without a title:

```
Text with ![Alt text](/path/to/img.jpg) inline link without a title.
```

• Reference Images

For reference-type images, use a second set of square brackets that include a label (image identifier) to identify the image (it may consist of letters, numbers, spaces, and punctuation and it is not case-sensitive). You can optionally use a space to separate the sets of brackets. The labels (image identifiers) do not appear in the output.

```
Text with ![Alt text1][id] a reference-type image.
```

Then, somewhere in the document, you need to define your image label on a line by itself. The image identifier must be within square brackets followed by a colon, then after one or more spaces the URL or path of the image. Optionally this can be followed by a title enclosed in single quotes, double quotes, or parentheses.

```
[id]: url/to/image "Optional Title"
```

Blockquotes

The Markdown editor uses email-style greater than characters (>) for *blockquotes*. You only need to put the > before the first line of a hard-wrapped paragraph, but it looks better (and is clearer) if you put a > before every line.

- **Example: Blockquotes**

```
> This is a blockquote with two paragraphs. Lorem ipsum dolor sit amet,
> consectetur adipiscing elit. Aliquam hendrerit mi posuere lectus.
> Vestibulum enim wisi, viverra nec, fringilla in, laoreet vitae, risus.
>
> Donec sit amet nisl. Aliquam semper ipsum sit amet velit. Suspendisse
> id sem consectetur libero luctus adipiscing.
```

- **Example: Nested Blockquotes**

Blockquotes can be nested by adding additional levels of > characters.

```
> This is the first level of quoting.
>
> > This is nested blockquote.
>
> Back to the first level.
```

- **Example: Blockquotes with Other Markdown Elements**

Blockquotes can also contain other Markdown elements (such as headers, lists, and code blocks).

```
> ## This is a header.
>
> 1. This is the first list item.
> 2. This is the second list item.
>
```

```
> Here's some example code:
>
>     return shell_exec("echo $input | $markdown_script")
```

Quoting Code (Inline and Code Blocks)

The Markdown editor supports quoting code or commands inline within a sentence or in distinct blocks.

• Inline

You can quote or emphasize code within a sentence (inline) with single backticks (`). The text within the backticks will not be formatted.

Example: Inline Code Emphasis

```
This is a normal sentence with a `code` in the middle.
```

• Code Blocks

You can format code or text into its own distinct block by inserting a blank line before and after the content and using at least 4 spaces (or 1 tab), or by using opening and closing triple backticks (```) on separate lines.

Example: Code Block

```
This is a normal paragraph:

    This is a code block

This is a normal paragraph:

```
This is a code block
```
```

One level of indentation is removed from each line of a codeblock and it continues until it reaches a line that is not indented (or until the closing backticks).

Example: Code Block with Indentation

```
    tell application "something"
        beep
    end tell
```

For example, in HTML the result would look like this:

```
<pre><code>tell application "Foo"
    beep
```

```
end tell
</code></pre>
```

You can also add an optional language identifier to enable syntax highlighting in your code blocks. The Oxygen JSON Editor Markdown editor syntax highlight supports the following languages: *Java*, *JavaScript*, *CSS*, and *Python*.

Example: Syntax Highlighting in Code Block

```
```css
input[type="submit"] {
 color: white;
 font-weight: bold;
}
```
```

Inline XHTML (for HTML output only)

The Markdown editor supports writing inline XHTML. Since Markdown is just a writing format, it requires a conversion for publishing purposes. If you are using the HTML conversion, for any markup that is not covered by Markdown syntax, you can simply use XHTML syntax.

Example: Inline XHTML

```
This is a regular paragraph.

<table>
  <tr>
    <td>Col 1</td>
    <td>Col 2</td>
  </tr>
</table>

This is another regular paragraph.
```

Lists

The Markdown editor supports ordered and unordered lists. You can also insert [blockquotes \(on page 462\)](#) and [code blocks \(on page 463\)](#) inside list items. List markers typically start at the left margin, but may be indented by up to three spaces.

• Unordered Lists

For unordered lists, you can use asterisks (*), plus signs (+), and hyphens (-) interchangeably.

```
* List item 1
+ List item 2
- List item 3
```

• **Ordered Lists**

For ordered lists, use numbers followed by periods. The actual numbers you use have no effect on the output. It simply converts them to list items within an ordered list and the actual number of list items will determine the numbers in the output.

```
1. List item 1
8. List item 2
5. List item 3
```

• **Nested Lists**

You can create nested lists by indenting lines by three spaces.

```
1. Ordered list item 1
  1. Nested ordered list item 1
  2. Nested ordered list item 2
    * 2nd level nested unordered list item 1
    * 2nd level nested unordered list item 2
      * 3rd level nested unordered list item 1
  2. Ordered list item 2
```

• **Paragraphs Inside Lists**

If list items are separated by blank lines, Markdown will wrap the items in a paragraph in the output.

```
* List item 1

* List item 2
```

For both DITA and HTML output, this would result in:

```
<ul>
<li><p>List item 1</p></li>
<li><p>List item 2</p></li>
</ul>
```

• **Multiple Paragraphs Inside Lists**

List items may consist of multiple paragraphs. Each subsequent paragraph in a list item must be indented by either 4 spaces or one tab. Optionally, you can also indent each line of a paragraph to make it look nicer.

```
1. This is a list item with two paragraphs. Lorem ipsum dolor
   sit amet, consectetur adipiscing elit. Aliquam hendrerit
   mi posuere lectus.

   Vestibulum enim wisi, viverra nec, fringilla in, laoreet
   vitae, risus. Donec sit amet nisl. Aliquam semper ipsum
```

```
sit amet velit.
```

```
2. Suspendisse id sem consectetur libero luctus adipiscing.
```

• **Blockquotes Inside Lists**

To put a *blockquote* within a list item, the blockquote delimiters (>) need to be indented so that they are under the first letter of the text after the list item marker.

```
* A list item with a blockquote:
  > This is a blockquote
  > inside a list item.
```

• **Code Blocks Inside Lists**

To put a code block within a list item, insert an empty line in between the list item and the code block, and the code block needs to be indented twice (with 8 spaces or 2 tabs), or if you are using the triple backticks method, the opening triple backtick needs to be indented with 4 spaces or 1 tab.

```
* A list item with a code block:

    This is a code block inside a list item

    ...

    This is a code block inside a list item using the backticks method
    ...
```

Task Lists

You can create task lists by prefacing list items with a hyphen followed by a space followed by square brackets (- []). To mark a task as complete, use - [x].

Example: Task Lists

```
- [ ] Unfinished task 1
- [x] Finished task 2
```

Definition Lists

You can create definition lists by using a colon plus a space for each list item.

Example: Definition Lists

```
Term 1
: Definition A
: Definition B
```

Tables

You can create tables in the Markdown editor by using pipes (|) and hyphens (-).

• Creating a Table

Pipes are used to separate each column, while hyphens are used to create column headers. The pipes on either end of the table are optional. Cells can vary in width and do not need to be perfectly aligned within columns, but there must be at least three hyphens in each column of the header row.

```
| First Header | Second Header |
| ----- | ----- |
| Column 1 Row 1 Cell | Column 2 Row 1 Cell |
| Column 1 Row 2 Cell | Column 2 Row 2 Cell |
```

• Formatting Rules in Table Cells

You can use formatting rules inside the cells of the table (such as links, inline code blocks, and text styling).

```
| First Header | Second Header |
| --- | --- |
| `inline code` | Content with bold text inside cell |
```

• Aligning Text in Tables

You can align text to the left, right, or center of a column by including colons (:) to the left, right, or on both sides of the hyphens within the header row.

```
| Left-aligned | Center-aligned | Right-aligned |
| :--- | :---: | ---: |
| Content Cell | Content Cell | Content Cell |
```

• Joining Cells (Span a Cell Over Multiple Columns)

You can join cells horizontally (span a cell over multiple columns) by using multiple consecutive pipe characters (|) to the right of the particular cell. The number of consecutive pipes indicate the number of columns the cell will span (|| for two, ||| for three, and so on).

```
| First Header | Second Header | Third Header | Fourth Header |
| ----- | ----- | ----- | ----- |
| Content Cell | *Cell Span Over 3 Columns* |||
```

Emoji

You can add *emoji* in the Markdown editor by surrounding the EMOJICODE with colons (:EMOJICODE:).

Example: Emoji

```
:smile:
:laughing:
```

The resulting emoticons will appear in the output, but they are not displayed in the *Preview* pane.

For a full list of available emoji codes, see [Emoji Cheat Sheet](#).

Backslash Escapes

You can ignore Markdown formatting by using backslash escapes (\\) to generate literal characters that would otherwise have special meaning in the Markdown syntax. For example, if you want to surround a word with literal asterisks (instead of an italic or emphasis tag), you can use backslashes to escape the asterisks.

```
\\*literal asterisks\\*
```

The Markdown editor provides backslash escapes for the following characters:

```
\\  backslash
`  backtick
*  asterisk
_  underscore
{}  curly braces
[]  square brackets
()  parentheses
#  hash mark
+  plus sign
-  minus sign (hyphen)
.  dot
!  exclamation mark
```

Automatic Escaping for Special Characters

The Markdown editor includes support for automatically escaping special characters such as angle brackets (< >) and ampersands (&). If you want to use them as literal characters, you must escape them as entities, as in the table below. The exception to this is in HTML output, if the special characters form a valid tag (for example,), they are treated as literal characters and escaping is not necessary.

Literal Character	Escaping Code
<	<
>	>
&	&

Footnotes

The Markdown editor in Oxygen JSON Editor supports normal and inline footnotes. The following examples show the required syntax.

- **Example: Normal Footnote**

```
Here is a footnote reference,[^1]
```

```
[^1]: Here is the footnote.
```

- **Example: Normal Footnote with Multiple Blocks**

```
Here is a footnote reference,[^longnote]
```

```
[^longnote]: Here is the footnote with multiple blocks.
```

```
    Subsequent paragraphs are indented with 4 spaces or 1 tab to show that they  
    belong to the previous footnote.
```

- **Example: Inline Footnote**

```
Here is an inline note.^[Inlines notes are easier to write, since  
you don't have to pick an identifier and move down to type the  
note.]
```

Related information

[Default Markdown Syntax](#)

[GitHub Flavored Markdown Rules](#)

[Markdown Editor \(on page 449\)](#)

[Actions Available in the Markdown Editor \(on page 450\)](#)

7.

Built-in Frameworks (Document Types)

Oxygen JSON Editor includes a variety of specialized editors, views, and features that are dynamic according to the type of document that you open or create. Oxygen JSON Editor includes fully supported built-in [frameworks \(on page 639\)](#) for popular document types (e.g. JSON, YAML, OpenAPI, Markdown, HTML, CSS), each with a specialized set of [editing features for the particular document type \(on page 352\)](#).

The built-in [frameworks \(on page 639\)](#) are defined according to a set of rules and a variety of settings that improve editing capabilities for its particular file type. These settings include:

- A default grammar used for validation and content completion in **Text** mode.
- Built-in transformation scenarios used for publishing XML documents.
- [XML Catalogs \(on page 640\)](#) used for mapping resources.
- New document templates to make it easy to create XML documents.

OpenAPI (Swagger) Document Type (Framework)

OpenAPI specification, previously known as Swagger specification, is a specification that defines a standard, programming language-agnostic interface description for HTTP APIs, which allows both humans and computers to discover and understand the capabilities of a service without requiring access to source code, additional documentation, or inspection of network traffic. Use cases for machine-readable API definition documents include interactive documentation, code generation for documentation, automation of test cases, and more. OpenAPI documents describe an API's services and are represented in either YAML or JSON format.

Oxygen JSON Editor includes three OpenAPI frameworks:

- OpenAPI 2.0
- OpenAPI 3.0
- OpenAPI 3.1

Editing OpenAPI Documents

You can edit OpenAPI files in **Text** mode and you have access to all the usual text editing actions.



Tip:

There is an OpenAPI sample document named `petsore.json` located in `[OXYGEN-INSTALL-DIR]/samples/json/openapi` that you can use to see how these documents are rendered in Oxygen JSON Editor.

Validation and Content Completion

Validation and content completion is supported in Oxygen JSON Editor for OpenAPI documents (version 2.0, 3.0, 3.1). The validation and content completion in OpenAPI documents are driven by schemas according to the OpenAPI version in the document. Each of the three frameworks (OpenAPI 2.0, OpenAPI 3.0, and OpenAPI 3.1) have a unique schema specified for content completion and validation. When opening an OpenAPI document (in JSON or YAML format), Oxygen JSON Editor automatically associates the corresponding schema based on the OpenAPI version of the document.

Resources

- Oxygen JSON Editor includes a tool for generating documentation for OpenAPI components in HTML format: [Generating OpenAPI Documentation \(on page 575\)](#).
- Oxygen JSON Editor includes a testing tool for OpenAPI files: [OpenAPI Tester \(on page 567\)](#).
- For more details about the *OpenAPI Specification*, along with example documents, go to <https://spec.openapis.org/oas/latest.html>.
- Video: [OpenAPI Document Editing in Oxygen XML Editor](#)
- Webinar: [OpenAPI Editing, Testing, and Documenting](#)
- Webinar: [OpenAPI/AsyncAPI Support in Oxygen](#)

OpenAPI Test Scenario Document Type (Framework)

Oxygen JSON Editor includes a specialized framework for working with OpenAPI test scenario files. It includes the usual text editing actions, a visual editing interface, content completion, and automatic validation.

Default Document Template

There is a default scenario file template available when creating [new documents from templates \(on page 225\)](#) and they can be found in the **Framework Templates > OpenAPI Test Scenario**.

Content Completion

Oxygen JSON Editor helps you edit OpenAPI test scenario files through the [Content Completion Assistant \(on page 638\)](#), offering proposals for properties and values that can be inserted at the cursor position. It can be manually activated with the **Ctrl + Space** shortcut.

Validation

Oxygen JSON Editor includes built-in validation for OpenAPI test scenario documents to help you keep them well-formed. The documents are validated automatically as you type against the schema specified in the framework and problems are highlighted within the document.

OpenAPI Tools

The following additional OpenAPI tools are available as free add-ons:

- **OpenAPI Tester** - Provides the ability to inspect OpenAPI request responses and to ensure that they work as expected. It can be used for [OpenAPI 3.x](#) in JSON or YAML format. For details, see [OpenAPI Tester Add-on \(on page 567\)](#).
- **Run OpenAPI Test Scenario** - Provides the ability to run a test suite for an OpenAPI document in JSON format. It performs the requests based on the specified OpenAPI document and the data entered in the test file, and then checks if the server responses are as expected. For details, see [Run OpenAPI Test Scenario \(on page 570\)](#).
- **OpenAPI Documentation Generator** - Provides the ability to generate documentation for *OpenAPI* components in HTML format, including annotations and cross references. The documentation displays information about the servers, paths, components and tags defined in the [OpenAPI 3.0](#) documents and it is presented in a visual diagram style with various sections, hyperlinks, and filtering options. For details, see [OpenAPI Documentation Generator Add-on \(on page 575\)](#).

Resources

- For more details about the *OpenAPI Specification*, along with example documents, go to <https://spec.openapis.org/oas/latest.html>.
- Video: [OpenAPI Document Editing in Oxygen XML Editor](#)
- Webinar: [OpenAPI Editing, Testing, and Documenting](#)
- Webinar: [OpenAPI/AsyncAPI Support in Oxygen](#)

AsyncAPI Document Type (Framework)

Oxygen JSON Editor includes a specialized framework for working with [AsyncAPI](#) files. The application supports the following AsyncAPI versions: `1.0.0`, `1.1.0`, `1.2.0`, `2.0.0`, `2.1.0`, `2.2.0`, `2.3.0`, `2.4.0`.

Editing AsyncAPI Documents

You can edit AsyncAPI files in **Text** mode and you have access to all the usual text editing actions.

Default Document Templates

There are some default AsyncAPI templates available when creating [new documents from templates \(on page 225\)](#) and they can be found in the **Framework Templates > AsyncAPI 1.x** and **Framework Templates > AsyncAPI 2.x** folders. Each of those folders contain a default new document template for a JSON version and a YAML version. Some other useful examples can be found at [public AsyncAPI GitHub project](#).

Content Completion

Oxygen JSON Editor helps you edit AsyncAPI files through the [Content Completion Assistant \(on page 638\)](#), offering proposals for properties and values that can be inserted at the cursor position. It can be manually activated with the **Ctrl + Space** shortcut.

Validation

Oxygen JSON Editor includes built-in validation for AsyncAPI documents to help you keep them well-formed. The documents are validated automatically as you type against the schema specified in the framework and problems are highlighted within the document.

Resources

- For details about the *AsyncAPI Specification*, go to <https://www.asyncapi.com/docs/reference/specification/v2.0.0>.
- Webinar: [OpenAPI/AsyncAPI Support in Oxygen](#)

JSON-LD Document Type (Framework)

Oxygen JSON Editor includes a specialized framework for working with **JSON-LD** files. *JSON-LD* (JavaScript Object Notation for Linked Data) consists of multi-dimensional arrays and is considered an easy-to-use lightweight *Linked Data* format.

Editing JSON-LD Documents

You can edit JSON-LD files in the [specialized JSON text mode editor \(on page 359\)](#) and you have access to its various features and actions.

Default Document Template

There are some default *JSON-LD* templates available when creating [new documents from templates \(on page 225\)](#) and they can be found in: **Framework Templates > JSON-LD**. Some other useful examples can be found at [public JSON-LD GitHub project](#).

Content Completion

Oxygen JSON Editor includes an intelligent [Content Completion Assistant \(on page 638\)](#) that offers proposals for inserting JSON structures that are valid at the current editing location. For more details, see [Content Completion Assistant in JSON \(on page 371\)](#).

Validation

Oxygen JSON Editor includes built-in validation for JSON-LD documents to help you keep them well-formed. The documents are validated automatically as you type against the schema specified in the framework and problems are highlighted within the document. For more details, see [Validating JSON Documents \(on page 363\)](#).

8.

Working with XPath Expressions

XPath is a language for addressing specific parts of a document. XPath models an XML document as a tree of nodes. An XPath expression is a mechanism for navigating through and selecting nodes from the document. An XPath expression is, in a way, analogous to an SQL query used to select records from a database.



Note:

If an XPath expression is run over a JSON document, it is converted to XML and the XPath is executed over the converted XML document.

There are various types of nodes, including element nodes, attribute nodes, and text nodes. XPath defines a way to compute a string-value for each type of node.

XPath defines a library of standard functions for working with strings, numbers and boolean expressions.

Examples:

- **child::*** - Selects all children of the root node.
- **./name** - Selects all `<name>` elements and 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.
- **//prolog** - Finds all `<prolog>` elements.
- **//prolog[@platform='mac']** - Finds all `<prolog>` elements that have the `@platform` attribute value set to `mac`.
- **//child::prolog** - Selects all `@prolog` elements and the child content.
- **/*[count(//accountNumber) > 5]** - Searches for instances where more than 5 `<accountNumber>` elements are found.
- **collection('file:/C:/path/to/folder/?select=*.xml')/*[not(//prolog)]** - Finds a list of all XML files that do not contain any `<prolog>` elements.

To find out more about XPath, see <http://www.w3.org/TR/xpath>.

XPath Builder View

The **XPath/XQuery Builder** view allows you to compose complex XPath expressions and execute them over the currently edited XML document. For XPath 2.0 / 3.1, you can use the `doc()` function to specify the source file that will have the expressions 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 JSON Editor executes the expressions over the XML document in the associated scenario.

**Note:**

If an XPath expression is run over a JSON document, it is converted to XML and the XPath is executed over the converted XML document.

If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

The upper part of the view contains the following actions:

XPath version chooser drop-down menu

A drop-down menu that allows you to select the type of the expression you want to execute. You can choose between:

- XPath 1.0 (Xerces-driven)
- XPath 2.0, XPath 2.0 SA, XPath 3.1, XPath 3.1 SA, 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 3.1 SA have a location limited to the line number of the start element (there are no column information and no end specified).

**Note:**

Oxygen JSON Editor uses Saxon to execute XPath 3.1 expressions. Since Saxon implements a part of the 3.1 functions, when using a function that is not implemented, Oxygen JSON Editor returns a compilation error.

**Execute XPath button**

Use this button to start the execution of the XPath or XQuery expression you are editing. The result of the execution is displayed in the **Results** view.

**Favorites button**

Allows you to save certain expressions that you can later reuse. To add an expression as a favorite, click this 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 JSON Editor automatically groups favorites in folders named after the method of execution.

**History drop-down menu**

Keeps a list of the last 15 executed XPath expressions. Use the  **Clear history** action from the bottom of the list to remove them.

**Settings drop-down menu**

Contains the following three options:

 Update on cursor move

When selected and you navigate through a document, the XPath expression corresponding to the XML node at the current cursor position is displayed. For JSON documents, it displays the XPath expression for the current property.

 Evaluate as you type

When you select this option, the XPath expression you are composing is evaluated in real time.



Note:

This option and the automatic validation are disabled when you edit [huge documents \(on page 308\)](#) or when the scope is other than **Current file**.

 Options

Opens the Preferences page of the currently selected processing engine.

XPath scope menu

Oxygen JSON Editor allows you to define a scope for the XPath operation to be executed. You can choose where the XPath expression will be executed:

-  **Current file** - Currently selected file only.
-  **Project** - All the files in the project.
-  **Selected project resources** - The files selected in the project.
-  **All opened files** - All files that are opened in the application.
-  **Opened archive** - Files that are opened in the [Archive Browser view \(on page 480\)](#).
-  **Working sets** - The selected [working sets \(on page 640\)](#).

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 \(on page 640\)](#).
-  **XPath file filter** - You can filter the files from the selected scope that will have the XPath expression executed. By default, the XPath expression will be executed only on XML or JSON 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.

While you edit an XPath or XQuery expression, Oxygen JSON Editor assists you with the following features:

- *Content Completion Assistant (on page 638)* - It offers context-dependent proposals and takes into account the cursor position in the document you are editing. The set of functions proposed by the *Content Completion Assistant* also depends on the engine version. Select the engine version from the drop-down menu available in the toolbar.
- Syntax Highlighting - Allows you to identify the components of an expression. To customize the colors of the components of the expression, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to **Editor > Syntax Highlight** (on page 151).
- 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.

Related Information:

[XPath Expression Results View \(on page 477\)](#)

XPath Expression Results View

When you run an XPath expression, Oxygen JSON Editor displays the results of its execution in the **Results** view.

This view contains the following columns:

- **Description** - The result that Oxygen JSON Editor displays when you run an XPath expression.
- **XPath location** - The path to the matched node.
- **Resource** - The name of the document that you run the XPath expression on.
- **System ID** - The path to the document itself.
- **Location** - The location of the result in the document.

To arrange the results depending on a column, click its header. To group the results by their resource, or by their system ID, right-click the header of any column in the **Results** view and select **Group by "Resource"** or **Group by "System ID"**. If no information regarding location is available, Oxygen JSON Editor displays **Not available** in the **Location** column. Oxygen JSON Editor displays the results in a valid XPath expression format.

```
- /node[value]/node[value]/node[value] -
```

Example:

The following snippets are taken from a DocBook book based on the DocBook XML DTD. The book contains a number of chapters. To return all the chapter nodes of the book, enter `//chapter` in the XPath expression field and press **Enter**. This action returns all the `chapter` nodes of the DocBook book in the **Results View**. Click a record in the **Results View** to locate and highlight its corresponding chapter element and all its children nodes in the document you are editing.

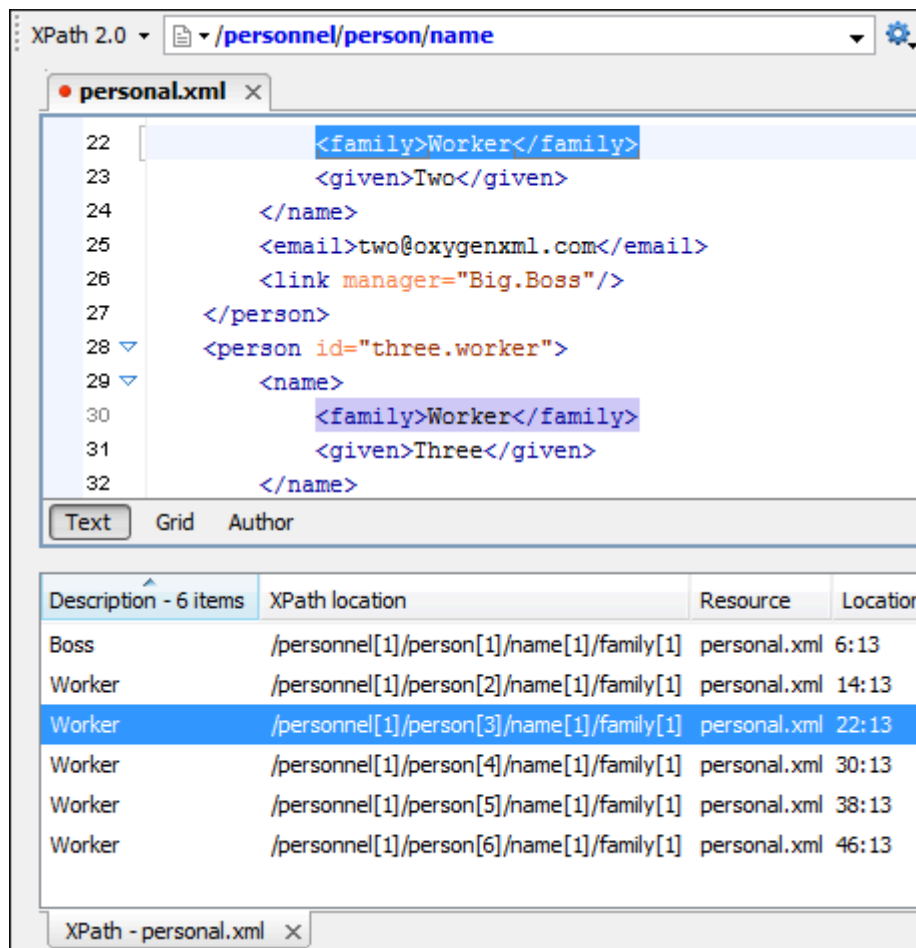
To find all `example` nodes contained in the `sect2` nodes of a DocBook XML document, use the following XPath expression: `//chapter/sect1/sect2/example`. Oxygen JSON Editor adds a result in the **Results View** for each `example` node found in any `sect2` node.

For example, if the result of the above XPath expression is:

```
- /chapter[1]/sect1[3]/sect2[7]/example[1]
```

it means that in the edited file, the `example` node is located in the first chapter, third section level one, seventh section level 2.

Figure 132. XPath Results Highlighted in Editor Panel with Character Precision



XPath and XML Catalogs

The evaluation of the XPath expression tries to resolve the locations of documents referenced in the expression through *XML Catalogs* (on page 640).

Example:

As an example, consider the evaluation of the `collection(URIofCollection)` function (XPath 2.0). To resolve the references from the files returned by the `collection()` function with an *XML catalog*, specify the class name of the catalog-enabled parser for parsing these collection files. The class name is `ro.sync.xml.parser.CatalogEnabledXMLReader`. Specify it as it follows:

```
let $docs := collection(iri-to-uri(  
    "file:///D:/temp/test/XQuery-catalog/mydocsdir?recurse=yes;select=*.xml;  
    parser=ro.sync.xml.parser.CatalogEnabledXMLReader"))
```

9.

Working with Archives

Oxygen JSON Editor includes a useful **Archive Browser** view (*on page 480*) that offers the means to work with files directly from various types of archives (for example, opening and saving files directly in archives, or browsing and modifying archive structures). The archive support is available for all ZIP-type archives, including:

- ZIP archives
- EPUB books
- *JAR archives (on page 639)*
- Office Open XML (OOXML) files
- Open Document Format (ODF) files
- *IDML files (on page 639)*

You can transform, validate, and perform many other operations on files directly from an archive. For instance, you can transform, or validate files directly from OOXML or ODF packages, and the structure and content of the ZIP archives can be opened, edited, and saved, similar to any other ZIP archive browsing tool. Also, when browsing for a URL in various dialog boxes, you can use the  **Browse for archived file** action to browse and select files from a particular archive.

Resources

For more information about working with an EPUB archive in Oxygen JSON Editor, watch our video demonstration:

<https://www.youtube.com/embed/OIGTNQwOCi8>

Browsing Archives

Oxygen JSON Editor includes a helper view called the **Archive Browser** that allows you to view the contents and structure of an archive, and it offers a variety of toolbar and contextual menu actions. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

To open an archive in the **Archive Browser** view, use one of the following methods:

- Open an archive from the **Project view (on page 251)**.
- Select an archive in one of the file chooser dialog boxes in Oxygen JSON Editor (such as the **Open** dialog box).
- Drag an archive from a system file explorer and drop it in the **Archives Browser** view.

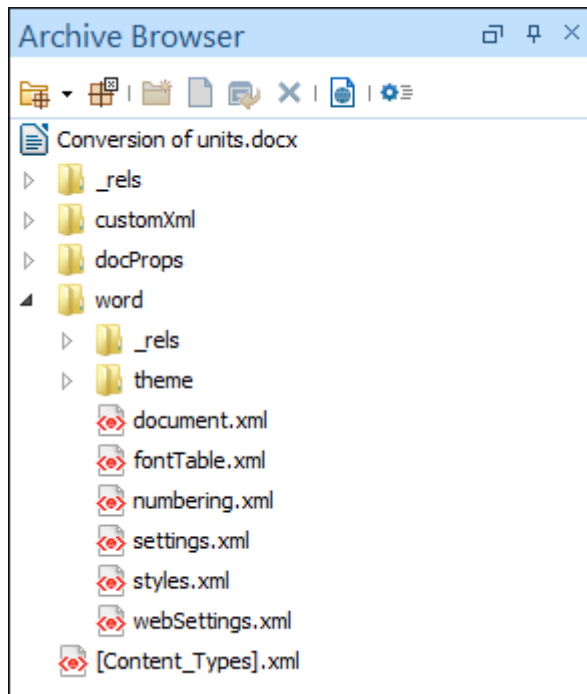
When displaying an archive, the **Archive Browser** view locks the archive file. It is then automatically unlocked when the **Archive Browser** view is closed.



Tip:

If a file is not recognized by Oxygen JSON Editor as a supported archive type, you can add it in the [Archive preferences page \(on page 170\)](#).

Figure 133. Archive Browser



Archive Browser Toolbar Actions

The following actions are available on the **Archive Browser** toolbar:



Open Archive menu

Provides access to the **Open Archive** action that opens a new archive in the browser. If the extension is not known as an archive extension, you will be directed to the [Archive preferences page \(on page 170\)](#) to add a new extension. The submenu keeps a list of recently open archive files and a **Clear history** action that allows you to delete the list.



Close

Closes the browsed archive and unlocks the archive file.



Validate (available for EPUB archives only)

Checks the EPUB archive to see if its content and structure is valid.



New folder

Creates a folder as child of the selected folder in the browsed archive.

New file

Creates a file as child of the selected folder in the browsed archive.

Add files

Adds existing files as children of the selected folder in the browsed archive.



Note:

You can also add files in the archive by dragging them from the file browser or the [Project view \(on page 251\)](#) and dropping them in the **Archive Browser** view.

Delete

Deletes the selected resource in the browsed archive.

Open in System Application

Opens the selected resource in the default system application that is associated with that type of file.

Archive Options

Opens the [Archive preferences page \(on page 170\)](#).

Archive Browser Contextual Menu Actions

The following additional actions are available from the contextual menu for resources in the **Archive Browser** view:

Open

Opens a resource from the archive in the editor.

Extract

Extracts a resource from the archive in a specified folder.

New folder

Creates a folder as child of the selected folder in the browsed archive.

New file

Creates a file as child of the selected folder in the browsed archive.

Add files

Adds existing files as children of the selected folder in the browsed archive.



Note:

On macOS, the **Add file** action is also available and it allows you to add one file at a time.

Rename

Renames a resource in the archive.

**Find/Replace in Files**

Opens the **Find/Replace in Files** dialog box (*on page 277*) that allows you to search for and replace specific pieces of text inside the archive.

**Cut**

Cuts the selected archive resource.

**Copy**

Copies the selected archive resource.

**Paste**

Pastes a file or folder into the archive.

**Delete**

Removes a file or folder from archive.

Copy location

Copies the URL location of the selected resource.

**Refresh**

Refreshes the selected resource.

Properties

Shows the properties of the selected resource.

Resources

For more information, watch our video demonstration about working with an EPUB in the **Archive Browser** view:

<https://www.youtube.com/embed/OIGTNQwOCi8>

Working with Archive Files

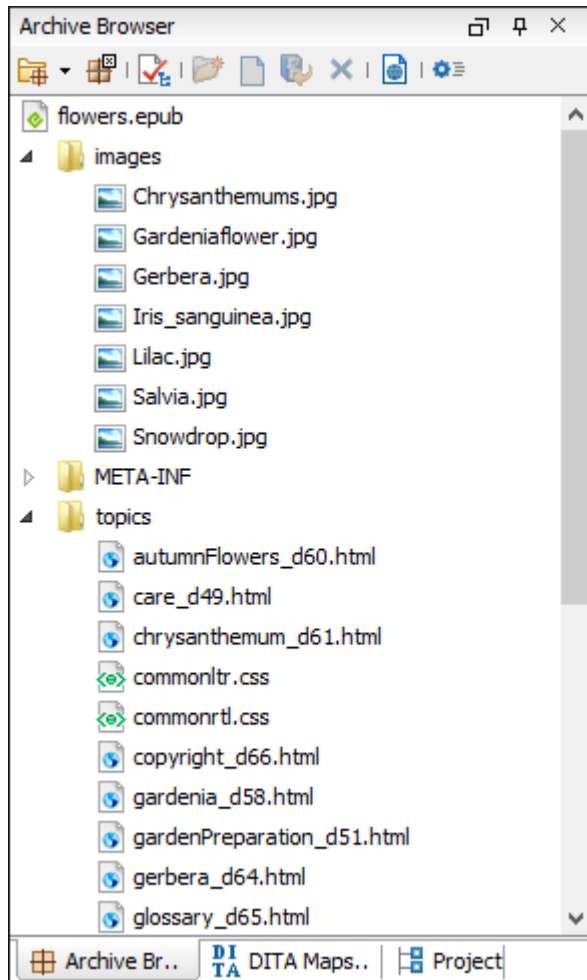
Oxygen JSON Editor includes support for working with various types of archives, including the following:

- **EPUB** - An e-book file format that can be used on many types of devices, such as smart phones, tablets, e-readers, or computers.
- **OOXML** - An XML-based file format for representing spreadsheets, charts, presentations, and word processing documents.
- **ODF** - An free and open-source XML-based file format for electronic office documents, such as spreadsheets, charts, presentations, and word processing documents.

When these types of files are opened in the **Archive Browser view** (on page 480), their internal components are expanded:

- Document content (XHTML and image files).
- Packaging files.
- Container files.

Figure 134. EPUB File Displayed in the Archive Browser View



When an archive is expanded in the **Archive Browser view** (on page 480), you can add or delete files that compose the archive structure. All changes made to the structure of an archive are saved immediately. You can open files from within the archive to edit them in the main editing pane and save changes back to the archive. You can also use the **Open in System Application** action to open the archive in the default system application that is associated with that type of file.

EPUB-Specific Validation

When working with EPUB archives, the **Archive Browser** (on page 480) includes a **Validate** action on the toolbar that checks the EPUB archive to make sure the structure and content are valid. Oxygen JSON Editor uses the open-source *EpubCheck* validator to perform the validation. This validator detects many types of errors, including OCF container structure, OPF and OPS mark-up, as well as internal reference consistency.

Resources

For more information about working with an EPUB archive in Oxygen JSON Editor, watch our video demonstration:

<https://www.youtube.com/embed/OIGTNQwOCi8>

Creating an Archive

To create an archive from scratch, follow these steps:

1. Go to **File > New** or click  **New** on the main toolbar.
2. Choose your particular type of archive template. For example, select one of the **ODF**, **OOXML**, or **EPUB** templates.
3. Click **Create** and choose the name and location of the file.
4. Click **Save**.

A skeleton archive is saved on disk and open in the **Archive Browser** view (*on page 480*).



Tip:

Use toolbar and contextual menu actions to edit, add, and remove resources from the archive. For EPUB archives, you can use the  **Validate** action to verify the integrity of the EPUB archive.

Migrating Archives to DITA or TEI

Certain types of archives can be converted to DITA or TEI. For example, OOXML (Office Open XML) archive files with the DOCX file extension can be migrated to DITA or TEI.

To migrate DOCX files to DITA or TEI, follow these steps:

1. Open and expand the archive in the **Archive Browser** (*on page 480*).
2. Open the **document.xml** file contained in the archive.
3. Run one of the following built-in transformation scenarios:
 - a. **DOCX DITA** to migrate to DITA.
 - b. **DOCX TEI P5** to migrate to TEI.
4. You may need to do some manual reconfiguring to map DOCX styles to DITA or TEI content.



Tip:

Oxygen JSON Editor also includes a built-in transformation scenario called **ODT TEI P5** for converting ODF archive files with the ODT file extension to TEI and a similar process can be used to migrate ODT files to TEI.

10.

Add-ons

Oxygen JSON Editor offers various default *add-ons* (on page 640) that can be installed to provide additional functionality to Oxygen JSON Editor. Some additional community submissions are also available, although community add-ons are not officially supported or endorsed. For a full list of *add-ons* that are officially supported for Oxygen JSON Editor, see [Oxygen XML Add-on Repositories](#).

This chapter contains information about the default add-ons that are available to install directly from Oxygen JSON Editor.

To install one of the default add-ons, follow this procedure:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box.
2. Enter or paste <https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml> in the **Show add-ons from** field or select it from the drop-down menu.
3. Select the add-on you want to install and click **Next**.
4. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
5. Restart the application.

Resources

For more information about extending Oxygen JSON Editor through add-ons, see the following webinars/presentations:

- [Webinar: Extending the Functionality of Oxygen Using Add-ons](#)
- [Webinar: Add-ons You Can Use for Technical Writing](#)
- [XML Prague Conference Presentation: All About Oxygen Add-ons](#)

Collaboration

Git Client Add-on

An add-on is available that contributes a built-in Git client directly in Oxygen JSON Editor. Once the add-on is installed, a **Git Staging** view is available that includes various actions that perform common Git commands, such as *push*, *pull*, *change branch*, *commit*, and more. It also includes a built-in tool for comparing and merging changes.

Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** to quickly initiate the installation process:

Install

Manual Installation

To manually install the add-on, follow this procedure:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box. Enter or paste **https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml** in the **Show add-ons from** field or select it from the drop-down menu.



Note:

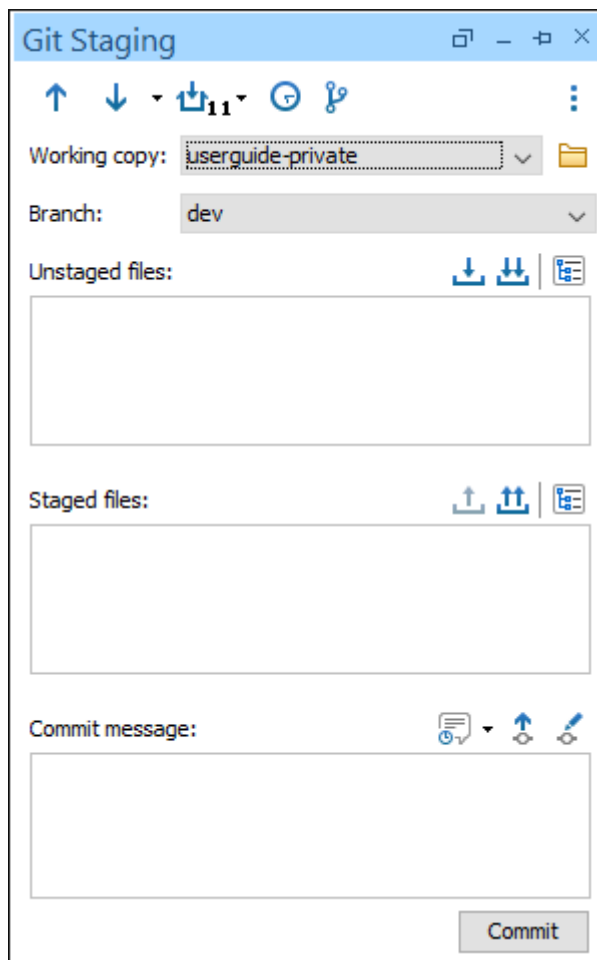
If you have issues connecting to the default update site, you can [download the add-on package](#), unzip it, then use the **Browse for local files** action in the **Install new add-ons** dialog box to locate the downloaded **addon.xml** file.

2. Select the **Git Client** add-on and click **Next**.
3. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
4. Restart the application.

Result: The **Git Staging** view is now available. To open the view, select **Git Staging** from the **Window > Show View** menu (or select **Git Client** from the **Tools** menu). This view acts as a Git client integrated directly in Oxygen JSON Editor, and it provides support for committing changes to a Git repository, comparing and merging changes, resolving conflicts, and other Git commands. A **Git** menu is also contributed in the main menu bar that includes various Git actions.

Git Staging View Interface

Once the **Git Client** add-on is installed, the **Git Staging** view is available by selecting it from the **Window > Show View** menu (or select **Git Client** from the **Tools** menu). The **Git Staging** view is the main interface where most of the actions that trigger Git commands and other features can be accessed.

Figure 135. Git Staging View

The **Git Staging** view includes the following actions/features (most are also available in the **Git** menu):

Push

Pushes your local repository changes to the remote repository. The arrow icon has a small plus sign in the bottom-right corner when there are changes that have not yet been pushed to the repository.

Pull

Pulls the changes from the remote repository into your local repository.

Stash drop-down menu

Includes the following submenu items:

Stash changes

Creates a new stash from the working copy changed file.

List stashes

Opens a dialog box that lists the existing stashes and you can choose to apply or remove a stash.

Show current repository history

Opens the **Git History view** ([on page 493](#)) at the bottom of the application.

Show Git Branch Manager view

Opens the **Git Branch Manager view** ([on page 497](#)) on the right side of the application.

Ellipsis (three vertical dots) menu

Includes the following submenu items:

Clone new repository

Clones a repository into a new directory.

Push

Pushes your local repository changes to the remote repository.

Pull (merge)

Pulls the latest changes from the remote repository and combines them with your local unpublished changes by creating one new merge commit.

Pull (rebase)

Pulls the latest changes from the remote repository, rewrites their commits, and reapplies them on top of your local unpublished changes.

Show branches

Opens the **Git Branch Manager view** ([on page 497](#)) on the right side of the application.

Show tags

Opens a dialog box that lists all the tags. You can delete tags, checkout a certain tag, or push a tag that is only present in the local branch to the remote.

Show history

Opens the **Git History view** ([on page 493](#)) at the bottom of the application.

Submodule

Opens a combo box where you can select the desired submodule to open and work with.

Stash changes

Creates a new stash from the working copy changed file.

List changes

Opens a dialog box that lists the existing stashes and you can choose to apply or remove a stash.

Manage remote repositories

Opens a dialog box where you can add, edit, or delete existing remote repositories.

Track remote branch

Opens a dialog box where you can select a remote branch that the local branch will track for executing fetch, push, or pull commands.

Edit repository "config" file

Opens the configuration file for the repository in the main editor so that it can be edited.

Preferences

Opens the **Git Client** preferences page where you can configure some options that relate to the add-on.

Reset all credentials

Resets all credentials to the default values.

Unstaged files area

Any changed, unstaged files are listed in this box. The following actions are available at the top-right corner of the box:

**Stage selected**

Moves the selected unstaged, changed files to the **Staged files** area.

**Stage all**

Moves all unstaged, changed files to the **Staged files** area.

**Switch to tree view**

Switches to a tree-like view.

Staged files area

All staged files are listed in this box. The following actions are available at the top-right corner of the box:

**Unstage selected**

Moves the selected staged files to the **Unstaged files** area.

**Unstage all**

Moves all staged files to the **Unstaged files** area.

**Switch to tree view**

Switches to a tree-like view.

Commit message area

This box in this area is used to write a commit message. The following actions are available at the top-right corner of the box:

**Choose a previously used comment**

Allows you to re-use a commit message that you used in the past.

**Automatically push changes to remote branch**

When a commit is performed, the committed changes are also pushed to the remote repository.

**Ammend last commit**

Allows you to edit the selected commit. You can combine staged changes with the previous commit instead of creating an entirely new commit. It can also be used to simply edit the previous commit message without changing its snapshot. This action should not be performed on public commits (commits that were pushed to the remote repository).

Cloning a Repository

Click the **Clone new repository** button (it has a '+' sign as the icon) and provide the following:

- **Repository URL** - The URL of the remote repository to be cloned.
- **Checkout branch** - A specific branch of the repository that is being cloned. The default branch will be cloned if another one is not specified.
- **Destination path** - The local path where the repository will be cloned. When you enter a URL of a repository, it will be proposed to automatically save it in a folder with the same name.

After cloning a repository, it will automatically be set as the current working copy.

Making an Oxygen Project a Git Repository

When showing the **Git Staging** side-view after opening an Oxygen JSON Editor project that is not already a Git repository, Oxygen JSON Editor offers the possibility to make that project also a local Git repository. This is very useful, for example, for newly created projects. After creating the local repository, bind it to a remote repository. You will be asked to specify the URL of the corresponding remote repository on your first attempt to push or pull changes.

Authentication

The *Git Client* supports **HTTPS** and **SSH** connections to GitHub, GitLab, Bitbucket, and more. It also includes support for `ssh-agent` forwarding.

**Note:**

For SSH connections, you can use the SSH agent to use the SSH keys already stored on it. To enable support for SSH agent access, go to **Git(Menu) > Settings > Preferences > SSH Connections** (or **Git Staging View > Ellipsis menu > Preferences > SSH Connections**) and select the **Use SSH Agent** option.

To access the remote repository, you need to provide your authentication details (if not using unprotected SSH keys). If no such information is found in the add-on's settings, you will be prompted for them.

The *Git Client* allows you to authenticate over HTTPS by using either a basic authentication method (username + password) or a personal access token.

**Notes:**

- The authentication using personal access tokens has been tested with GitHub and GitLab.
- Bitbucket uses a concept similar to personal access tokens, named *app passwords*. An app password must be provided as the password for *Basic authentication*, along with the correct username.
- As of August 13, 2021, GitHub will no longer accept password-based authentication.

If you have the **two-factor authentication (2FA)** enabled for GitHub or GitLab, to authenticate over HTTPS, you must generate a personal access token for your profile, and back in the *Git Staging* view in Oxygen JSON Editor, use the generated token value as the authentication password when asked for your credentials.

If, for example, you have been using a GitHub account but you decide to switch to another GitHub account, you would need to reset your credentials so that you will be prompted for new ones. This is because only one set of credentials for each Git platform/server is stored. To reset your credentials, go to the toolbar at the top of the **Git Staging** side-view, click the settings icon (a cogwheel), and select **Reset all credentials**.

Selecting a Working Copy

Click the **Browse** button to the right of the **Working copy** combo box to select a working copy from your file system. The selected folder must be a Git repository.

Switching Between Local Branches

To switch between local branches from the **Git Staging** view, use the **Branch** combo box. Local branches can also be changed using the **Git Branch Manager** ([on page 497](#)).

New branches can be created from the **Git Branch Manager** ([on page 497](#)) or from the **History** table using the **Create branch** action in the contextual menu.

Stashing Files

A Git stash is a way of temporarily saving the changes you have made to your working copy so that you can continue to use other Git operations in the meantime, and you can then re-apply the stashed changes later.

On the toolbar, there is a **Stash** drop down menu with a **Stash changes** action for creating a new stash out of the working copy changed file and a **List stashes** action that presents a dialog box with the existing stashes. From, the **List stashes** dialog box, you can also choose to apply or remove a stash.

Creating Tags

Tagging is used to capture a specific point in history, for example to mark a release. A tag is like a branch that doesn't change. You can create tags from the **History view** ([on page 493](#)) contextual menu. In the *Ellipsis* (*three vertical dots*) menu on the right side of the **Git Staging** view's toolbar, there is a **Show tags** action that lists all the tags. Both local and already pushed tags can be deleted from this dialog box. For tags present only in the local branch, you can choose to push them to the remote repository. The **Show tags** action is also available in the **Git** menu from the main menu bar.

Working with Submodules

When cloning a repository that contains submodules, all submodules are initialized and cloned as well. When pulling changes from the remote repository, the submodules are also updated. The update of the submodules when performing a pull operation depends on the **Update all submodules after pulling changes from the remote repository** option from **Options > Preferences > Plugins > Git Client** (the option is enabled by default).

When checking out a branch for a parent repository with submodules, the submodules are also checked out at the index in the parent repository to reflect the actual state of the repository at that particular time.

To open and work with a Git submodule, use the **Submodules** action from the *Ellipsis* (*three vertical dots*) menu (on the right side of the **Git Staging** view's toolbar) and select the desired submodule from the presented combo box. As an alternative, if the submodule is modified and is presented in the **Unstaged files area** ([on page 495](#)), the **Open** contextual menu action can be used to open it.

The tooltip of a modified submodule shown in the **Unstaged files area** ([on page 495](#)) presents information about the currently and previously tracked commits.

Showing the Repository History

To show the history, invoke the **Show current repository history** action from the toolbar of the *Git Staging* panel (look for the clock icon) or go to **Window > Show view > Git History**. This opens the **Git History** view at the bottom of the application.

In the toolbar, you can choose which branches will have the history shown. You have the following options:

- **Current branch** - Shows the commits for the current branch, including unpushed local and unpulled remote commits.
- **Current local branch** - Shows the commits only for the current local branch, including unpushed local commits.
- **All branches** - Shows the commits for all branches (both local and remote), including unpushed local and unpulled remote commits.
- **All local branches** - Shows the commits only for local branches, including unpushed local commits.

For each commit in the history table, the following actions are available:

- **Create branch** - Used to create a new branch starting from the selected commit. The new branch is automatically checked out by default. To disable this behavior, deselect the **Checkout branch** option in the *Create branch* dialog box.

**Tip:**

To publish a new branch to the remote repository and start tracking that branch, you need to simply push the local branch using the dedicated action from the **Git Staging** side-view.

- **Create tag** - Used to create a new tag on the selected commit. You have the option to push the tag to the remote repository.
- **Checkout** - Used to checkout a branch at a specific commit (either in detached head form or by creating a new branch at that commit).
- **Revert commit** - Used to create a new commit that reverts all the changes from the selected commit.
- **Reset "[branch_name]" to this commit** - Used to undo changes by moving the HEAD of the current branch to the selected commit.

The **Git History** view presents all the affected resources for each commit in a list, in the bottom-right area. It includes a text filter field at the top that you can use to conduct searches (e.g. by Date, Author, or Commit ID). An included revision graph helps you to understand how commits connect with one another. For each resource, the following actions are available in the contextual menu:

- **Open** (available for added and modified resources) - This action opens the selected resource.
- **Open previous version** (available for deleted resources) - This action opens the version of the selected resource from before its deletion.
- **Open working copy version** (available for added and modified resources) - This action opens the working copy version of the selected resource. It also works if the resource has been renamed in between versions.
- **Reset file to this commit** (available for added and modified resources) - This action checks out the selected version of the resource, overwriting its current version. It does not work if the resource has been renamed or deleted in between versions.
- **Compare with previous version** (available for modified resources) - This action compares the selected version of the selected resource with the previous one using the **Compare Files** tool ([on page 313](#)).
- **Compare with working copy version** (available for modified resources) - This action compares the selected version of the selected resource with the current one using the **Compare Files** tool ([on page 313](#)).
- **Compare with each other** (available when selecting 2 versions of a single file) - This action compares the selected versions with each other using the **Compare Files** tool ([on page 313](#)).

Blame

The contextual menu of each unstaged resource contains a **Show blame** action that opens the selected resource in the main editing area and colors the editor lines with different colors based on the revision

information. Selecting a line in the opened editor will highlight the corresponding entry from the history table in the **Git History** side-view.

This action is also available in the contextual menu of the current editor and of the Git resources from the **Project** side-view ([on page 251](#)).

Unstaged Files Area

In the **Unstaged files** area, you will see all the modifications that have occurred in your working copy (files that have been modified, new files, and deleted files) and are not part of the next commit.

- Various actions are available in the contextual menu (**Open**, **Open in compare editor**, **Stage**, **Discard**, **Show history**, **Show blame**, and more).
- You can stage files (i.e. move them to the **Staged files area** ([on page 495](#))) using the actions from the toolbar found above the top-right corner of this area. You can choose between staging all the files, by clicking the **Stage all** button (double arrow icon), and staging specific files, by selecting them and clicking the **Stage selected** button (single arrow icon).
- You can switch between the list view and the tree view by clicking on the **Switch to tree/list view** button positioned to the right of the staging buttons.

Staged Files Area

In the **Staged files** area, you will see all the resources that are ready to be committed. The files from this area can be unstaged and sent back to the **Unstaged files area** ([on page 495](#)). This area has actions similar to those from the **Unstaged files area** ([on page 495](#)), with the exception of the **Show history** and **Show blame** actions that are not available here.

Comparing Changes and Conflict Resolution

At any time, if you want to see the differences between the last commit and your current modifications, you can double-click a file from either the **Unstaged files area** ([on page 495](#)) or **Staged files area** ([on page 495](#)), and the **Compare Files** ([on page 313](#)) window will appear and highlight the changes.

If the file has a conflict (has been modified both by you and another), **Oxygen's 3-Way file comparison feature** ([on page 316](#)) will show a comparison between the local change, the remote change, and the original base revision.

Committing

After staging the files, on the bottom of the view you can provide a commit message and commit them to your local repository. For convenience, you can also select a previously provided message.

In the toolbar above the **Commit message** text area, there are a few toggle buttons that affect your commit if they are enabled:

- **Amend last commit** - Enabling this option is a convenient way to modify the most recent commit. It lets you combine staged changes with the previous commit instead of creating an entirely new commit. It can also be used to simply edit the previous commit message without changing its snapshot. This action should not be performed on public commits (commits that were pushed to the remote repository).
- **Automatically push changes to remote when committing** - If this option is enabled, when a commit is performed, the committed changes are also pushed to the remote repository.

Push/Pull (with Merge or Rebase)

To push your local repository changes to the remote repository, use the  **Push** button from the view's toolbar (up arrow).

To bring the changes from the remote repository into your local repository, use one of the actions from the **Pull** drop-down menu located on the toolbar (). You can choose between **Pull (merge)** and **Pull (rebase)**. The invoked action is promoted as the current action of the toolbar button.



Note:

When pushing a local branch that does not have a corresponding remote branch, a remote branch will automatically be created with the same name as the local branch.

File Conflict Solving Workflow

After editing a file, committing it to the local repository, and trying to push it to the remote repository, if a warning appears about not being up to date with the repository, follow these steps:

1. Pull the data from the repository using one of the *Pull* actions.
2. In the **Unstaged files area** ([on page 495](#)), select each conflicted file and resolve the conflicts. You can do this, for example, by opening the conflicted files in the compare editor, either by double-clicking on them or by using the contextual menu action, and then choose what changes you want to keep and discard, and save the document. You can also use the **Resolve using Mine**, **Resolve using Theirs**, or **Mark as resolved** actions from the contextual menu of a resource.
3. If you choose to use the compare editor, after you close it, the file will be staged automatically and moved to the **Staged files area** ([on page 495](#)).

At this point, the next actions depend on which *Pull* action was chosen:

- **Pull (merge):**
 1. When all the conflicts are resolved and no more files are left in the **Unstaged files area** ([on page 495](#)), the changes can be committed.
 2. Enter a message and commit. You will now have new changes to push.
 3. Push the changes to the remote repository.

**Note:**

You can abort the merge by clicking the **Abort merge** button. This will revert the repository to its previous state prior to the pull attempt.

- **Pull (rebase):**

1. When all the conflicts are resolved, click the **Continue rebase** button.
2. Push any outgoing changes.

**Note:**

You can abort the rebase by clicking the **Abort rebase** button. This will revert the repository to its previous state prior to the pull request.

Working with Large Files in Git

Git repositories are designed to track changes to text-based files, such as source code, and are optimized for small file sizes. Large binary files (such as image or video files) can slow down the Git repository and make version control difficult. Git's *Large File Storage (LFS)* provides a solution to this problem by efficiently storing and managing large files outside of the repository.

To start using LFS, follow the instructions at <https://git-lfs.com/>. Once you have installed LFS and defined the tracked large files in your repository, large files will be automatically detected and handled properly.

**Warning:**

If you are using a proxy, you need to set the `HTTPS_PROXY` environment variable for git operations that involve LFS for it to work correctly.

Setting the `HTTPS_PROXY` environment variable on Windows from the command line

```
set HTTPS_PROXY=http://my.proxy.com:5000
```

The Project View and the Current Editor

For resources from Git repositories, this add-on also contributes a variety of actions in the contextual menus of the **Project side-view** ([on page 251](#)) and the current editor (**Text** and **Author** modes). These actions include: **Show history**, **Show blame**, **Git Diff** (only in the *Project* view), and **Commit** (only in the *Project* view).

Git Branch Manager

To show all the local and remote branches, click the **Show Git Branch Manager** button on the toolbar of the *Git Staging* panel (look for the branches icon) or select **Git Branch Manager** from **Window > Show view**. By default, the *Git Branch Manager* is presented to the right of the editing area.

The *Git Branch Manager* side-view displays all the branches as a tree. The tree can be filtered using the text field at the top of the panel and you can reload the information by using the **Refresh** action. When hovering the

cursor over a branch name, a tooltip is displayed that provides information about the last commit performed on that branch (such as the author and the date of the commit).

The following actions are available in the contextual menu for each local branch:

- **Checkout** - Checks out the selected branch and switches the local repository to the selected branch.
- **Create branch** - Creates a new branch using the selected branch as the starting point. The new branch is automatically checked out by default. To disable this behavior, deselect the **Checkout branch** option in the *Create branch* dialog box.



Tip:

To publish a new branch to the remote repository and start tracking that branch, you need to simply push the local branch using the dedicated action from the **Git Staging** side-view.

- **Merge "SELECTED_BRANCH" into "CURRENT_BRANCH"** - Merges all the changes from *SELECTED_BRANCH* into *CURRENT_BRANCH*.
- **Squash merge "SELECTED_BRANCH" into "CURRENT_BRANCH"** - Merges all the changes made on the *SELECTED_BRANCH* since it diverged from the *CURRENT_BRANCH*, on top of the *CURRENT_BRANCH*, and it records the results in a new commit.
- **Delete** - Deletes the selected branch.

For the remote branches, the **Checkout branch** action checks out the selected branch and creates a local branch from the selected remote branch.

Preferences

The *Git Client* add-on contributes a preferences page to Oxygen JSON Editor. To access it, go to **Options > Preferences > Plugins > Git Client** or **Git(Menu) > Settings > Preferences** or click the **Ellipsis menu** button from the toolbar of the **Git Staging** view and select **Preferences**. This preferences page includes the following options:

- **When detecting a Git repository inside a newly opened project** - This determines what happens to the current working copy when a project that contains a Git repository is opened in the **Project side-view** ([on page 251](#)). You can choose between:
 - **Always switch to the new working copy**
 - **Never switch to the new working copy**
 - **Always ask** (default value)
- **Notify me about new commits in the remote repository** - When this option is selected, Oxygen JSON Editor will show notification messages when it detects that new commits have been pushed to the remote repository. By default, this option is not selected.

- **Update all submodules after pulling changes from the remote repository** - If this option is selected, when a repository is updated using the *Pull* operation, all sub-modules are updated as well. This option is selected by default.
- **Detect and open Oxygen projects (.xpr) from opened working copies** - If this option is selected (by default, it is not selected) and a working copy that contains one or more project files (**.xpr**) is opened:
 - If one **.xpr** file is found, it will be opened automatically.
 - If more than one **.xpr** files are found and none of them are opened, a dialog box will appear where the **.xpr** can be selected.
- **Validate each file before committing** - When this option is selected, each file to be committed will be validated individually. If validation problems are detected, the commit operation will be stopped and a dialog box will appear informing you that the problems can be viewed in the "Results" area (in the "Git pre-commit validation" tab). If the **Reject commit when validation problems occur** option is selected (the default state), the dialog box will include a **Commit anyway** button that allows the commit operation to be completed even if validation issues have been found.
- **Global Options/Project Options** - If you select **Project Options**, the settings are stored in the project file (**.xpr**) that can easily be [shared with other users \(on page 189\)](#).

Under the preferences page, there is the "SSH Connections" page, which includes the following options:

- **Use SSH agent** - when this option is selected, the selected SSH agent support is used to benefit from the SSH keys already stored in it. On **Linux**, **OS X**, and **BSD**, the only agent communication mechanism supported is the usual communication via a Unix domain socket. On Windows you can choose between:
 - **Pageant**
 - **Win 32 Open SSH**(default value)

Editor Variables

The *Git Client* contributes the following editor variables:

- **\${git(working_copy_name)}** - The name of the working copy directory.
- **\${git(working_copy_path)}** - The absolute file path of the working copy directory.
- **\${git(working_copy_url)}** - The location of the working copy directory as a URL.
- **\${git(short_branch_name)}** - The short name of the current branch (e.g. `dev`).
- **\${git(full_branch_name)}** - The full name of the current branch (e.g. `refs/heads/dev`).

Resources

For more information about the Git Client add-on, as well as details regarding other popular add-ons that extend the functionality of Oxygen JSON Editor, see the following webinars/presentations/articles:

- [Webinar: Add-ons You Can Use for Technical Writing](#)
- [Webinar: Extending the Functionality of Oxygen Using Add-ons](#)

- [Webinar: Docs as Code: Documentation Management Inspired by Software Development](#)
- [Webinar: Using DITA for Small Technical Documentation Teams](#)
- [Blog Post: Using Git For Technical Writing](#)

Project Validation

Validating Each File Before a Commit

To enable automatic validation of each file before a commit, select the **Validate each file before committing** option (on page 499) in the **Git Client** preferences page (**Git > Settings > Preferences**, or **Options > Preferences > Plugins > Git Client**, or click the **Settings** button from the toolbar of the **Git Staging** view and select **Preferences**).

If validation problems are detected, the commit operation will be stopped and a dialog box will appear informing you that the problems can be viewed in the "Results" area (in the "Git pre-commit validation" tab). If the **Reject commit when validation problems occur** option is selected, the dialog box will include a **Commit anyway** button that allows the commit operation to be completed even if validation issues have been found.

Resources

For more information about the validation features that can be enabled for the Git Client, watch our video demonstration:

<https://www.youtube.com/embed/pePngNw2J94>

Migration/Conversions

Batch Documents Converter Add-on

Oxygen JSON Editor offers an add-on that contributes actions in the following submenus:

- **Batch Documents Converter** submenu located in the **Tools** menu and the contextual menu of resources in the **Project** view.
- **Additional conversions** submenu located in **File > Import/Convert**.
- **Import** submenu located in the **Append child**, **Insert Before**, and **Insert After** submenus from the contextual menu of the **DITA Maps Manager** view when the opened DITA map is a local file.

The first time you invoke any of these actions, Oxygen JSON Editor will ask you if you want to install it and offer a wizard to help with the installation process.

Once installed, you need to restart Oxygen JSON Editor and those same actions will then contain the list of available conversions. Selecting an action from the submenu will open a dialog box where you can configure the options for the corresponding conversion. You can batch convert between the following formats:

- HTML to XHTML
- HTML to DITA

- HTML to DocBook4
- HTML to DocBook5
- Markdown to XHTML
- Markdown to DITA
- Markdown to DocBook4
- Markdown to DocBook5
- Word (.doc or .docx) to XHTML
- Word (.doc or .docx) to DITA
- Word (.doc or .docx) to DocBook4
- Word (.doc or .docx) to DocBook5
- Excel to DITA
- Confluence to DITA
- DocBook to DITA
- OpenAPI to DITA
- JSON to XML
- XML to JSON
- JSON to YAML
- YAML to JSON
- YAML to XML
- XML to YAML

When actions are invoked from the contextual menu of the **DITA Maps Manager** view, the resulting documents from the conversion are automatically inserted in the map as follows:

- Actions from **Append child** inserts map nodes as children of the currently selected node.
- Actions from **Insert Before** inserts map nodes as siblings of the currently selected node, above the current node in the map.
- Actions from **Insert After** inserts map nodes as siblings of the currently selected node, below the current node in the map.

Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** to quickly initiate the installation process:

A rectangular button with rounded corners, a thin border, and the word "Install" centered inside.

Install

Manual Installation

To manually install the **Batch Documents Converter** add-on:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box.
2. Enter or paste <https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml> in the **Show add-ons from** field or select it from the drop-down menu.
3. Select the **Batch Documents Converter** add-on and click **Next**.

4. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
5. Restart the application.

Result: A **Batch Documents Converter** submenu will now be available in the **Tools** menu and in the contextual menu. This submenu will contain a list of the various types of available conversions. Selecting one of the types of conversions will open a dialog box where you can configure options for the conversion.

Configuration

Options for configuring the conversions can be found in the preferences page of the add-on (**Options > Preferences > Plugins > Batch Documents Converter**) or in the conversion dialog box.

Conversions from Word (Word Styles Mapping)

The conversions from MS Word work best if you only use the MS Word styles to semantically mark up your document. It is important that sections from the Word document are well defined using the heading styles.

Use the **Word styles mapping** option from the **Batch Documents Converter** preferences page to configure any of the types of Word conversions (Word to HTML, Word to DITA, Word to DocBook4, and Word to DocBook5) by setting a mapping between Word elements and styles to the corresponding HTML element.

If the Word document contains paragraphs formatted with custom styles that are not based on default styles, they have to be set in the Word styles mapping configuration. Those that are not set will be converted into simple paragraphs.

The styles mapping configuration is inherited between styles. If you use a custom style that is based on a default style, the default style mapping configuration will be inherited and also used for the custom style. The mapping from the base style is not inherited if the custom style has a mapping defined in the Word styles mapping configuration.

To define a mapping in the **Word Styles Mapping** table, you can use the already defined default configuration. For example, if you use a custom Word style named `Document Title` that is not based on a default style, you can map this to the HTML "h1" element:

Word element	Word style	HTML elements
p	Document Title	h1:fresh

The resulting 'h1' element will be transformed into the corresponding element when converting to DITA, DocBook 4, and DocBook 5.

The **Word styles mapping** table contains the following columns:

Word element

This column allows one of the following Word elements:

- **p** - Word paragraph
- **r** - Word run
- **b** - bold text
- **i** - italicized text
- **u** - underlined text
- **strike** - strikethrough text
- **table** - table
- **p:unordered-list(x)** - unordered list (where 'x' is the nesting level of the list)
- **p:orderd-list(x)** - ordered list (where 'x' is the nesting level of the list)

Word style

This column can be used to map a paragraph, run, or table with a specific style (referenced by name).

Styles can also be referenced by style ID. This is the ID used internally in the .docx file. To map a paragraph or run with a specific style ID, append a dot followed by the style ID in the **Word element** column (for example: `p.Heading1`).

HTML elements

This column can be used to map the resulting HTML elements. It allows a single element or multiple nested elements.

The nested elements can be declared by using the '>' character (for example: `ul > li`).

The `class` attribute can be specified on the resulting HTML elements by appending a dot followed by the class value, after the element (for example: `p.myClass`).

When converting Word to DITA, these `class` attributes are automatically converted to `outputclass` attributes. This may be useful if you want to apply extra processing on the resulting DITA document using a custom XLS stylesheet.

The `:fresh` syntax can be used to create new elements. If it is not used, the converter will try to reuse the element and close it only when it is necessary.

For example, if the following configuration is set:

p	Heading 1	h1

When the converter finds consecutive Word paragraphs with the style named `Heading 1`, these will be converted into a single `h1` element that contains the text appended from all of the Word paragraphs.

If `h1:fresh` is set in the last column, the converter will create separate `h1` elements.

The `:separator('separator_string')` syntax can be used to specify a separator between paragraphs that are merged when `:fresh` syntax is not specified.

For example, if the following configuration is set:

p	Code Block	pre:separator('\n')

when the converter finds multiple consecutive paragraphs styled with the `Code Block` style, it will merge them into a single `<pre>` HTML element and the "\n" (new line) separator will be used between merged text.

To ignore elements, the '!' character can be added in the **HTML elements** column.

The **Export** button can be used to export the word styles configuration to an XML file. This exported file can be used to configure the MS Word from Oxygen JSON Editor by copying the file in the DITA-OT plugin directory: `[OXYGEN_INSTALL_DIR]/frameworks/dita/DITA-OT/plugins/com.oxygenxml.dynamic.resources.converter`.

The **Import** button allows you to import the word styles configuration from an exported XML file.



Note:

The **Word styles mapping** configuration is applied only for the newer version of MS Word files formatted in the Microsoft Office Open XML (DOCX) format.

Maximum Heading Level for Creating Topics

The **Maximum heading level for creating topics** option from the **Batch Documents Converter** preferences page allows you to set a maximum heading level that the converter will process as DITA topics. The headings with a higher level will be converted to `section` elements.

When the output is a DITA topic, this option sets the maximum heading level that will be converted as a nested topic in the document.

When the output is a DITA map, this option sets the maximum heading level that will be extracted as a DITA topic file and referenced in the DITA map hierarchy.



Note:

This option only applies to the HTML to DITA and Word to DITA conversions.

Word to DITA

The **Create DITA maps from Word documents containing multiple headings** option from the conversion dialog box allows you to decide whether the output will be a DITA map or a DITA topic. When this option is selected, the sections from your Word document marked by titles or headings will be separated into individual DITA topics and referenced in a DITA map. If the word document does not contain multiple sections, the output will be a single topic. When this

option is not selected, the output will be a topic with nested topics and sections according to the number of titles and headings from the Word document.



Note:

Mathematical equations in Word documents should be automatically converted to MathML equations if they are in Office Math Markup Language (OMML) format. If the mathematical equations are in Microsoft Equation Editor format, they first need to be converted to the newer OMML format. See: <https://support.microsoft.com/en-us/office/editing-equations-created-using-microsoft-equation-editor-08a44b8c-ae15-41a7-bc15-7239890c0cec>.

Markdown to DITA

The **Create DITA maps from Markdown documents containing multiple headings** option from the conversion dialog box allows you to decide whether the output will be a DITA map or a DITA topic. When this option is selected, all headings from your Markdown document will be separated into individual DITA topics and referenced in a DITA map. If the Markdown document does not contain multiple headings, the output will be a single topic. When this option is not selected, the output will be a topic with nested topics or sections according to the number of headings from the document.

The **Create short description elements** option from the conversion dialog box allows you to decide whether or not the `shortdesc` elements are created in the output DITA document. When this option is selected, the first paragraph before the headings from the Markdown document will be converted into DITA short description elements. When this option is not selected, the output will not contain the short description element.

HTML to DITA

The **Create DITA maps from HTML documents containing multiple headings** option from the conversion dialog box allows you to decide whether the output will be a DITA map or a DITA topic. When this option is selected, the headings from your HTML document will be separated into individual DITA topics and referenced in a DITA map. If the HTML document does not contain multiple sections, the output will be a single topic. When this option is not selected, the output will be a topic with nested topics or sections according to the number of headings from the document.

The **Ignore HTML 'div' elements** option from the conversion dialog box allows you to decide whether or not the `<div>` elements will be ignored. When this option is selected, all `<div>` elements will be ignored. When this option is not selected, only `<div>` elements that include the `@class` or `@id` attribute will be handled by the converter.

Confluence to DITA

The **Confluence to DITA** conversion processes the HTML content generated by the Atlassian® Confluence (see <https://www.atlassian.com/software/confluence>) export process. To export

Confluence content to HTML, log in to your Atlassian® Confluence account and navigate to the specific space that you want to export. Then go to **Space Settings > Export space** and choose to export it as HTML. The resulting [index.html](#) file must be provided in the **Input files** list from the conversion dialog box.

DocBook to DITA

The **Create DITA maps from DocBook documents containing multiple sections** option from the conversion dialog box allows you to decide whether the output will be a DITA map or a DITA topic. When this option is selected, the sections from your DocBook document will be converted into individual DITA topics and referenced in a DITA map. When this option is not selected, the output will be a single topic with nested topics.

OpenAPI to DITA

The **OpenAPI to DITA** conversion can be used to convert JSON or YAML files that use and conform to the OpenAPI specification (versions 2.0, 3.0, or 3.1) into DITA documents. The **Create DITA maps from OpenAPI documents** option from the conversion dialog box allows you to decide whether the output will be a DITA map or a DITA topic. When this option is selected, the converter will create separate DITA topics for the introduction (including OAS 'Info', 'Server', 'Security Requirement' and 'External Documentation' objects), 'Tag', 'Operation', 'Callback', and 'Components' objects. These topics will be referenced in a DITA map. When this option is not selected, the output will be a single topic with nested topics.

Word to DITA Conversion Notes

The following are some notes about the Word to DITA conversion:

- Paragraphs styled with default Word heading styles (or with custom styles based on default Word heading styles) are handled as topics or sections in the converted DITA output.
- You can choose whether the converted output is a DITA map with referenced topics or a single DITA topic. See the [Create DITA maps from Word documents containing multiple headings \(on page 504\)](#) option.
- You can choose the level of headings that are converted as topics or sections. See the [Maximum Heading Level for Creating Topics \(on page 504\)](#) option.
- You can customize the conversion by adding mappings from your own Word styles to HTML elements. The configured HTML element is converted to the proper DITA element. The `@class` attribute is transformed to the DITA `@outputclass` attribute. See the [Conversions from Word \(Word Styles Mapping\) \(on page 502\)](#) section.
- Ordered and unordered lists are converted to DITA and the list level is preserved.
- Bold, italic, underline, strikethrough, superscript, and subscript styles are converted to the corresponding DITA elements.
- The formatting of table properties (such as borders) is currently ignored, but the formatting of the text inside the table is treated the same as in the rest of the document. Only the header row formatting is taken into account when converting tables to DITA.
- Footnotes and endnotes are converted.

- Images embedded in Word documents are saved to separate files and referenced in the generated DITA topics.
- Links (cross-references and external links) are converted.
- Line breaks are taken into account.
- Mathematical equations in Word documents are automatically converted to MathML equations if they are in **Office Math Markup Language** (OMML) format. If the mathematical equations are in **Microsoft Equation Editor** format, they first need to be converted to the newer **OMML** format.
- Symbols are converted.
- Index entries are converted.
- The Table of Contents is ignored in the DITA result.

Resources

For more information about the Batch Converter add-on, as well as details regarding other popular add-ons that extend the functionality of Oxygen JSON Editor, see the following resources:

- Video: [Integrating REST-API Content into DITA Documentation in Oxygen](#)
- Blog post: [Migrating MS Word to DITA Using the Batch Documents Converter](#)
- Webinar: [Working with DITA in Oxygen - Migrating to DITA and Refactoring](#)
- Webinar: [Integrating Various Document Formats \(OpenAPI, Word, Markdown, HTML, Excel\) into DITA Documentation](#)
- Webinar: [Extending the Functionality of Oxygen Using Add-ons](#)

Terminology

Terminology Checker Add-on

Oxygen JSON Editor offers an add-on that provides support for checking terminology. Once the add-on is installed, you can create a terminology file with a set of rules for each term (or sequence of characters) you want flagged. After referencing the custom file, Oxygen JSON Editor will automatically highlight matched terms in the **Author** visual editing mode and offer some contextual menu actions.



Tip:

The terminology checker works for any document opened in the **Author** visual editing mode, including XML file types, and JSON and HTML5 document types.

Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** to quickly initiate the installation process:

Install

Manual Installation

To manually install the add-on, follow this procedure:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box.
2. Enter or paste <https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml> in the **Show add-ons from** field or select it from the drop-down menu.



Note:

If you have issues connecting to the default update site, you can [download the add-on package](#), unzip it, then use the **Browse for local files** action in the **Install new add-ons** dialog box to locate the downloaded `addon.xml` file.

3. Select the **Terminology Checker** add-on and click **Next**.
4. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
5. Restart the application.

Creating Custom Rules for the Terminology Checker

To create your own custom rules for the terminology checker, follow this procedure:

1. Create a terminology file. There is a template available to help you get started in the . Click the **New** button on the toolbar or select **File > New** and search for the *Terminology File* template. Here is an example of the structure for this type of file:

```
<incorrect-terms lang="en">
  <incorrect-term ignorecase="true">
    <match>Oxygen</match>
    <suggestion></suggestion>
    <message>Product name should be inside a tag.</message>
  </incorrect-term>
</incorrect-terms>
```

2. Save the newly created terminology rules XML file either in a new subfolder named **oxygen-term-checker** located in the current project folder (the current project opened in the application **Project** view), or in a custom folder.
3. If you saved the terminology file in a custom folder path, go to the **Options > Preferences > Plugins > Terminology Checker** preferences page and set the **Additional Terminology folder** path to point to that folder.
4. Click **OK** several times to apply the changes and close the preferences dialog box.

Result: If any of the terms (or sequence of characters) that are defined in the terminology file are detected in any open file, Oxygen JSON Editor highlights the matches in the **Author** visual editing mode.



Note:

If you have a folder named **oxygen-term-checker** in the current project that is open in the **Project** view, all the files in that folder will also be loaded by the terminology checker.

Structure of Terminology Rules File

The following elements can be used in the terminology rules XML file:

`<incorrect-terms>`

This is the root element of the XML rules file.

You can specify the `@lang` attribute on the `<incorrect-terms>` root element. When set, the terms defined in this terminology file are applied when the closest `@xml:lang` attribute of the checked node matches the value specified. Not setting this attribute means that the incorrect terms are applied for all nodes.

If the `@xml:lang` attribute is not defined in your document, the language specified in the [Spell Check preferences \(on page 156\)](#) is used.



Note:

If the value of the document's `@xml:lang` attribute is not a superset of the value of the `@lang` attribute for the `<incorrect-terms>` element, there will not be a match.

Table 3. Language Matching Matrix

@lang value for <code><incorrect-terms></code> element	@xml:lang value	
	en	en_US
en	match	match
en_US	not matched	match

You can specify the `@phase` attribute on the `<incorrect-terms>` root element. The value of this attribute is inherited by `<incorrect-term>` children nodes. Not setting this attribute means the default phase is used.

The allowed values are:

- **always** - Incorrect terms are always presented (default value).
- **editing** - Incorrect terms are shown when the document is opened in the **Author** mode.
- **validation** - Incorrect terms are shown when the document is checked from a validation scenario.

For example, set this attribute if you want to apply the most important rules when validating with the *Validate and Check for Completeness* action, while still keeping them applied in the editing window.

`<incorrect-term>`

Defines ways to match and correct an incorrect term. The `<incorrect-term>` element must include a `<match>` element.

The `@ignorecase` attribute specifies whether or not the match is case-sensitive.

The `@severity` attribute can be set to one of the following values: `info`, `warning`, or `error`. Example:

```
<incorrect-term severity="error">
  <match>he</match>
  <message>Pronouns should be avoided.</message>
</incorrect-term>
```

An experimental `@part-of-speech` attribute can be set on the `<incorrect-term>` element with the value set to a part of speech tag (for example: `adjective`, `verb`, etc.) If set, when scanning for terminology problems, the problem is presented only if the term's part of speech matches the one specified. The processor used to identify the part of speech is *Apache OpenNLP* and this feature is supported only for the English language.



Note:

The results may not be 100% accurate, so you should double-check them.

`<match>`

Specify the text fragment to match.

You can specify the `@type` attribute on the `<match>` element, with the values `character`, `whole-word`, or `regular-expression`. The default value is `whole-word`, unless the matched term contains Japanese, Chinese, or Korean characters because Asian languages often do not use spaces to separate words. Example:

```
<incorrect-term>
  <match type="character">ing</match>
  <message>Progressive tense should not be allowed</message>
</incorrect-term>
```

`<suggestion>`

The `<suggestion>` element can be left blank or there can be one or more of them inside the `<incorrect-term>` element. It supports regular expressions grouping.

If you want to replace the match with an XML fragment, you can set the `@format` attribute on the `<suggestion>` element with the value `xml`. For example:

```
<incorrect-term ignorecase="true">
  <match type="whole-word">Oxygen XML Editor</match>
  <suggestion format="xml">&lt;ph keyref="oxygen"/></suggestion>
  <message>Replace all occurrences of product with key reference.</message>
</incorrect-term>
```

<message>

The <message> element is optional. If present, its content is displayed in a tooltip when you hover over a highlight. It supports regular expressions grouping.

<link>

The <link> element is optional. If present, it provides the source for this rule. Example:

```
<incorrect-term ignorecase="true">
  <match type="whole-word">Oxygen XML Editor</match>
  <suggestion format="xml">&lt;ph keyref="oxygen"/></suggestion>
  <link>https://www.oxygenxml.com/doc/ug-editor/topics/terminology-checker.html</link>
</incorrect-term>
```

<xpath-context>

The <xpath-context> element can be used to define simple XPath expressions that match specific elements.

You can specify @include and @exclude attributes. The elements covered by this simplified XPath will be checked for matches (or the exclusion of a match). A list of comma-separated XPath values can be used. Example:

```
<xpath-context include="p, div, codeblock">
```

The following are 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.

You can use one or more of the following attribute conditions:



Attention:

Default attribute values are not taken into account.

- `element[@attr]` - Matches all instances of the specified element when it includes the specified attribute.
- `element[not(@attr)]` - Matches all instances of the specified element when it does not include the specified attribute.
- `element[@attr = 'value']` - Matches all instances of the specified element when it includes the specified attribute with the given value.
- `element[@attr != 'value']` - Matches all instances of the specified element when it includes the specified attribute and its value is different than the one given.

Using Vale Rules with the Terminology Checker

The **Terminology Checker** has partial support for applying custom *Vale* rules.

Supported *Vale* scopes: **heading, table.header, table.cell, list, paragraph, code, strong, emphasis, sentence.**

Supported *Vale* extension points: **Existence, Substitution, Occurrence, Repetition, Conditional.**

Result: If any of the terms (or sequence of characters) that are defined in the terminology file are detected in any open file, Oxygen JSON Editor highlights the matches in the **Author** visual editing mode.



Note:

If you have a folder named **oxygen-term-checker** in the current project that is open in the **Project** view, all the files in that folder will also be loaded by the terminology checker. As an example, the Oxygen JSON Editor user guide has a folder with some of the *Microsoft style guide* rules: <https://github.com/oxygenxml/userguide/tree/master/DITA/oxygen-term-checker>. Once the user guide project is open in the Oxygen JSON Editor **Project** view, the add-on will start using those rules to check the content.

Resources: You can find already created *Vale* rules that implement various checks on the following websites:

- *Vale* rules that aim to replicate *Grammarly* checks: <https://github.com/testthedocs/Openly/tree/master/Openly>.
- *Vale* rules that aim to automate the *Microsoft* style guide: <https://github.com/errata-ai/Microsoft/tree/master/Microsoft>.
- *Vale* rules that aim to automate the *Google* style guide: <https://github.com/errata-ai/Google/tree/master/Google>.

Working with the Terminology Checker

The **Terminology Checker** side view shows all problems found in the document. You can right-click each problem to apply possible fixes or to find out more details about it. The tooltip for each problem displays a custom message and more information (e.g. for *Vale* rules, it also displays the name of the *Vale* rule file that defines the rule). You can filter problems based on their severity, match, and message and the toolbar has actions to navigate between problems or to open the **Terminology Checker** preferences page.

You can also right-click problems highlighted in the **Author** visual editing mode to access the following contextual menu actions:

- **Replace with "..."** - Replaces the currently highlighted match with the content inside the `<suggestion>` element.
- **Replace all with "..."** - Replaces all instances of the highlighted match found in the current document with the content inside the `<suggestion>` element.
- **Correct all matching highlights** - Replaces all highlighted matches (all matched terms) within the document with the content inside the first `<suggestion>` element from the terminology file.

The terminology checking can be disabled by clicking the  **Show/Hide Terminology Highlights** toolbar button.



Other Notes:

- The checker automatically skips deleted content with tracked changes and space-preserved elements (e.g. codeblocks).
- When replacements are performed, the capitalization is preserved.
- In the Oxygen JSON Editor **Options > Preferences > Plugins > Terminology Checker** page, you can define the highlight colors to be used for each issue depending on its severity. You can also reference a folder that contains the terminology rules. This folder can contain other folders with terminology files or just the terminology files. The option that controls automatic capitalization can also be found in this preferences page.
- If you select **Project Options** (in the **Terminology Checker** preferences page), the settings are stored in the project file (`.xpr`) that can be [shared with other users \(on page 189\)](#).

Terminology Checker Preferences

The **Options > Preferences > Plugins > Terminology Checker** preferences page contains various settings for configuring tool. The preferences page can be saved at project level to share these settings, as is common for a group of users who use the same project configuration.

Highlight background

You can specify various colors to influence the background colors for terminology highlights that are added in the **Author** visual editing mode.

Highlight decoration

You can specify various colors to influence the highlight decoration styles for terminology highlights that are added in the **Author** visual editing mode.

Editing

Preserve case when performing replacements

Controls whether or not the original letter casing is automatically preserved when replacing words. The option is selected by default.

Report unsupported Vale rules as errors

If selected (default), errors that are related to Vale terms (such as unsupported extension points or invalid properties) are reported. If not selected, unsupported Vale rules are ignored (although an error is still reported if the file is invalid).

Learned terms

Default project terminology folder

Displays the default location where all the terminology rule files (XML or Vale) are stored. By default, the rule files located in the **oxygen-term-checker** subfolder of the current project folder (the current project loaded in the **Project** view) are automatically loaded and used.

Additional terminology folder

You can use this option to specify an additional terminology folder where XML and Vale rule files are located. You can use editor variables such as `${pd}/terms` to specify the path to the terminology folder.

Checking Multiple Resources

Once installed, the terminology checker add-on can be used to batch-check multiple files:

- Right-click on the root of the DITA map opened in the **DITA Maps Manager** view and choose **Check terminology**.
- Right-click a folder in the **Project** view and choose **Check terminology**.
- Create a new validation scenario or edit an existing validation scenario, and add a new validation stage. For the **File type** field, choose **XML Document** and for the **Validation engine** field, choose **Terminology checker**. The validation scenario can be used in multiple ways:
 - In the **Project** view, you can right-click a folder and validate using a specific validation scenario.
 - In the **DITA Maps Manager** view, you can use the **Validate and Check for Completeness** toolbar action and choose to **Batch validate referenced DITA resources**. This will apply the associated validation scenario for each topic or map referenced in the context of the main DITA map.

Terminology Files Contributed from Other Oxygen Add-ons

Any **Oxygen** add-on can contribute terminology files that will be used by the Terminology Checker. The contributed terminology files will be loaded and used if the contributor add-on is enabled.

The following pre-conditions must be fulfilled:

1. The contributor add-on's `plugin.xml` descriptor file should reference the rules folder in the `plugin.xml` as a `librariesFolder` with a `global` scope:

```
<plugin
    id="unique.identifier.name"
    name="My Style Guide"
    description="Style Guide"
    version="1.0"
    vendor="Vendor Name"
    class="ro.sync.exml.plugin.Plugin"
    classLoaderType="preferReferencedResources">
    <runtime>
        <librariesFolder name="Rules_Folder" scope="global"/>
    </runtime>
</plugin>
```

2. The contributor add-on should have a marker file named `oxy-terms-auto-detect` inside the rules folder. The terminology files can be added in the rules folder or organized in subfolders (the Terminology Checker scans the subfolders to identify the terminology files). Inside the `oxy-terms-auto-detect` file, there should be a textual description of the terminology file contents, which is used when presenting add-on contributed terms in the Terminology Checker preferences page (**Options > Preferences > Plugins > Terminology Checker**).

ASD Simplified Technical English Specification (ASD-STE100) Rules

An extra add-on is available that contributes *ASD Simplified Technical English Specification* rules to the **Terminology Checker**. It contains technical rules based on ASD-STE100 (<http://www.asd-ste100.org>), but note that these rules are not endorsed by ASD-STE100.

To install these rules, use the [manual add-on install procedure \(on page 508\)](#) and select **ASD Simplified Technical English Specification (ASD-STE100) Writing Style Guide Rules (experimental)** for the add-on to install.

MS Writing Style Guide Vale Rules

An extra add-on is available that contributes a set of rules based on the *Microsoft Writing Style Guide* to the **Terminology Checker**. It contains a Vale-compatible (<https://github.com/errata-ai/vale>) implementation of the MS Writing Style Guide (<https://learn.microsoft.com/en-us/style-guide/welcome/>) as provided by the *errata-ai* open-source project (<https://github.com/errata-ai/Microsoft>). Note that this project is neither maintained nor endorsed by *Microsoft*.

To install these rules, use the [manual add-on install procedure \(on page 508\)](#) and select **MS Writing Style Guide Vale Rules** for the add-on to install.

Resources

For more information about the Terminology Checker add-on, along with details regarding other popular add-ons that extend the functionality of Oxygen JSON Editor, watch the following webinar:

- [Webinar: Add-ons You Can Use for Technical Writing](#)

Vale Linter for Markdown and HTML Validation Add-on

The Vale Validation add-on runs the [Vale linter](#) over the currently edited file and presents the validation errors in the results area at the bottom of the application. A *Linter* is a tool that automatically verifies specific rules against your code or documentation. This is useful for enforcing a style guide or for catching commonly mistaken branding issues.

Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** to quickly initiate the installation process:

A rectangular button with rounded corners, containing the word "Install" in a sans-serif font.

Manual Installation

To manually install this add-on, follow this procedure:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box.
2. Enter or paste <https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml> in the **Show add-ons from** field or select it from the drop-down menu.



Note:

If you have issues connecting to the default update site, you can [download the add-on package](#), unzip it, then use the **Browse for local files** action in the **Install new add-ons** dialog box to locate the downloaded **addon.xml** file.

3. Select the **Vale Linter for Markdown and HTML Validation** add-on and click **Next**.
4. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
5. Restart the application.

Setup the Oxygen Vale Validation

To set up the Vale validation, follow this procedure:

1. Download and unzip the proper [Vale executable](#) for your OS. On Linux and macOS, you must give executable permission to the Vale executable. You can do this by opening a console in the Vale directory and running:

```
chmod u+x vale
```

2. Go to **Options > Preferences > Plugins > Oxygen Vale Validation** and specify the path to the previously downloaded Vale executable.
3. In the same preferences page, you can also specify a path to a [Vale configuration file \(.vale.ini\)](#). Vale [automatically detects this file](#) by looking 6 levels up in the current file's ancestor directories, but you can also impose one.

Vale Styles

Vale uses collections of individual [YAML](#) files (or "rules") to enforce particular writing constructs. These collections are referred to as *styles* and are organized in a nested folder structure at a user-specified location. The [.vale.ini](#) file is where you will control most of Vale's behavior, including which files to lint and how to lint them. Vale [automatically detects .vale.ini](#), but you can also specify the path to [.vale.ini](#) from the plugin's preferences page (**Options > Preferences > Plugins > Oxygen Vale Validation**).

Third-party Styles

Vale has a growing selection of pre-made styles available for download from its [style library](#).

Validation

After setting up the Vale executable, creating or downloading Vale styles, and specifying the path to [.vale.ini](#), the add-on will intercept the and Manual Validation and contribute errors and warning obtained by running Vale validation over the current file. The errors and warnings are highlighted in the editor.



Note:

Although Vale supports [multiple file formats](#), the Vale Validation add-on currently only supports [Markdown \(*.md files\)](#) and [HTML files](#).

Productivity

Oxygen AI Positron Assistant

The **Oxygen AI Positron Assistant** add-on uses the advanced **Oxygen AI Positron** service to support technical documentation writers throughout their content creation process.

https://www.youtube.com/embed/Do_KWYZfCFg?si=2Mrm7Bh1pmPvehqV

Overview

In a simplified form, technical documentation is often done in two stages: analysis and implementation. In the analysis stage, technical writers could use various resources such as web searches, ChatGPT, or discussions

with colleagues or engineers to further understand the subject that needs to be documented. In the second stage, technical writers would use tools such as Oxygen JSON Editor to write the actual content.

The **Oxygen AI Positron Assistant** add-on provides various ways to use **ChatGPT** to help writers while editing or reviewing the technical documentation. For example, it can be used to receive hints about what to write next, improve the readability of content, or re-structure the content in various ways.

**Note:**

Content received from the **OpenAI ChatGPT** model may be inaccurate or contain misleading information, so it needs to be thoroughly reviewed and revised accordingly.

**Terms:**

The terms of use for the service can be found [here](#).

[AI Positron Assistant Samples Playground](#).

Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** to quickly initiate the installation process:

Install

Manual Installation

To manually install this add-on, follow this procedure:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box. Enter or paste **<https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml>** in the **Show add-ons from** field or select it from the drop-down menu.

**Note:**

If you have issues connecting to the default update site, you can [download the add-on package](#), unzip it, then use the **Browse for local files** action in the **Install new add-ons** dialog box to locate the downloaded **addon.xml** file.

2. Select the **Oxygen AI Positron Assistant** add-on and click **Next**.

**Important:**

There are two different iterations of the add-on and you can only have one or the other installed at once:



- **Oxygen AI Positron Assistant** - The regular version of the add-on.
- **Oxygen AI Positron Assistant (Enterprise)** - This Enterprise version is for those who want to connect to their own OpenAI or MS Azure OpenAI account.

3. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
4. Restart the application.

Result: The **AI Positron Assistant** side view is now available.

Connecting to the Oxygen AI Positron Service

You can use the **AI Positron Assistant** side view to easily configure login details and connect to the **Oxygen Positron Service** in the web browser.

To initiate the connection process, use the **Connect** button in the **AI Positron Assistant** view (or from the user drop-down menu at the top-right corner of the view).



Note:

The **Oxygen Positron Service** uses the OpenAI ChatGPT server version 3.5 API to propose document generation and change suggestions.

AI Server Requests and Credits

Each user has a limit to the number of requests that are sent to the AI server each month and this is managed through the use of **credits**.



Important:

To use your own company-specific AI service, the specific **Oxygen AI Positron Assistant Enterprise** ([on page 528](#)) add-on needs to be installed instead of the one listed above.

Generating and Refining Content

Once you log in to the server, the following actions for generating content are available in the **Actions** drop-down menu at the top of the **AI Positron Assistant** side view, in the main chat panel when a conversation has not yet started, in the **AI Positron Assistant** submenu within the contextual menu, and in the **AI** main menu. The progress and results of triggering an action are displayed in the **main chat pane** ([on page 521](#)):

Content Generation

- **New DITA Topic** - Generates a DITA XML topic based on a text description entered in a popup dialog box.
- **Continue Writing** - Generates additional text based on the content preceding the cursor position.

- **Short Description** - Generates a short description (inside a `<shortdesc>` element) based on a summary of the selected text (or the entire document if there is no selection). You can configure the style and the approximate number of sentences to be generated.
- **Index Terms** - Generates a `<keywords>` element that contains index terms obtained from the selected text (or the entire document if there is no selection).
- **Follow Instructions (available when editing schemas, XSLT stylesheets, and Schematron files)** - Replaces the selected instructions with content generated based on them.

Rewrite

- **Correct Grammar** - Generates a suggestion for correcting the grammar and spelling within the selected content.
- **Improve Readability** - Modifies the selected content to improve readability and fix grammar/spelling errors. If you hover the mouse prompt over this button, a  **Settings** button becomes available in the top-right corner. Clicking the  **Settings** button opens a pop-up window where you can choose the writing level of the content to be generated. You can choose between: **5th grade (Very Easy)**, **8th grade (Plain English)**, and **College (Advanced)**.
- **Use Active Voice** - Generates a suggestion for replacing the selected content with content that has been converted from passive to active voice.
- **Itemize** - Generates a suggestion for converting the selected content into a list of items.
- **Join Items** - Generates a suggestion for converting the selected list of items into a paragraph.

Overview

- **Answer Questions** - Generates answers to questions that the AI finds within the selected content (or the entire document if there is no selection).
- **Generate Questions** - Generates a list of five questions that are answered within the selected content (or the entire document if there is no selection).
- **Summarize** - Generates a summary of the selected content (or the entire document if there is no selection).
- **Readability** - Generates suggestions for changing the selected content (or the entire document if there is no selection) to improve its general readability.

Translation

- **English, German, French, Japanese** - These actions translate the selected text to the target language, while preserving the original XML markup.
- **Other...** - This action behaves like the previous ones, but it allows you to provide the target language. You can either choose from the predefined values or type another one.

Marketing

- **Release Notes** - Creates release notes based on a set of features or issue ticket numbers with optional descriptions.
- **Marketing Post** - Creates a marketing post based on a list of ideas or release notes.
- **Improve SEO** - Rewrites the content to enhance search engine optimization.
- **Pain-Agitate-Solution** - Rewrites the content using a marketing style based on the *Pain-Agitate-Solution* framework.
- **Features-Advantages-Benefits** - Rewrites the content using a marketing style based on the *Features-Advantages-Benefits* framework.

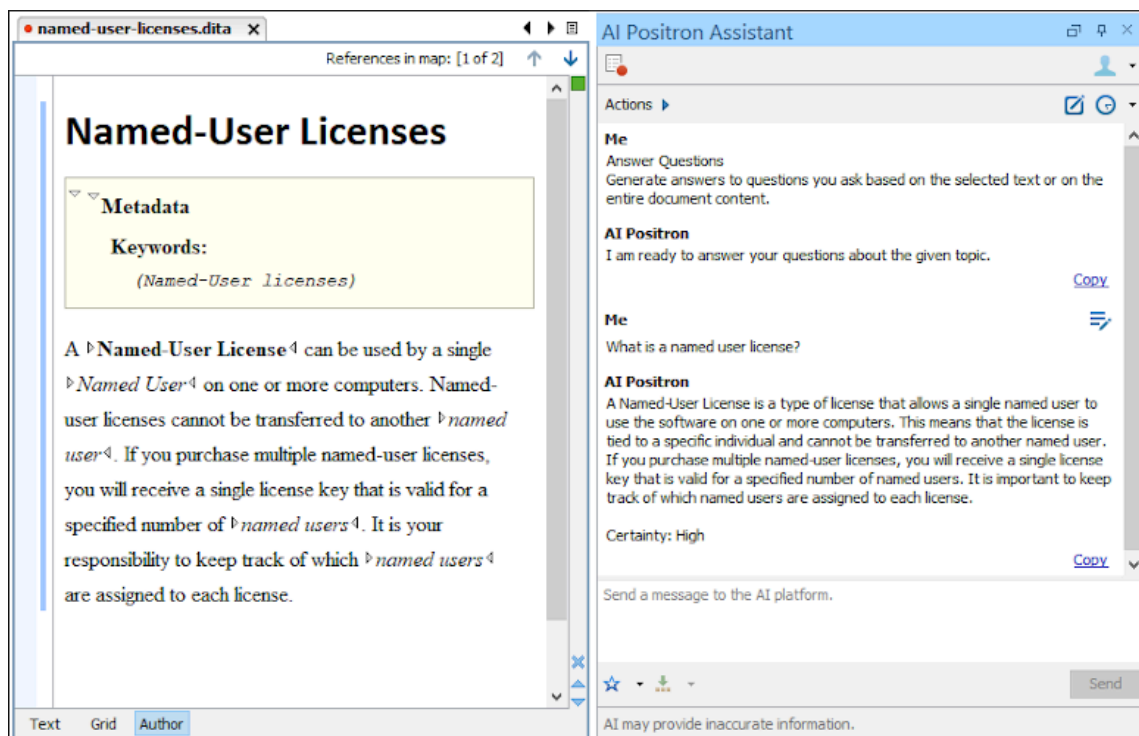
**Tip:**

Custom actions can be configured in the [AI Positron Assistant preferences page \(on page 523\)](#).

AI Positron Assistant View

The add-on provides access to the **AI Positron Assistant** side-view. If the view is not displayed, it can be opened by selecting it from **Window > Show View**.

Figure 136. AI Positron Assistant View



The **Actions** drop-down menu at the top of the **AI Positron Assistant** view contains the available **AI-powered actions that can be used to generate and refine content (on page 519)**. Simply select the action to trigger it. You can hover the mouse cursor over an action to see a description of what the action does. A set of 5 recently used actions are also available in the **Actions** drop-down menu.

The **Record** button in the top-left corner of the view allows you to [create custom actions or prompts by recording changes \(on page 525\)](#).

There is also a user drop-down menu at the top-right corner of the **AI Positron Assistant** view that contains the following:

- **My account** - Opens a webpage where you can see your current subscription package and credit status.
- **Disconnect** - Disconnects **Oxygen** from the `Oxygen Positron Service`.
- **Preferences** - Opens the [Oxygen AI Positron Assistant preferences page \(on page 523\)](#) where you can configure the **AI Positron service address** and provide a **Context** for the user that the AI will use to create more relevant and personalized responses.

The main chat pane presents the results after processing an action and allows you to further refine the responses by sending messages to the **Positron** service platform. When an AI Positron action is triggered, the chat pane displays the progress and results.

The response is received from the server in streaming mode (the AI sends chunks of the response as it is being generated rather than waiting to send the entire response after it is generated). Once the entire response is received from the server, the following actions are available under the response:

- **Insert/Replace** - Inserts the response at the cursor location within the document (or replaces the selected content).
- **Preview** - Allows you to preview the content that would be inserted at the cursor location within the document.
- **Copy** - Copies the response to the system clipboard.

The Chat  **History** drop-down toolbar button makes it easy to go back to previous conversations and continue them.

You can use the bottom pane to refine the response by sending a message to the AI platform and it will generate a new response based upon your message. You can create your own favorite prompts and use supported variables to specify the content that is sent to the platform. You can use the  **Favorites** drop-down button to store a favorite prompt. You can use the  **Insert Variables** drop-down button to select one of the supported variables:

- **\${selection}** - Expands to the currently selected content.
- **\${document}** - Expands to the content of the entire document.



Tip:

Previously sent prompts can be modified directly in the chat thread. Once a prompt is edited, a new chat thread is started based on the new prompt's content. For edited prompts, you can use the

 **Next** /  **Previous** buttons to navigate between chat threads.

To clear the information in the chat pane and start a new chat, click the  **New Chat** button in the top-right corner of the view.

AI Refactoring

The **AI Positron Assistant** add-on contributes an **AI Positron Refactoring** action in the contextual menu (**Refactoring > AI Positron Refactoring**) of both the **Project** and **DITA Maps Manager** views in Oxygen JSON Editor. It can be used to refactor multiple files (local or remote) at once.

You can invoke the **AI Positron Refactoring** action to apply either a predefined AI action or a custom prompt to modify the selected resources. The resulting **AI Positron Refactoring** dialog box presents an estimate of the amount of credits that will be consumed by the operation, and you have the option to preview the changes before applying them over the original content.

For example, you could use the predefined **Translate to** action to translate multiple DITA topics into a certain language or apply the **Correct Grammar** or **Improve Readability** actions on multiple resources.

Oxygen AI Positron Assistant Preferences Page

Various settings can be configured in **Options > Preferences > Plugins > Oxygen AI Positron Assistant**:

Context

The context provides useful information about the user to the AI and is used in each action and chat request to create more relevant and personalized responses.

Load default actions

Specifies if default actions are loaded.

Additional actions folder

You can use this option to specify a local folder where you have stored additional actions.

Enable XPath Functions

Enables the use of AI-specific XPath functions in Oxygen JSON Editor when applying Schematron validation or XSLT transformations. This feature is disabled by default.

Cache responses and reuse them for identical prompts

If enabled (default), responses for identical requests are stored (cached), resulting in fewer requests being sent to the AI server and faster completion times. A **Clear cache** button located to the right of this option can be used to clear the cache.

Cache size

Specifies a maximum limit for the cache size.

Notify me when the number of requests exceeds

You can select this option and specify a number of AI requests that when exceeded, a confirmation dialog box is displayed asking if you want to continue using the XPath AI functions. If you select "No" for the answer, the XPath functions will be disabled.

AI Service Configuration Preferences Page

Various service-related connection settings can be configured in **Options > Preferences > Plugins > Oxygen AI Positron Assistant > AI Service Configuration**:

AI Positron Service address

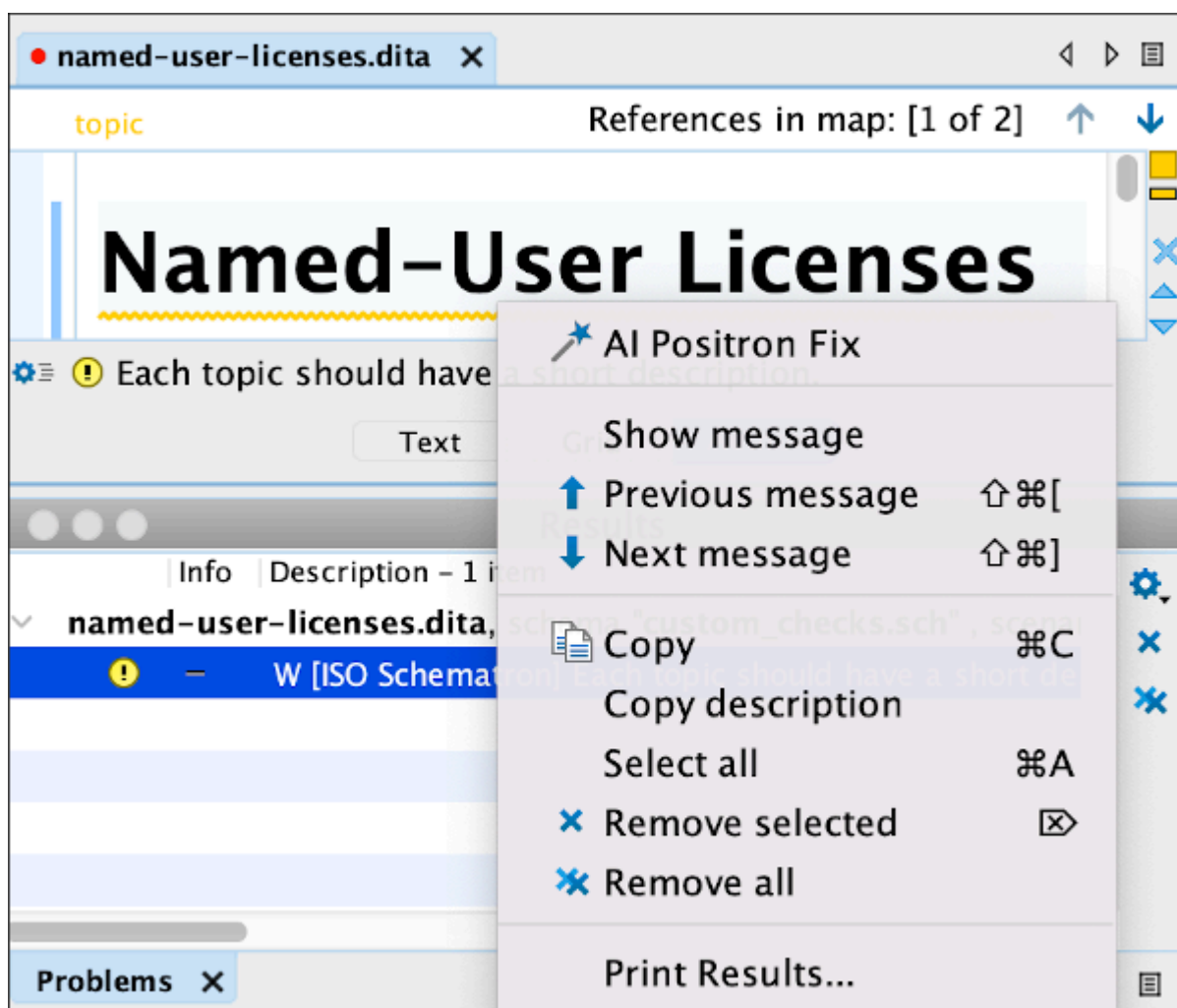
Currently, there is only one public platform that provides this service.

Default model

The default model is used for the chat pane and for actions that do not explicitly specify a fixed model. Each chosen model consumes a certain [number of credits \(on page 519\)](#) per token.

Validation Quick Fixes

When validation problems are displayed in the **Results** pane, you can right-click on a problem and use the **AI Positron Fix** action to ask the **AI Positron** platform for help with fixing the problem. It will propose content in the chat pane (within the **AI Positron Assistant** view) that can be used to solve the problem.



Creating Custom Actions

In the [AI Positron Assistant preferences page \(on page 523\)](#), you can define a reference to a folder that contains custom actions.

Once the add-on is installed, the **New Document** wizard (*on page 225*) can be used to create either a new **AI Positron Custom Action** file that contains a custom action definition in JSON format or an **AI Positron Custom Actions List** that contains a JSON array with multiple defined actions. Validation and content completion are automatically provided for such custom action files. If the action definition files are saved in the custom actions folder defined in the **AI Positron Assistant** preferences page, the **AI Positron Assistant** view should automatically reload its **Actions** drop-down list to include them.

The most simple action defines an action id, title, type, and context:

```
{
  "id": "my.action.id",
  "title": "Improve Grammar",
  "type": "replace-selection-with-fragment",
  "input-type": "markup",
  "context": "Improve grammar in the following content preserving the XML markup:"
}
```

Defined actions can contain expandable parameters and their values can be customized before invoking the action:

```
{
  "id": "my.action.id",
  "title": "Improve Grammar",
  "type": "replace-selection-with-fragment",
  "input-type": "markup",
  "context": "${style} Improve grammar in the following content preserving the XML markup:",
  "expand-params": [
    {
      "name": "style",
      "label": "Style",
      "value": "",
      "alternate-values": ["Use active voice.", "Use passive voice."],
      "alternate-value-labels": ["Active voice", "Passive voice"],
      "choice-type": "single-choice"
    }
  ]
}
```

Create Custom Prompts/Actions by Recording Changes

The  **Record** button in the top-left corner of the view allows you to create new AI actions. It opens the **Record examples for instructions** dialog box where you can provide a set of instructions that are intended for the AI to follow. Then, after clicking the **Start recording** button at the bottom of the dialog box, you can record a collection of examples in the editing area that will help the AI better follow the given instructions. The examples are recorded from the changes made in the open editors.

After providing examples, you need to click the  **Record** button again to stop the recording. You will then have the opportunity to save the final result as either a Positron action or as a favorite chat prompt.

For example, if you want to add DITA markup to menu cascades, you can follow these steps:

1. Click the  **Record** button.
2. In the **Record examples for instructions** dialog box, enter some instructions like: *You are a technical writer. Add DITA markup to menu cascades.*
3. Click **Start recording**.
4. Open a DITA topic that has a menu cascade without markup (for example: `File > Export`).
5. Edit the topic and add markup, transforming it to:

```
<menucascade>
  <uicontrol>File</uicontrol>
  <uicontrol>Export</uicontrol>
</menucascade>
```

6. Click the  **Record** button again to stop the recording. The system generates the following instructions with examples:

```
You are a technical writer. Add DITA markup to a menu cascades.
###
Input:
  <p>File > Export</p>

Output:
  <p><menucascade><uicontrol>File</uicontrol>
  <uicontrol>Export</uicontrol></menucascade></p>

Input: ${selection}
Output:
```

7. In the resulting dialog box, save the final result as either a Positron action or as a favorite chat prompt.

Custom Validation Rules

The add-on contributes two XPath extension functions (available in the content completion proposals for Schematron, XSLT, XQuery, and XPath) that can be used to rephrase content or to perform validation checks on existing content:

ai:transform-content(instruction, content)

Use this function from namespace <http://www.oxygenxml.com/ai/function> to automatically transform content using AI.

The function has two string parameters:

- `instruction` - The OpenAI instruction to be performed on the content.
- `content` - The content to be transformed.

It returns a string that represents the transformed content.

Here is an example of a custom Schematron schema that uses the `transform-content` function to correct the number of words used in a short description:

```
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt3"
  xmlns:sqf="http://www.schematron-quickfix.com/validator/process">
  <sch:ns uri="http://www.oxygenxml.com/ai/function" prefix="ai"/>
  <sch:pattern>
    <sch:rule context="shortdesc">
      <sch:report test="count(tokenize(.,'\s+')) > 50" sqf:fix="rephrase">
        The phrase must contain less than 50 words.</sch:report>
      <sqf:fix id="rephrase">
        <sqf:description>
          <sqf:title>Rephrase phrase to be less that 50 words</sqf:title>
        </sqf:description>
        <sqf:replace match="text()" select="ai:transform-content(
          'Reformulate phrase to be less that 50 words', .)"></sqf:replace>
      </sqf:fix>
    </sch:rule>
  </sch:pattern>
</sch:schema>
```

`ai:verify-content(instruction, content)`

Use this function from namespace `http://www.oxygenxml.com/ai/function` to automatically validate content using AI.

The function has two string parameters:

- `instruction` - The OpenAI instruction to be performed on the content.
- `content` - The content to be validated.

It returns a boolean value that represents the result of the validation.

Here is an example of a custom Schematron schema that uses the `verify-content` function to check a short description for instances of a passive voice:

```
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt3"
  xmlns:sqf="http://www.schematron-quickfix.com/validator/process">
  <sch:ns uri="http://www.oxygenxml.com/ai/function" prefix="ai"/>
  <sch:pattern>
    <sch:rule context="shortdesc">
```

```

    <sch:report test="ai:verify-content('Does the following content has passive
voice?', .)"/>
      sqf:fix="rephrase">The phrase uses passive voice.</sch:report>
    <sqf:fix id="rephrase">
      <sqf:description><sqf:title>Rephrase text to be active voice</sqf:title>
    </sqf:description>
      <sqf:replace match="text()"
        select="ai:transform-content('Rephrase text to be active voice', .)"/>
    </sqf:fix>
  </sch:rule>
</sch:pattern>
</sch:schema>

```

Resources

To see a visual demonstration of the AI Positron Assistant add-on, along with various uses cases for using the tool, see the following recorded webinar: [AI as a Tool for Technical Content Creation](#).

See ways to use AI tools from XSLT stylesheets and Schematron schemas in the following recorded webinar: [Leveraging the Power of AI and Schematron for Content Verification and Correction](#).

Related information

[AI Positron Assistant Samples Playground](#)

[Blog Post About AI Positron Add-on By Tom Johnson](#)

[Webinar: AI as a Tool for Technical Content Creation](#)

[Webinar: Leveraging the Power of AI and Schematron for Content Verification and Correction](#)

Oxygen AI Positron Assistant Enterprise

The **Oxygen AI Positron Assistant Enterprise** add-on provides advanced support for technical documentation writers throughout their content creation process by using a company-specific AI service (by configuring your company's specific **OpenAI** key or **Microsoft Azure OpenAI Service** connection).

A detailed list of actions and functionality available in the add-on is presented in the [Oxygen AI Positron Assistant \(on page 517\)](#) topic.

Installation

To install this add-on, follow this procedure:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box. Enter or paste **<https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml>** in the **Show add-ons from** field or select it from the drop-down menu.

**Note:**

If you have issues connecting to the default update site, you can [download the add-on package](#), unzip it, then use the **Browse for local files** action in the **Install new add-ons** dialog box to locate the downloaded **addon.xml** file.

2. Select the **Oxygen AI Positron Assistant (Enterprise)** add-on and click **Next**.

**Important:**

There are two different iterations of the add-on and you can only have one or the other installed at once:

- **Oxygen AI Positron Assistant** - The regular version of the add-on.
- **Oxygen AI Positron Assistant (Enterprise)** - This Enterprise version is for those who want to connect to their own OpenAI or MS Azure OpenAI account.

3. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
4. Restart the application.

Result: The **AI Positron Assistant (Enterprise)** side view is now available.

Licensing and Configuration

You can configure the company-specific AI service details in the [AI Service Configuration Preferences Page](#) (on page 529).

A [special license key](#) needs to be purchased to enable this direct access. You can use the **Register** button in the **AI Positron Assistant** side view to configure the special license key.

**Important:**

If you want to use the default **Oxygen AI Positron** service instead of a company-specific AI service, the [Oxygen AI Positron Assistant](#) (on page 517) add-on needs to be installed and used instead of the one listed above.

AI Service Configuration Preferences Page

Various service-related connection settings can be configured in **Options > Preferences > Plugins > Oxygen AI Positron Assistant (Enterprise) > AI Service Configuration**:

AI Connector

Specifies the connector type: **OpenAI** or **Microsoft Azure OpenAI**.

OpenAI:

If **OpenAI** is chosen as the connector type, the following settings are available:

Address

The web address of the OpenAI service. By default: `https://api.openai.com`.

Key

The **OpenAI** API key necessary to work with the connector.



Note:

This option does not get saved in the Project-level options.

Organization

For users who belong to multiple organizations, they can specify which organization is used for an API request. Usage from these API requests will count as usage for the specified organization.

Default model

The default model is used for the chat view and for actions that do not explicitly specify a model.

Enable text moderation

This setting applies moderation (checks whether content complies with OpenAI's usage policies) to both the input text sent to the AI service and the response received from the AI service. It is enabled by default.



Tip:

By default, when executing an action using the **OpenAI** connector, three requests are made:

- A moderation on input content request to `configured_web_address/v1/moderations`.
- A completion request to `configured_web_address/v1/chat/completions`.
- A moderation on content returned by AI to `configured_web_address/v1/moderations`.

If your AI service does not require moderation (for example, moderation is already made by chat/completions endpoint) you can disable it by unchecking this checkbox.

Extra Headers

Extra name/value parameters to set in the headers that are specific for the AI requests.

**Tip:**

If the service uses **Bearer Authentication**, you can specify the key in the **Key** text field. If another authentication method is used, the **Key** field can be left empty, and the **Extra Headers** table can be used to set the authentication info on the request header. Note that editor variables can be used in this field and you can set your key in editor variables and specify the value in this table like this: `${env(AI_SERVICE_KEY)}` to access pre-set values of environmental variables.

Microsoft Azure OpenAI:

If **Microsoft Azure OpenAI** is chosen as the connector type, the following settings are available:

Endpoint

The web address where the connector service is located. This value can be found in the **Keys & Endpoint** section when examining your resource from the **Azure** portal. For example: `https://your-company-name.openai.azure.com/`.

Key

The **Microsoft Azure OpenAI Service** key necessary to work with the connector.

Deployment

The deployment name that was chosen when the model was deployed in **Microsoft Azure**.

Extra Headers

Extra name/value parameters to set in the headers that are specific for the AI requests.

**Note:**

You can use [editor variables \(on page 197\)](#) such as `${env(ENV_NAME)}` in all configuration and header parameter values.

**Troubleshooting:**

In case the add-on fails to connect with the custom settings, it might be useful to enable debug logging in the application to see what requests and responses are made to/from the AI server. You can enable [debug logging \(on page 627\)](#) in the application by adding a `logback.xml` file in the application installation folder. A minimal `logback.xml` configuration XML file content to enable logging only for the AI Positron add-on's connections would look like this:

```
<configuration>
  <appender name="R2" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <file>${user.home}/Desktop/oxygenLog/oxygen.log</file>
    <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
      <fileNamePattern>${user.home}/Desktop/oxygenLog/oxygen%i.log.gz</fileNamePattern>
      <minIndex>1</minIndex>
      <maxIndex>20</maxIndex>
    </rollingPolicy>
  </appender>
  <root>
    <level>debug</level>
    <appender-ref ref="R2">

```



```

</rollingPolicy>
<triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
  <maxFileSize>12MB</maxFileSize>
</triggeringPolicy>
<encoder>
  <pattern>%r %marker %p [ %t ] %c - %m%n</pattern>
</encoder>
</appender>
<logger name="com.oxygenxml.positron.connector.openai" level="debug"/>
<logger name="com.oxygenxml.positron.core" level="debug"/>
<root level="error">
  <appender-ref ref="R2"/>
</root>
</configuration>

```

The logging information will be copied in the **Desktop/oxygenLog** folder once the application is started. To avoid performance problems, the `logback.xml` file must be deleted once the logging has been obtained.

Oxygen Emmet Plugin

An **Oxygen Emmet Plugin** is available as an add-on and it provides the means for high-speed coding and editing in **Text** mode via a content assistance mechanism. It can be used for HTML, XSL, CSS, LESS, and other formats. For example, with the Emmet add-on installed, you can type abbreviations (similar to CSS selectors) and expand them into full-fledged HTML code. The add-on contributes a submenu named **Emmet** in the contextual menu and it contains actions for expanding abbreviations or wrapping content with an expanded abbreviation. The two actions can also be invoked using the **Alt + Shift + E (Ctrl + Shift + E on macOS)** or **Alt + Shift + W (Ctrl + Shift + W on macOS)** keyboard shortcuts.

Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** to quickly initiate the installation process:

Install

Manual Installation

To manually install this add-on, follow this procedure:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box. Enter or paste **https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml** in the **Show add-ons from** field or select it from the drop-down menu.

**Note:**

If you have issues connecting to the default update site, you can [download the add-on package](#), unzip it, then use the **Browse for local files** action in the **Install new add-ons** dialog box to locate the downloaded **addon.xml** file.

**Tip:**

For HTML, CSS, LESS, or XSL files, you can simply right-click anywhere in the editor pane and select **Emmet** from the pop-up menu.

2. Select the **Oxygen Emmet Plugin** add-on and click **Next**.
3. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
4. Restart the application.

Result: The Emmet actions will now be available using the keyboard shortcuts or in the **Emmet** submenu (located in the contextual menu of **Text** mode).

Emmet Actions

The two contributed actions are:

Expand abbreviation [**Alt + Shift + E** (**Ctrl + Shift + E** on macOS)]

In **Text** mode, after entering an abbreviation, invoking this action will expand a valid abbreviation into a code snippet, depending on the document type.

In **Author mode**, invoking the action opens a dialog box where you can enter an abbreviation. After you click **OK**, a valid abbreviation is expanded into a code snippet, depending on the document type.

**Tip:**

For HTML, CSS, LESS, or XML-based document types, you can also use **Ctrl + Space** to expand Emmet abbreviations.

Wrap with abbreviation [**Alt + Shift + W** (**Ctrl + Shift + W** on macOS)]

It opens a dialog box where you can enter an abbreviation and after clicking **OK**, the abbreviation is expanded with the selected content added in the last element of the generated snippet.

Abbreviation Expansion Examples

Here are some examples for HTML:

- **Expand abbreviation** example:

```
#page>div.logo+ul#navigation>li*5>a{Item $}
```

expands into:

```
<div id="page">
  <div class="logo"></div>
  <ul id="navigation">
    <li><a href="">Item 1</a></li>
    <li><a href="">Item 2</a></li>
    <li><a href="">Item 3</a></li>
    <li><a href="">Item 4</a></li>
    <li><a href="">Item 5</a></li>
  </ul>
</div>
```

- **Wrap with abbreviation** example:

If the following content is selected to be wrapped:

```
About
News
Products
Contacts
```

then

```
ul>li[title=$#]*>{$#}+img[src=https://www.ex1.com/$#][alt=item$]
```

expands into:

```
<ul>
  <li title="About">About</li>
  <li title="News">News</li>
  <li title="Products">Products</li>
  <li title="Contacts">Contacts</li>
</ul>
```

You can also use Emmet abbreviations for other XML documents. Here are some examples of expanded abbreviations for DITA:

- `prolog>author {AuthorName}`

expands into:

```
<prolog>
  <author>AuthorName</author>
</prolog>
```

- `simpletable>(strow>stentry*4)*4`

expands into a 4x4 simple table.

- `ul>li*3`

expands into an unordered list with 3 list items.

- `ol>li[id="item$"]*3`

expands into:

```
<ol id="ol_gff_bjd_mkb">
  <li id="item1"/>
  <li id="item2"/>
  <li id="item3"/>
</ol>
```

Here are a few CSS examples:

- `@f+`

expands into:

```
@font-face {
  font-family: 'FontName';
  src: url('FileName.eot');
  src: url('FileName.eot?#iefix') format('embedded-opentype'),
       url('FileName.woff') format('woff'),
       url('FileName.ttf') format('truetype'),
       url('FileName.svg#FontName') format('svg');
  font-style: normal;
  font-weight: normal;
}
```

- `-br`

expands into:

```
-webkit-border-right: ;
-moz-border-right: ;
```

```
-ms-border-right: ;
-o-border-right: ;
border-right: ;
```

**Tip:**

To see more examples of Emmet syntax, go to <https://docs.emmet.io/cheat-sheet/>.

Related Information:

[Emmet Syntax Cheat Sheet](#)

[Emmet Documentation](#)

Development

XSD to JSON Schema Converter

Oxygen JSON Editor includes a tool for converting an XML Schema file (XSD) to a JSON Schema file. The **XSD to JSON Schema** action for invoking the tool can be found in the **Tools > JSON Tools** menu. It requires an additional add-on to be installed, so the first time you invoke the action, Oxygen JSON Editor will present a dialog box asking if you want to install it. Once installed, you need to restart Oxygen JSON Editor and the **XSD to JSON Schema** action will invoke the tool.

Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** to quickly initiate the installation process:

Install

Manual Installation

To manually install the add-on, follow these instructions:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box.
2. Enter or paste <https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml> in the **Show add-ons from** field or select it from the drop-down menu.

**Note:**

If you have issues connecting to the default update site, you can [download the add-on package](#), unzip it, then use the **Browse for local files** action in the **Install new add-ons** dialog box to locate the downloaded `addon.xml` file.

3. Select the **XSD to JSON Schema** add-on and click **Next**.

4. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
5. Restart the application.

Result: The **XSD to JSON Schema** dialog box is now available and can be selected from the **Tools > JSON Tools** menu.

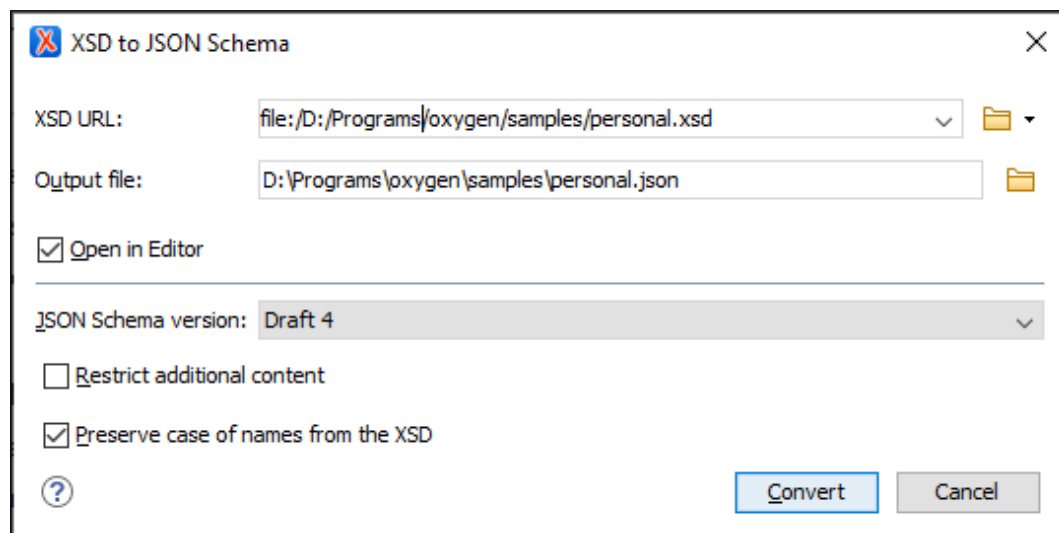
Converting XSD to JSON Schema

To convert an XML Schema (XSD) to a JSON Schema, follow these steps:

1. Select the **XSD to JSON Schema** action from the **Tools > JSON Tools** menu.

Step Result: The **XSD to JSON Schema** dialog box is displayed:

Figure 137. XSD to JSON Schema Dialog Box



2. In the **XSD URL** field, choose or enter the URL of the XML Schema document. The conversion supports XSD versions 1.0 and 1.1.
3. In the **Output file** field, choose the path for the resulting output file.
4. [Optional] You can select the **Open in Editor** option to open the resulting JSON Schema document in the main editing pane.
5. For the **JSON Schema version** option, choose the version of the resulting JSON schema. The possible choices are: **Draft 4**, **Draft 6**, **Draft 7**, **Draft 2019-09**, and **Draft 2020-12**.
6. [Optional] If you select the **Restrict additional content** option, then `additionalProperties` (for objects) and `additionalItems` (for arrays) will be set to `false` in the resulting schema. By default, these keys are not in the schema, meaning that providing additional content (according to the schema) is allowed.
7. [Optional] You can select the **Preserve case of names from the XSD** option if you want the names from the XSD to remain unchanged in the resulting JSON Schema. Otherwise, the default JAXB naming algorithm will be applied (for example, `"some.nAMe"` is changed to `"SomeNAME"`, or `"Some_oth3r_name"` is changed to `"SomeOth3RName"`).
8. Click the **Convert** button.

Result: The original XSD document is now converted to a JSON Schema document. The resulting JSON Schema will be the specified draft and will contain:

- The `$id` of the schema, generated from XSD `targetNamespace`.
- The `$definitions` section, which declares `complex` and `enum` types.
- The `anyOf` section, which lists possible top-level elements as an array of objects.

Other Possible Results:

- If an XSD type extends another type, then its schema is combined with the schema of the base type using the `allOf` keyword.
- If an extension in XSD defines an element with the same name as an attribute in the base, a property named `rest` is generated to avoid name conflicts in JSON.
- If a property of a complex type is a collection property, the schema of the collection items will be wrapped in the JSON array schema.

Conversion Mappings

The following table lists the specific conversion mapping details.

XML Schema Type	JSON Schema Representation
<i>anySimpleType</i>	string, number, integer, boolean, null
<i>anyType</i>	string, number, integer, boolean, null, object, array
<i>string</i>	string
<i>normalizedString</i>	string
<i>token</i>	string
<i>language</i>	string
<i>Name</i>	string
<i>NCName</i>	string
<i>ID</i>	string
<i>IDREF</i>	string
<i>IDREFS</i>	array of strings
<i>ENTITY</i>	string
<i>ENTITIES</i>	array of strings
<i>NMTOKEN</i>	string
<i>NMTOKENS</i>	array of strings
<i>boolean</i>	boolean

XML Schema Type	JSON Schema Representation
<i>base64Binary</i>	array of integers
<i>hexBinary</i>	array of integers
<i>float</i>	number
<i>decimal</i>	number
<i>integer</i>	integer
<i>nonPositiveInteger</i>	integer
<i>negativeInteger</i>	integer
<i>long</i>	integer
<i>int</i>	integer
<i>short</i>	integer
<i>byte</i>	integer
<i>nonNegativeInteger</i>	integer
<i>unsignedLong</i>	integer
<i>unsignedInt</i>	integer
<i>unsignedShort</i>	integer
<i>unsignedByte</i>	integer
<i>positiveInteger</i>	integer
<i>double</i>	number
<i>anyURI</i>	string with "format": "uri"
<i>QName</i>	object with "namespaceURI", "localPart", "prefix"
<i>duration</i>	string
<i>dateTime</i>	string with "format": "date-time"
<i>date</i>	string with "format": "date"
<i>time</i>	string with "format": "time"

Conversion Limitations

In most cases, the conversion creates an equivalent schema, but there are some limitations:

- Restrictions/facets are not taken into consideration when converting (`fractionDigits`, `pattern`, `totalDigits`, `whiteSpace`, `minInclusive`, `maxInclusive`, and the restrictions for length, except `enumeration`). However, extensions and indicators are properly converted (`minOccurs`, `maxOccurs`, `group`, `sequence`, `choice`).
- The `<documentation>` element is not converted into `<description>`.
- The `@substitutionGroup` attribute for an element that has no declared type becomes a reference to the element that can substitute it.
- The `@block` attribute is not taken into consideration during the conversion.

Generating JSON Schema Documentation

Oxygen JSON Editor includes a tool for generating documentation for a JSON Schema file in HTML format. To generate JSON Schema documentation, select **JSON Schema Documentation** from the **Tools > Generate Documentation** menu. You can also open the tool by using the  **Generate Documentation** toolbar button. This opens a dialog box where you can specify the location of the JSON Schema file and HTML output file.

This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen JSON Editor presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen JSON Editor and the **JSON Schema Documentation** action will invoke the tool.

Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** to quickly initiate the installation process:

Install

Manual Installation

To manually install the add-on, follow these instructions:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box.
2. Enter or paste <https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml> in the **Show add-ons from** field or select it from the drop-down menu.



Note:

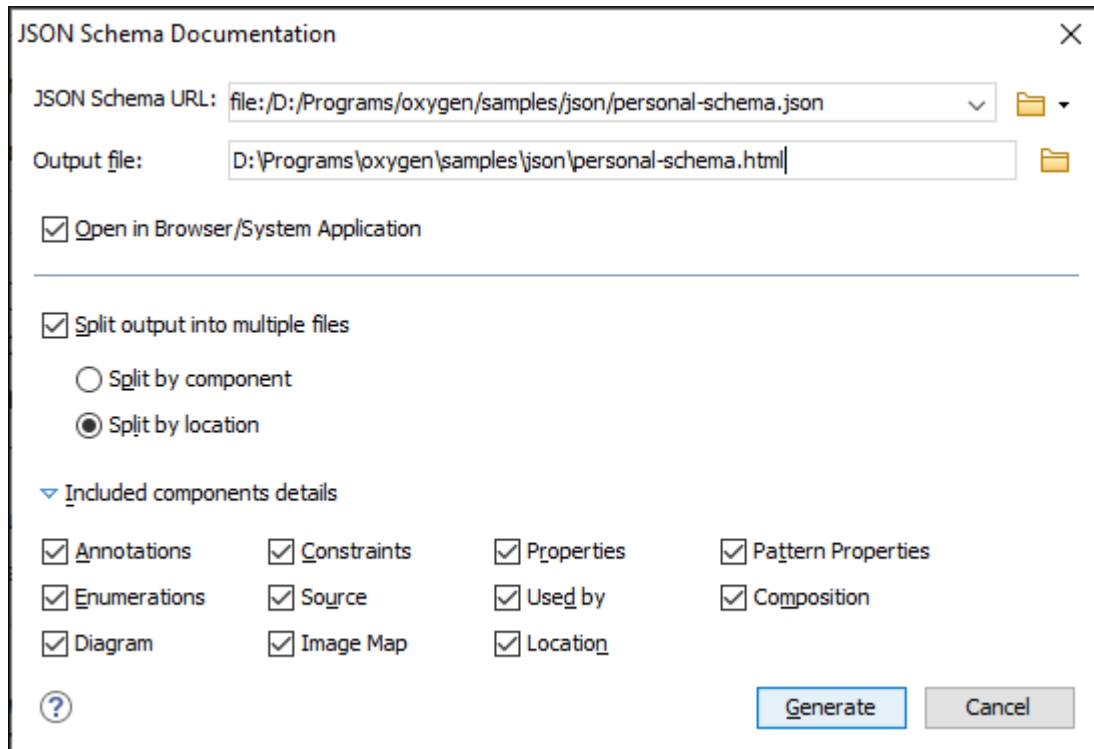
If you have issues connecting to the default update site, you can [download the add-on package](#), unzip it, then use the **Browse for local files** action in the **Install new add-ons** dialog box to locate the downloaded `addon.xml` file.

3. Select the **JSON Schema Documentation** add-on and click **Next**.
4. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
5. Restart the application.

Result: The **JSON Schema Documentation** dialog box is now available and can be selected from the **Tools > Generate Documentation** menu.

JSON Schema Documentation Dialog Box

Figure 138. JSON Schema Documentation Dialog Box



The **JSON Schema Documentation** dialog box includes the following fields and options:

JSON Schema URL

The URL of the JSON Schema file. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the **Browse** drop-down list. The tool supports schemas with versions *Draft 04, 06, 07* and, starting with version 4.0.0 of the add-on, *2019-09* and *2020-12*.

Output file

The path to the folder where the generated HTML file will be saved.

Split output into multiple files

If selected, the application splits the output into multiple files. You can choose between splitting them by **component** name or **location**.

Open in Browser/System Application

If selected, the generated result is opened in the system application associated with the output file type (HTML).

Included component details

This section can be used to specify whether or not the following components are shown in the generated documentation:

- **Annotations** - Displays the annotations (title, description) for each component (property or definition).
- **Constraints** - Displays the schema constraints for each component, according to its type.
- **Properties** - Displays the `properties` of an Object Schema.
- **Pattern Properties** - Displays the `patternProperties` of an Object Schema.
- **Enumerations** - Displays the enumerated values, if specified in the schema.
- **Source** - Displays the text schema source for each component.
- **Used By** - Displays the list of all the components that reference the current definition.
- **Composition** - Displays the `oneOf`, `anyOf`, and `allOf` compositors that are used for combining schemas.
- **Diagram** - Displays the diagram image for each component. The diagrams are generated according to the options specified in the [Schema Design preferences page \(on page 131\)](#). Diagrams are also generated for components within schemas from referenced files.
- **Image Map** - Diagrams will include hyperlinks that navigate to each particular component.
- **Location** - Displays the schema location for each component.

You can click **Generate** at any point to generate the JSON Schema documentation.

Generated JSON Schema Documentation in HTML Format

After generating the JSON Schema documentation, it is presented in a visual diagram style with various sections, hyperlinks, and options.

Figure 139. JSON Schema Documentation Example Opened in a Browser

The screenshot displays the JSON Schema Documentation interface. At the top, there are three tabs: 'Main Schema schema', 'Properties personnel', and 'Definitions'. The 'Definitions' tab is active, showing a list of definitions: 'personType', 'idType', 'nameType', 'nameValueType', and 'linkType'. Below the tabs, the 'Property personnel' section is expanded. It includes a 'Title' (The 'personnel' property), a 'Description' (Defines the personnel as a collection of person entries.), and a 'Diagram' showing a JSON Schema diagram. The diagram illustrates the 'personnel' property as an object with 'type' set to 'object' and 'additionalProperties' set to 'false'. It also shows a 'properties' object containing a 'person' property that references the 'personType' definition. Below the diagram, there is a 'Type' section (object), a 'Properties' table, and a 'Used by' section. The 'Properties' table has two columns: 'Name' and 'Occurrence'. The 'Used by' section shows the schema path '#/schema'. The 'Source' section displays the JSON Schema snippet for the 'personnel' property. The 'Location' section shows the path 'personal-schema.json#/properties/personnel'. On the right side, there is a 'Showing' panel with checkboxes for 'Annotations', 'Diagram', 'Properties', 'Constraints', 'Used By', 'Source', and 'Location', all of which are checked. A 'Close' button is also present.

Property personnel

Annotations

Title The 'personnel' property

Description Defines the personnel as a collection of person entries.

Diagram

The 'personnel' property
Defines the personnel as a collection of person entries.

Type **object**

Properties

Name	Occurrence
person	optional

Additional Properties false

Used by

Schema
#/schema

Source

```

{
  "personnel": {
    "type": "object",
    "additionalProperties": false,
    "title": "The 'personnel' property",
    "description": "Defines the personnel as a collection of person entries.",
    "properties": {
      "person": { "$ref": "#/definitions/personType" }
    }
  }
}

```

Location

personal-schema.json#/properties/personnel

Showing:

- ☒ Annotations
- ☒ Diagram
- ☒ Properties
- ☒ Constraints
- ☒ Used By
- ☒ Source
- ☒ Location

Close

The generated documentation includes a Table of Contents on the left pane with links to particular sections in the right pane. You can collapse or expand details by using the **Showing** options or the **[- Collapse** or **[+ Expand** buttons.

OpenAPI Tester

Oxygen JSON Editor includes a testing tool for *OpenAPI* files. The tool provides the ability to inspect OpenAPI request responses and to ensure that they work as expected. It can be used for *OpenAPI 3.x* in JSON or YAML format.

To use the tool, select **OpenAPI Tester** from the **Tools > JSON Tools** menu. This opens a dialog box where you can specify the location of the OpenAPI file that you want to test.

This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen JSON Editor presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen JSON Editor and the **OpenAPI Tester** action will invoke the tool.

Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** to quickly initiate the installation process:

Install

Manual Installation

To manually install the add-on, follow these instructions:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box.
2. Enter or paste <https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml> in the **Show add-ons from** field or select it from the drop-down menu.



Note:

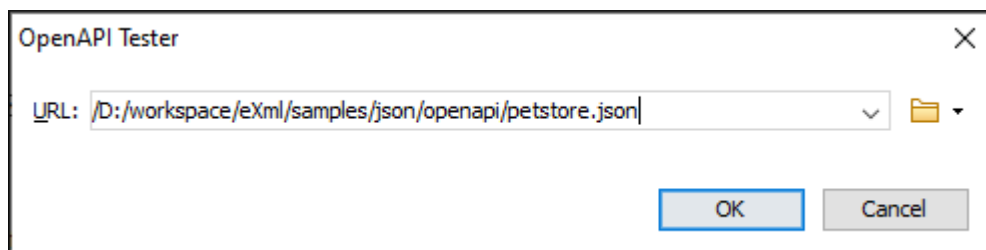
If you have issues connecting to the default update site, you can [download the add-on package](#), unzip it, then use the **Browse for local files** action in the **Install new add-ons** dialog box to locate the downloaded **addon.xml** file.

3. Select the **OpenAPI Tester** add-on and click **Next**.
4. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
5. Restart the application.

Result: The **OpenAPI Tester** dialog box is now available and can be selected from the **Tools > JSON Tools** menu.

OpenAPI Tester Dialog Box

Figure 140. OpenAPI Tester Dialog Box



After selecting **OpenAPI Tester** from the **Tools > JSON Tools** menu, the **OpenAPI Tester** dialog box is displayed where you can select the URL for the OpenAPI document (either local or remote). After clicking **OK**, the **OpenAPI Tester** view becomes visible on the right side of the editor. The view can also be opened by selecting **OpenAPI Tester** from the **Window > Show View** menu.

Figure 141. OpenAPI Tester View

The tester loads the selected OpenAPI document and fills in the corresponding fields. The tester fields are as follows:

URL

The URL of the OpenAPI file. You can specify the path by using the text field or the **Browse** button. If you specify the path directly in the text field, you need to click the **Reload** button. You can use the **Clear** button if you want to remove all the content from the view.

Server

The list of servers defined by the OpenAPI document. The global "servers" array can be overridden on the path level or operation level. If it is not provided or is empty, the server URL defaults to "/".

Path

The list of individual endpoints (paths) in the API. The full request URL is constructed as `<server-url>/path`.

Operation

The list of HTTP operations supported by the selected path. OpenAPI 3.0 supports get, post, put, patch, delete, head, options, and trace.

Parameters tab

The definition of the parameters for the selected operation. OpenAPI 3.0 supports parameters passed via path, query string, headers, and cookies. The required parameters will be marked with a red asterisk. For array parameters, items must be separated by a comma.

Authorization tab

The list of available authentication types. The authentication data is persistent and can be removed from the contextual menu.

Request Body tab

The media type and the body of the request (if the selected operation allows it). For example, GET will not allow specifying a body since that HTTP method disallows it. Oxygen JSON Editor will create a sample request body based on the JSON Schemas defined in the OpenAPI document. Usually, you just need to change a few values for the request to be valid.

Variables tab

The list of variables used for server templating (if applicable). Variables are indicated by {curly brackets} in the server URL.

Send button

Executes the request by creating a HTTP client with all the information extracted from the view.

Open response in editor

If selected, the response will be opened in the main editing pane.



Generate scenario test

Creates a test scenario file in JSON format, based on the information extracted from the view. The file is then opened in the main editing pane, and can be used to [Run OpenAPI Test Scenario](#). (on page 570)

Response area

Initially this area is empty. After using the **Send** button, it presents the message received from the server in response to the request. The expected content type of the message is JSON. The response status and possible errors that may occur are presented right below this area. The status has a green foreground for successful requests and a red foreground otherwise.

Resources

For more information about OpenAPI editing, testing, and documenting, watch our webinar:

<https://www.youtube.com/embed/gKdabeh49Qk>

Generating OpenAPI Documentation

Oxygen JSON Editor includes a tool for generating documentation for *OpenAPI* 3.0, or 3.1 documents in either JSON or YAML format, including annotations and cross references. The documentation displays information about the servers, paths, components and tags defined in the OpenAPI documents and you can choose whether the output is presented in HTML (with various sections, hyperlinks, and filtering options), DITA, or PDF.

Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** to quickly initiate the installation process:

Install

Manual Installation

To manually install the add-on, follow these instructions:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box.
2. Enter or paste <https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml> in the **Show add-ons from** field or select it from the drop-down menu.



Note:

If you have issues connecting to the default update site, you can [download the add-on package](#), unzip it, then use the **Browse for local files** action in the **Install new add-ons** dialog box to locate the downloaded `addon.xml` file.

3. Select the **OpenAPI Documentation Generator** add-on and click **Next**.
4. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
5. Restart the application.

Result: The **OpenAPI Documentation** dialog box is now available and can be selected from the **Tools > Generate Documentation** menu.

Generating OpenAPI Documentation

To generate OpenAPI documentation, select **OpenAPI Documentation** from the **Tools > Generate Documentation** menu. This opens a dialog box where you can specify the location of the OpenAPI file and the output file, as well as the type of output to generate.

This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen JSON Editor presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen JSON Editor and the **OpenAPI Documentation** action will invoke the tool.

Figure 142. OpenAPI Documentation Dialog Box

The **OpenAPI Documentation** dialog box includes the following fields and options:

OpenAPI URL

The URL of the OpenAPI file (it can be in either JSON or YAML format). You can specify the path by using the text field or the browsing button (📁).

Output file

The path to the folder where the generated file(s) will be saved.

Generate HTML output

Choose this option to generate the output in HTML format.

Split output into multiple files

If selected, the application splits the output into multiple files. You can choose between splitting them by **component** name or **location**.

Included JSON schema component details

This section can be used to specify whether or not details about the following components that belong to internal or imported schemas are shown in the generated documentation:

- **Annotations** - Displays the annotations (title, description) for each component (property or definition).
- **Constraints** - Displays the schema constraints for each component, according to its type.
- **Source** - Displays the text schema source for each component.
- **Location** - Displays the schema location for each component.
- **Used By** - Displays the list of all the components that reference the current definition.
- **Properties** - Displays the `properties` of an Object Schema.
- **Pattern Properties** - Displays the `patternProperties` of an Object Schema.
- **Enumerations** - Displays the enumerated values, if specified in the schema.
- **Diagram** - Displays the diagram image for each component. The diagrams are generated according to the options specified in the [Schema Design preferences page \(on page 131\)](#). Diagrams are also generated for components within schemas from referenced files.
- **Composition** - Displays the `oneOf`, `anyOf`, and `allOf` compositors that are used for combining schemas.

Generate DITA output

Choose this option to generate the output in DITA format.

Create DITA map from OpenAPI document

If selected, the application generates a DITA map with referenced topics based on the input OpenAPI document.

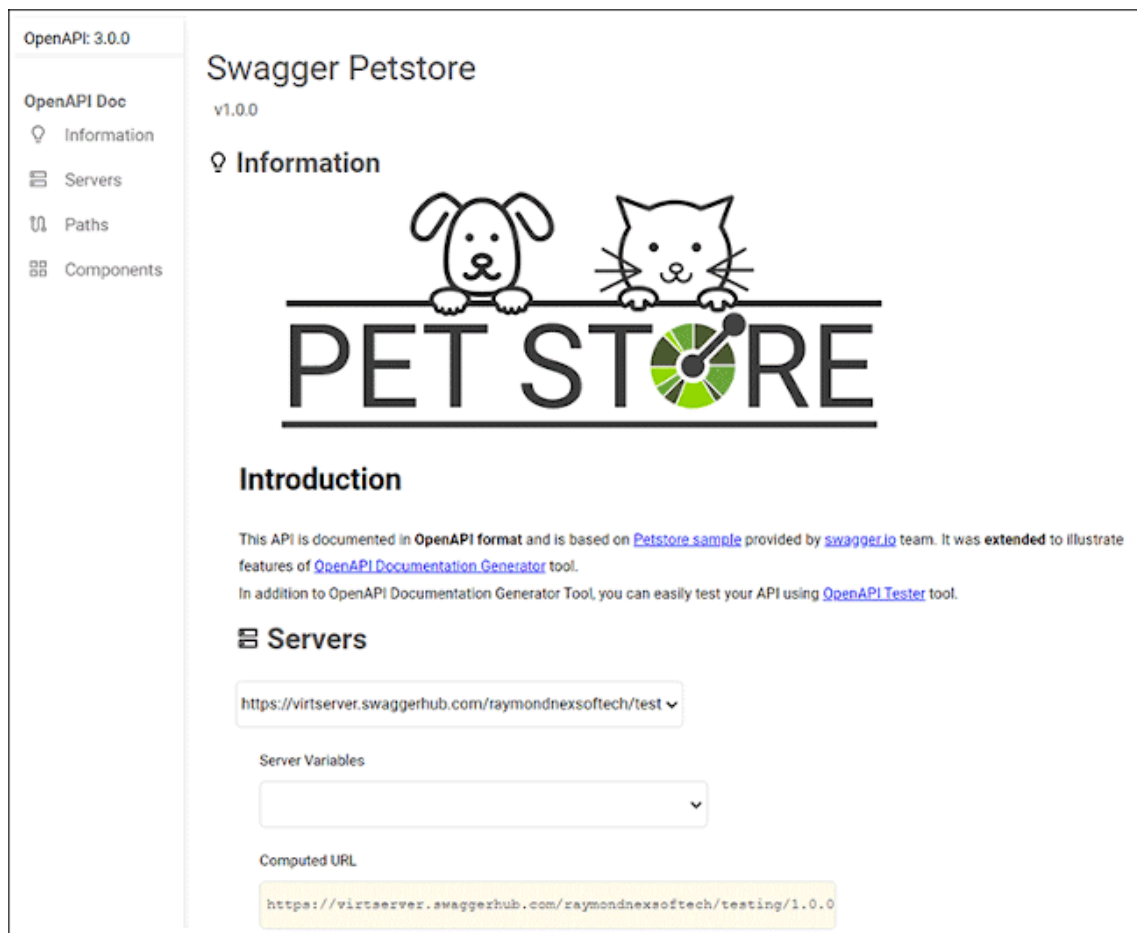
Generate PDF output

Choose this option to generate the output in PDF format.

Click the **Generate** button to generate the OpenAPI documentation.

Generated OpenAPI Documentation in HTML Format

When generating the OpenAPI documentation in HTML format, it is presented in the browser with various sections, hyperlinks, and options.

Figure 143. OpenAPI Documentation Example

The generated documentation includes a Table of Contents on the left pane with links to particular sections in the right pane. You can collapse or expand details by using the **Collapse** or **Expand** buttons.

Resources

For more information about OpenAPI editing, testing, and documenting, watch the following webinar:

<https://www.youtube.com/embed/gKdabeh49Qk>

11.

Tools

Oxygen JSON Editor includes a variety of helpful tools to help you accomplish tasks. This section presents many of those tools. These tools are available in the **Tools** menu and some of them can be launched through keyboard shortcuts or command-line scripts.

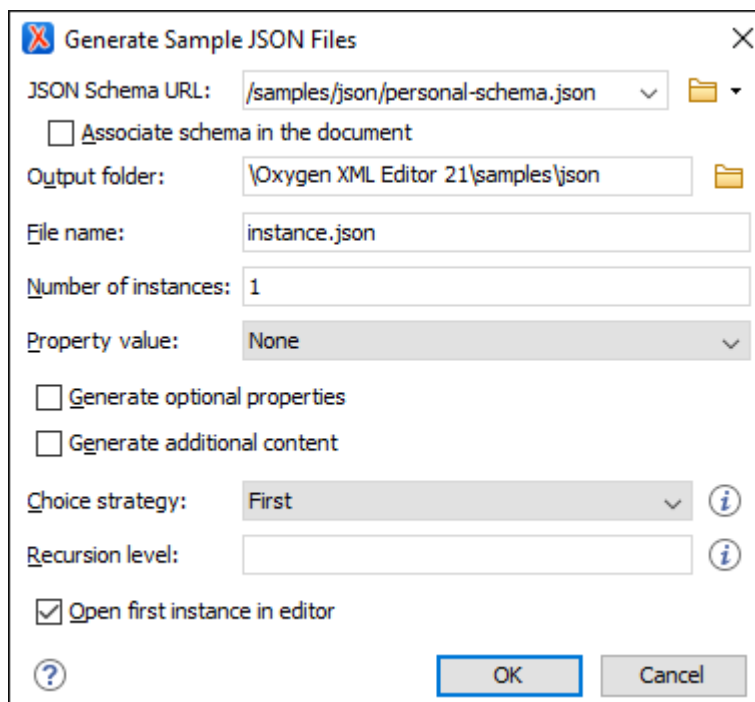
JSON Tools

Oxygen JSON Editor includes some useful tools for converting between JSON, XML and YAML, converting XSD to JSON Schema, generating JSON instances or a JSON Schema, and OpenAPI (JSON and YAML) testing.

Generating Sample JSON Files from a JSON Schema

Oxygen JSON Editor includes a tool for generating sample JSON files. To generate sample JSON files from a JSON Schema, select **Generate Sample JSON Files** from the **Tools > JSON Tools** menu. The action opens a dialog box where you can configure a variety of options for generating the files.

Figure 144. Generate Sample JSON Files Dialog Box



The dialog box titled "Generate Sample JSON Files" includes the following fields and options:

- JSON Schema URL:** A text field containing "/samples/json/personal-schema.json" with a folder icon to its right.
- ☐ **Associate schema in the document**
- Output folder:** A text field containing "\\Oxygen XML Editor 21\\samples\\json" with a folder icon to its right.
- File name:** A text field containing "instance.json".
- Number of instances:** A text field containing "1".
- Property value:** A dropdown menu currently set to "None".
- ☐ **Generate optional properties**
- ☐ **Generate additional content**
- Choice strategy:** A dropdown menu currently set to "First" with an information icon to its right.
- Recursion level:** A text field with an information icon to its right.
- ☒ **Open first instance in editor**
- Buttons: "OK" (highlighted with a blue border) and "Cancel".
- A help icon (?) is located in the bottom left corner.

The **Generate Sample JSON Files** dialog box includes the following fields and options:

Schema URL

The URL of the Schema location. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the  ▾ **Browse** drop-down list. The tool supports schemas with versions *Draft 04*, *06*, *07*, *2019-09*, and *2020-12*.

Associate schema in the document

If enabled, the specified schema will be associated with the generated files.

Output folder

Path to the folder where the generated JSON instances will be saved.

File name

The name of the instance(s) that will be generated. By default, `instance.json` is used.

Number of instances

The desired number of JSON instances to be generated. When more than one instance is generated, the index of the instance will be added to its file name.

Property value

You can specify the way the values of the properties are generated. The following options are available:

- *None* - Assigns empty values for properties (a template file will be generated). This is the default value.
- *Default* - Assigns the name of the property as the value (for strings) or assigns the specified minimum value (for numbers).
- *Random* - Assigns random values according to schema restrictions.

Generate optional properties

If selected, the JSON instance will be generated with optional properties that are defined in the JSON schema. Otherwise, only the required properties will be generated.

Generate additional content

If selected, the JSON instance will be generated with additional properties that are defined in the JSON schema as `additionalProperties` and additional items that are defined as `additionalItems` (in the case of an Array).

Choice strategy

You can specify the way an instance will be generated from a schema that contains a `CombinedSchema` (with either *oneOf* or *anyOf*). The following options are available:

- *First* - The first defined schema in *oneOf* or *anyOf* will be used.
- *Random* - A random schema defined in *oneOf* or *anyOf* will be used.

Recursion level

This option controls the maximum allowed depth (must be a number), in case the selected schema contains recursive calls of `$ref` schemas referencing one another. By default, it is set to 1, meaning that the generation for the recursive calls will stop after the first iteration.

Open first instance in editor

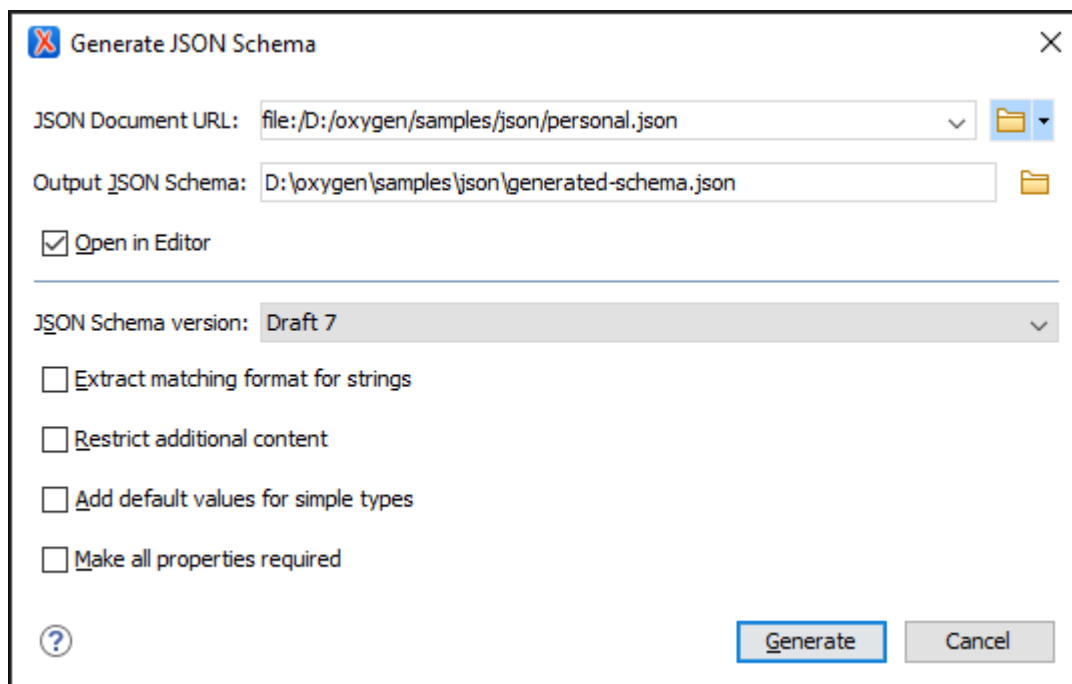
If selected, the first generated instance is opened in the editor.

You can click **OK** at any point to generate the sample JSON files.

Generating JSON Schema from a JSON File

Oxygen JSON Editor includes a tool for generating a sample JSON Schema from a JSON file. To generate a sample JSON Schema, select **Generate JSON Schema** from the **Tools > JSON Tools** menu. The action opens a dialog box where you can configure some options for generating the JSON Schema.

Figure 145. Generate JSON Schema Dialog Box



The **Generate JSON Schema** dialog box includes the following fields and options:

JSON Document URL

The URL of the JSON file. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the ▾ **Browse** drop-down list.

Output JSON Schema

The path to the folder where the generated JSON Schema will be saved.

Open in Editor

If selected, the generated JSON Schema is opened in the editor.

JSON Schema version

The version of the resulting JSON schema. The possible choices are: **Draft 4**, **Draft 6**, **Draft 7**, **Draft 2019-09**, and **Draft 2020-12**.

Extract matching format for strings

If selected, the generator will attempt to find a format that matches the string values from the JSON Document.

Restrict additional content

If selected, *additionalProperties* (for objects) and *additionalItems* (for arrays) will be set to *false* in the resulting schema. By default, these keys are not in the schema, meaning that providing additional content (according to the schema) is allowed.

Add default values for simple types

If selected, the *default* values (*0* for number, *""* for string, *false* for boolean) and *examples* for strings will be added.

Make all properties required

If selected, the generator will mark all the properties as required in the resulting schema.

You can click **Generate** at any point to generate the JSON Schema.

JSON to YAML Converter

Converting JSON to YAML in Oxygen

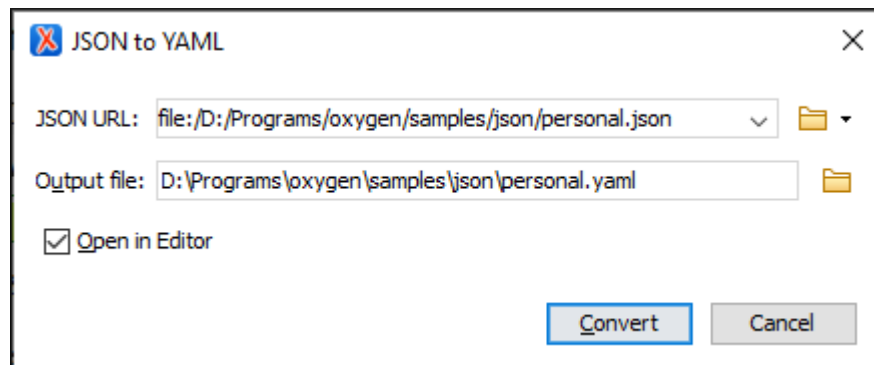
Oxygen JSON Editor includes a useful and simple tool for converting JSON files to YAML. The **JSON to YAML** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

To convert a JSON document to YAML, follow these steps:

1. Select the **JSON to YAML** action from the **Tools > JSON Tools** menu.

The **JSON to YAML** dialog box is displayed:

Figure 146. JSON to YAML Dialog Box



2. Choose or enter the **JSON URL** for the document you want to convert.
3. Choose the path of the **Output file** that will contain the resulting YAML document.

4. **[Optional]** Select the **Open in Editor** option to open the resulting YAML document in the main editing pane.
5. Click the **Convert** button.

Result: The original JSON document is now converted to a YAML document.

Related Information:

[YAML to JSON Converter \(on page 387\)](#)

YAML to JSON Converter

Converting YAML to JSON in Oxygen

Oxygen JSON Editor includes a useful and simple tool for converting YAML files to JSON. It even works on files that consist of multiple YAML documents, each separated by three dashes (---), in which case the conversion creates multiple JSON files with a number in the name.

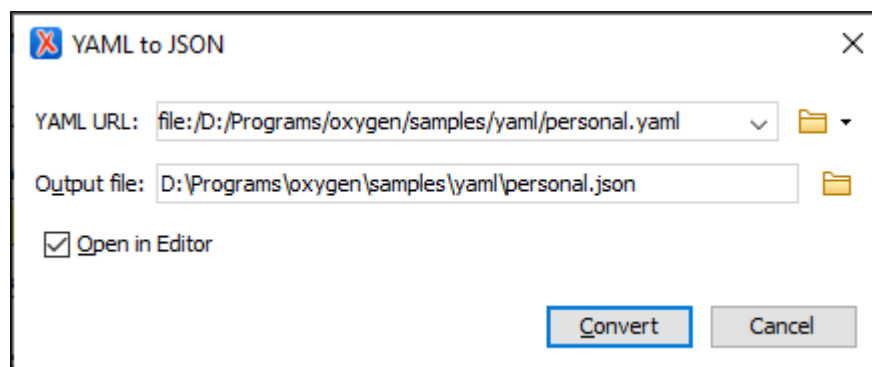
The **YAML to JSON** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

To convert a YAML document to JSON, follow these steps:

1. Select the **YAML to JSON** action from the **Tools > JSON Tools** menu.

The **YAML to JSON** dialog box is displayed:

Figure 147. YAML to JSON Dialog Box



2. Choose or enter the **YAML URL** for the document you want to convert.
3. Choose the path of the **Output file** that will contain the resulting JSON document.
4. **[Optional]** Select the **Open in Editor** option to open the resulting JSON document in the main editing pane.
5. Click the **Convert** button.

Result: The original YAML document is now converted to a JSON document.

Related Information:

[JSON to YAML Converter \(on page 387\)](#)

JSON to XML Converter

Online JSON to XML Converter

**Attention:**

For a simple **ONLINE** tool for converting a single JSON file to XML, or vice versa, go to: https://www.oxygenxml.com/xml_json_converter.html.

Converting JSON to XML in Oxygen

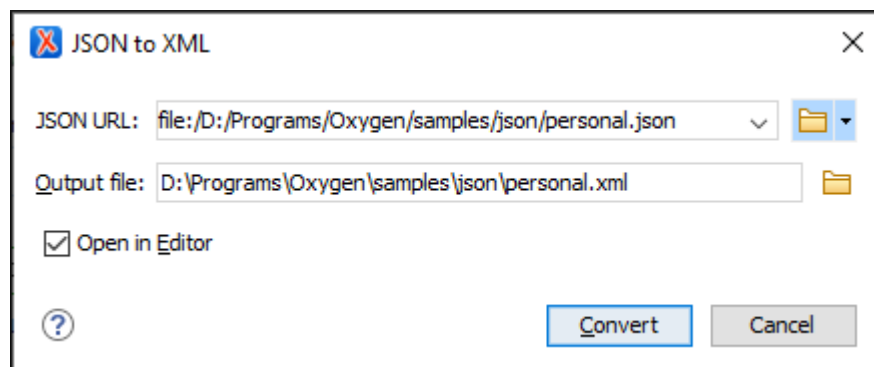
Oxygen JSON Editor includes a useful and simple tool for converting JSON files to XML. The **JSON to XML** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

To convert a JSON document to XML, follow these steps:

1. Select the **JSON to XML** action from the **Tools > JSON Tools** menu.

The **JSON to XML** dialog box is displayed:

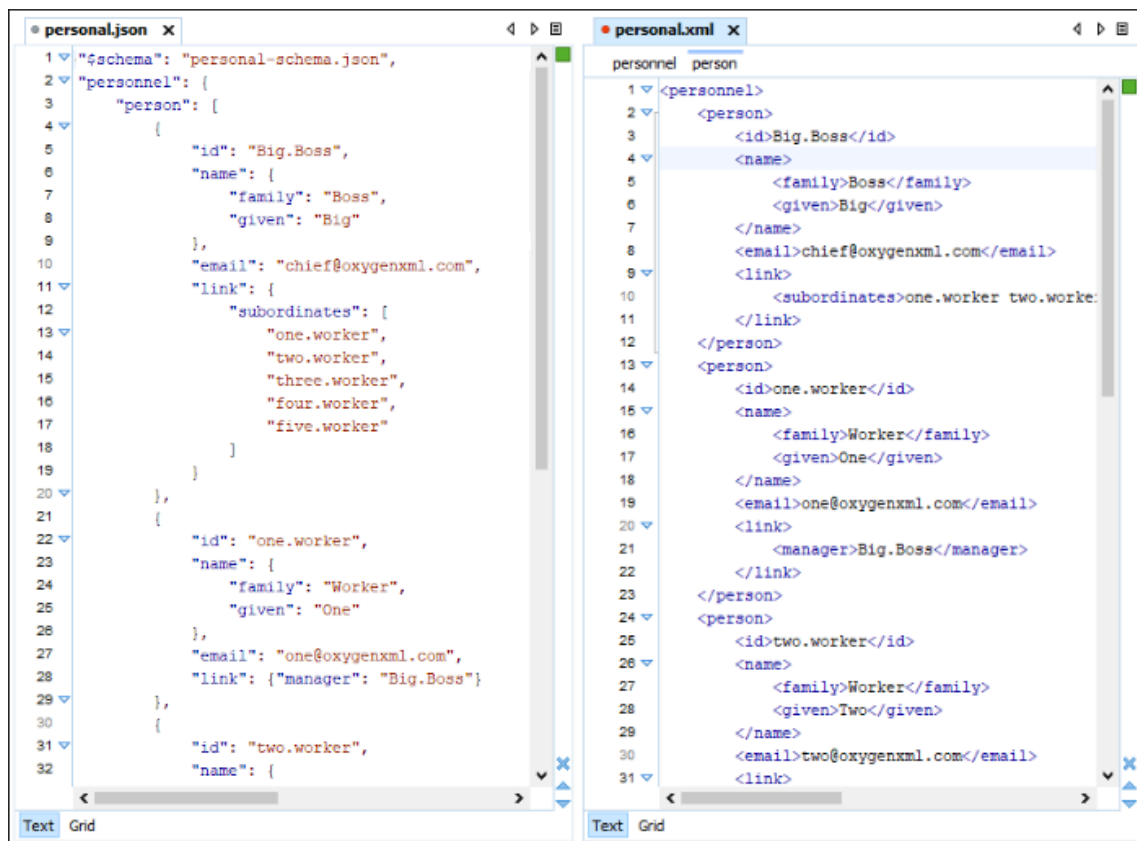
Figure 148. JSON to XML Dialog Box



2. Choose or enter the **Input URL** of the JSON document.
3. Choose the path of the **Output file** that will contain the resulting XML document.
4. Select the **Open in Editor** option to open the resulting XML document in the main editing pane.
5. Click the **Convert** button.

Result: The original JSON document is now converted to an XML document.

Figure 149. Example: XML to JSON Operation Result



Conversion Details

- If the JSON document has more than one property on the first level, the converted XML document will have an additional root element called `<JSON>`.

For example, the following JSON document:

```
{
  "personnel": {
    "person": [
      {"name": "Boss"},
      {"name": "Worker"}
    ]
  },
  "id": "personnel-id"
}
```

it is converted to:

```
<?xml version="1.0" encoding="UTF-8"?>
<JSON>
  <personnel>
    <person>
      <name>Boss</name>
```

```

    </person>
  <person>
    <name>Worker</name>
  </person>
</personnel>
<id>personnel-id</id>
</JSON>

```

- If the JSON document is an array, the converted XML document will have a root element called `<array>` and for each item within the array, another `<array>` is created.

```

[
  { "name": "Boss" },
  { "name": "Worker" }
]

```

it is converted to:

```

<?xml version="1.0" encoding="UTF-8"?>
<array>
  <array>
    <name>Boss</name>
  </array>
  <array>
    <name>Worker</name>
  </array>
</array>

```

- If the name of a JSON property contains characters that are not valid in XML element names (for example, \$), then the invalid characters will be escaped as its hexadecimal equivalent in the converted XML.

```

{"$id": "personnel-id"}

```

is converted to:

```

<_X24_id>personnel-id</_X24_id>

```

Related Information:

[XML to JSON Converter \(on page 384\)](#)

XML to JSON Converter

Online XML to JSON Converter



Attention:

For a simple **ONLINE** tool for converting a single XML file to JSON, or vice versa, go to: https://www.oxygenxml.com/xml_json_converter.html.

Converting XML to JSON in Oxygen

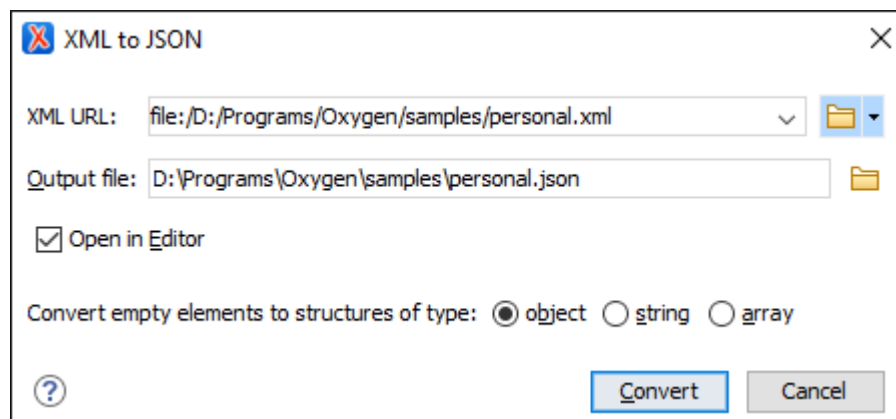
Oxygen JSON Editor includes a useful and simple tool for converting XML files to JSON. The **XML to JSON** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

To convert an XML document to JSON, follow these steps:

1. Select the **XML to JSON** action from the **Tools > JSON Tools** menu.

Step Result: The **XML to JSON** dialog box is displayed:

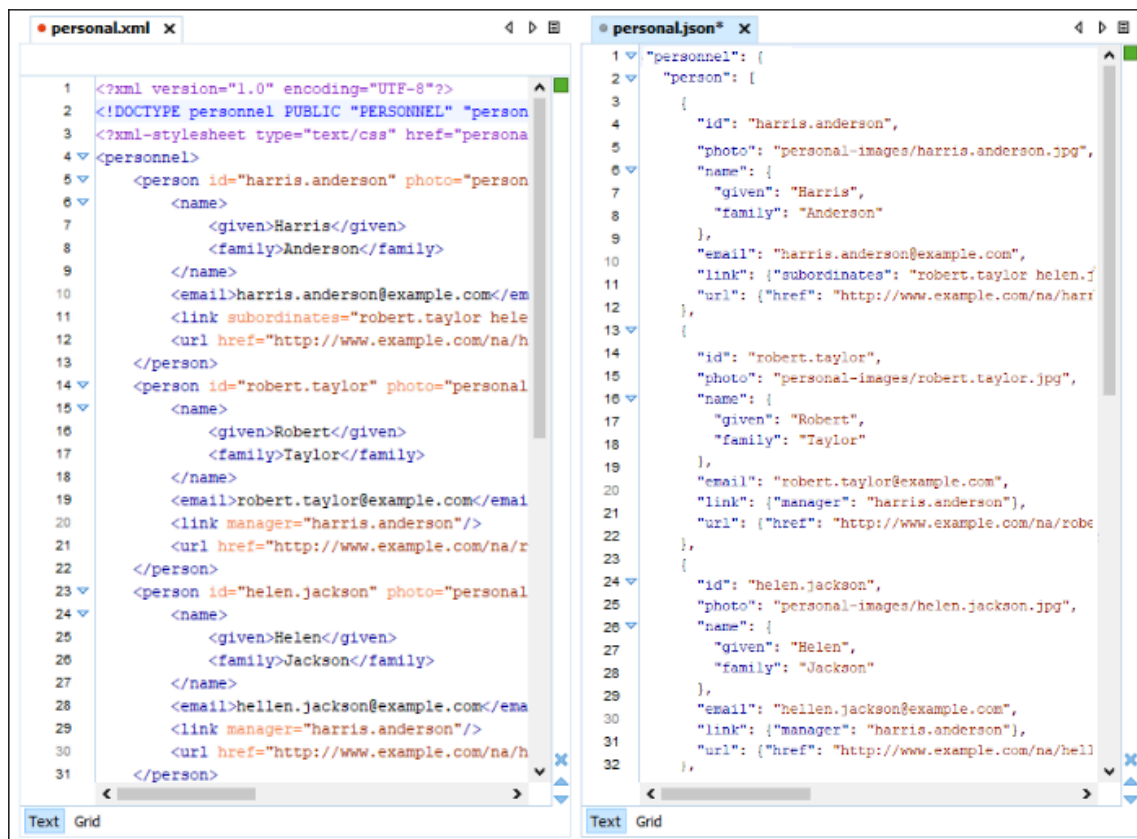
Figure 150. XML to JSON Dialog Box



2. Choose or enter the **Input URL** of the XML document.
3. Choose the path of the **Output file** that will contain the resulting JSON document.
4. Select how you want empty elements to be converted (default is **object**).
5. Select the **Open in Editor** option to open the resulting JSON document in the main editing pane.
6. Click the **Convert** button.

Result: The original XML document is now converted to a JSON document.

Figure 151. Example: XML to JSON Operation Result



Conversion Details

- Some XML components are ignored (e.g. comments and processing instructions).
- If any elements contain attributes in the XML document, the attributes are converted to properties in the converted JSON document. If the XML document contains more than one element with the same name, they will be converted into an array of object in the converted JSON document.

For example, the following XML document:

```
<personnel>
  <person id="person.one">
    <name>Boss</name>
  </person>
  <person id="person.two">
    <name>Worker</name>
  </person>
</personnel>
```

it is converted to:

```
{
  "personnel": {
    "person": [
      {
```

```

    "id": "person.one",
    "name": "Boss"
  },
  {
    "id": "person.two",
    "name": "Worker"
  }
]
}
}

```

- If the XML document contains elements with mixed content (text plus elements), the converted JSON document will contain a `#text` property with its value set as the text content. If there are multiple text nodes, the subsequent `#text` properties will contain a number (e.g. `#text1`, `#text2`). If there are multiple elements with the same name, the first property will have the element name and the subsequent properties will contain a number (e.g. `b`, `b#1`, `b#2`).

```
<p>This <b>is</b> an <b>example</b>!</p>
```

is converted to:

```

{
  "p": {
    "#text": "This ",
    "b": "is",
    "#text1": " an ",
    "b#1": "example",
    "#text2": "!"
  }
}

```

- If the XML document contains element names that contains hexadecimal codes (for example, if they were escaped during a [JSON to XML conversion \(on page 381\)](#)), it will be converted to the normal character value in the converted JSON document.

```
<_X24_id>personnel-id</_X24_id>
```

is converted to:

```
{"$id": "personnel-id"}
```

Related Information:

[JSON to XML Converter \(on page 381\)](#)

XSD to JSON Schema Converter

Oxygen JSON Editor includes a tool for converting an XML Schema file (XSD) to a JSON Schema file. The **XSD to JSON Schema** action for invoking the tool can be found in the **Tools > JSON Tools** menu. It requires an additional add-on to be installed, so the first time you invoke the action, Oxygen JSON Editor will present a dialog box asking if you want to install it. Once installed, you need to restart Oxygen JSON Editor and the **XSD to JSON Schema** action will invoke the tool.

Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** to quickly initiate the installation process:

A rectangular button with a thin border and the text "Install" centered inside.

Manual Installation

To manually install the add-on, follow these instructions:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box.
2. Enter or paste <https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml> in the **Show add-ons from** field or select it from the drop-down menu.



Note:

If you have issues connecting to the default update site, you can [download the add-on package](#), unzip it, then use the **Browse for local files** action in the **Install new add-ons** dialog box to locate the downloaded **addon.xml** file.

3. Select the **XSD to JSON Schema** add-on and click **Next**.
4. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
5. Restart the application.

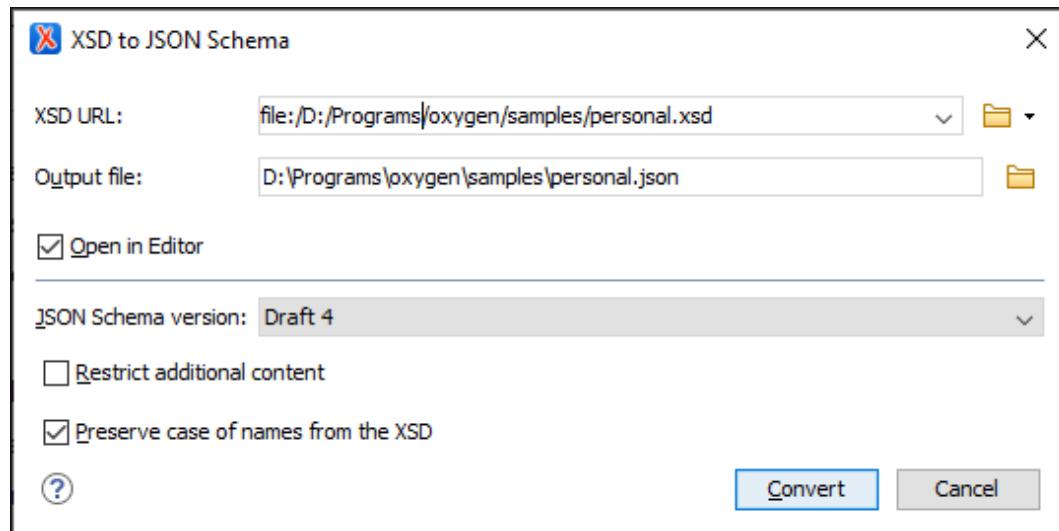
Result: The **XSD to JSON Schema** dialog box is now available and can be selected from the **Tools > JSON Tools** menu.

Converting XSD to JSON Schema

To convert an XML Schema (XSD) to a JSON Schema, follow these steps:

1. Select the **XSD to JSON Schema** action from the **Tools > JSON Tools** menu.

Step Result: The **XSD to JSON Schema** dialog box is displayed:

Figure 152. XSD to JSON Schema Dialog Box

2. In the **XSD URL** field, choose or enter the URL of the XML Schema document. The conversion supports XSD versions 1.0 and 1.1.
3. In the **Output file** field, choose the path for the resulting output file.
4. [Optional] You can select the **Open in Editor** option to open the resulting JSON Schema document in the main editing pane.
5. For the **JSON Schema version** option, choose the version of the resulting JSON schema. The possible choices are: **Draft 4**, **Draft 6**, **Draft 7**, **Draft 2019-09**, and **Draft 2020-12**.
6. [Optional] If you select the **Restrict additional content** option, then `additionalProperties` (for objects) and `additionalItems` (for arrays) will be set to `false` in the resulting schema. By default, these keys are not in the schema, meaning that providing additional content (according to the schema) is allowed.
7. [Optional] You can select the **Preserve case of names from the XSD** option if you want the names from the XSD to remain unchanged in the resulting JSON Schema. Otherwise, the default JAXB naming algorithm will be applied (for example, `"some.nAMe"` is changed to `"SomeNAME"`, or `"Some_oth3r_name"` is changed to `"SomeOth3RName"`).
8. Click the **Convert** button.

Result: The original XSD document is now converted to a JSON Schema document. The resulting JSON Schema will be the specified draft and will contain:

- The `$id` of the schema, generated from XSD `targetNamespace`.
- The `$definitions` section, which declares `complex` and `enum` types.
- The `anyOf` section, which lists possible top-level elements as an array of objects.

Other Possible Results:

- If an XSD type extends another type, then its schema is combined with the schema of the base type using the `allOf` keyword.
- If an extension in XSD defines an element with the same name as an attribute in the base, a property named `rest` is generated to avoid name conflicts in JSON.
- If a property of a complex type is a collection property, the schema of the collection items will be wrapped in the JSON array schema.

Conversion Mappings

The following table lists the specific conversion mapping details.

XML Schema Type	JSON Schema Representation
<i>anySimpleType</i>	string, number, integer, boolean, null
<i>anyType</i>	string, number, integer, boolean, null, object, array
<i>string</i>	string
<i>normalizedString</i>	string
<i>token</i>	string
<i>language</i>	string
<i>Name</i>	string
<i>NCName</i>	string
<i>ID</i>	string
<i>IDREF</i>	string
<i>IDREFS</i>	array of strings
<i>ENTITY</i>	string
<i>ENTITIES</i>	array of strings
<i>NMTOKEN</i>	string
<i>NMTOKENS</i>	array of strings
<i>boolean</i>	boolean
<i>base64Binary</i>	array of integers
<i>hexBinary</i>	array of integers
<i>float</i>	number
<i>decimal</i>	number
<i>integer</i>	integer
<i>nonPositiveInteger</i>	integer

XML Schema Type	JSON Schema Representation
<i>negativeInteger</i>	integer
<i>long</i>	integer
<i>int</i>	integer
<i>short</i>	integer
<i>byte</i>	integer
<i>nonNegativeInteger</i>	integer
<i>unsignedLong</i>	integer
<i>unsignedInt</i>	integer
<i>unsignedShort</i>	integer
<i>unsignedByte</i>	integer
<i>positiveInteger</i>	integer
<i>double</i>	number
<i>anyURI</i>	string with "format": "uri"
<i>QName</i>	object with "namespaceURI", "localPart", "prefix"
<i>duration</i>	string
<i>dateTime</i>	string with "format": "date-time"
<i>date</i>	string with "format": "date"
<i>time</i>	string with "format": "time"

Conversion Limitations

In most cases, the conversion creates an equivalent schema, but there are some limitations:

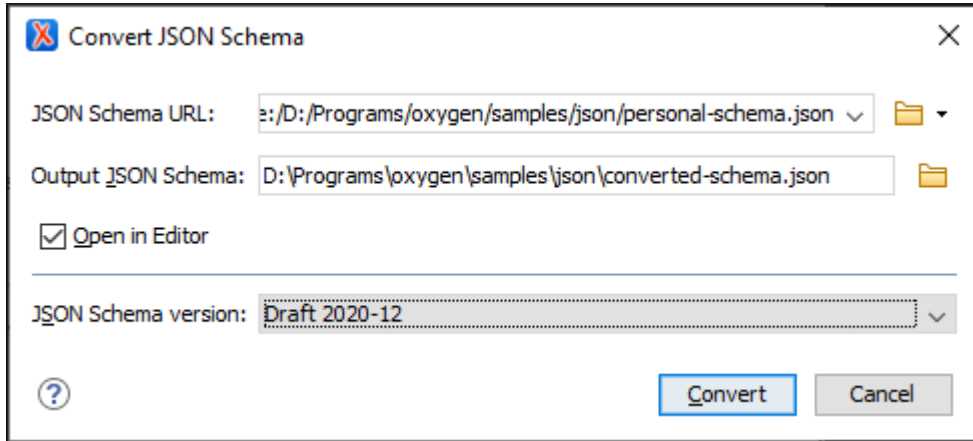
- Restrictions/facets are not taken into consideration when converting (`fractionDigits`, `pattern`, `totalDigits`, `whiteSpace`, `minInclusive`, `maxInclusive`, and the restrictions for length, except `enumeration`). However, extensions and indicators are properly converted (`minOccurs`, `maxOccurs`, `group`, `sequence`, `choice`).
- The `<documentation>` element is not converted into `<description>`.
- The `@substitutionGroup` attribute for an element that has no declared type becomes a reference to the element that can substitute it.
- The `@block` attribute is not taken into consideration during the conversion.

JSON Schema Converter

Oxygen JSON Editor includes a tool for converting an older version of a JSON schema (Draft 4, 6, or 7) to the latest versions (Draft 2019-09 or Draft 2020-12).

To convert a JSON schema, select **Convert JSON Schema** from the **Tools > JSON Tools** menu. The action opens a dialog box where you can configure some options for converting the JSON Schema.

Figure 153. Convert JSON Schema Dialog Box



The **Convert JSON Schema** dialog box includes the following fields and options:

JSON Schema URL

The URL of the JSON schema file. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the **Browse** drop-down list.

Output JSON Schema

The path to the folder where the converted JSON schema will be saved.

Open in Editor

If selected, the converted JSON schema is opened in the editor.

JSON Schema version

The version of the resulting JSON schema. The possible choices are: **Draft 2019-09** or **Draft 2020-12**.

You can click **Convert** at any point to generate the JSON Schema.

Conversion Notes

- The `$schema` declaration is changed according to the selected JSON schema version.
- The `definitions` keyword is converted to `$defs` and all the references are updated.
- The `dependencies` keyword is split into `dependentRequired` and `dependentSchemas`.
- The `items` keyword (tuple array) is converted to `prefixItems` (Draft 2020-12).
- The `additionalItems` keyword is converted to `items` (Draft 2020-12, only if `prefixItems` is present).
- The `exclusiveMinimum` and `exclusiveMaximum` keywords with boolean values (Draft 4) are removed.

- The `id` keyword (Draft 4) is converted to `$id`.
- The `$ref` keyword wrapped into 1-item `allof` is unwrapped because the latest versions allow processing `$ref` along with other keywords.

OpenAPI Tester

Oxygen JSON Editor includes a testing tool for *OpenAPI* files. The tool provides the ability to inspect OpenAPI request responses and to ensure that they work as expected. It can be used for [OpenAPI 3.x](#) in JSON or YAML format.

To use the tool, select **OpenAPI Tester** from the **Tools > JSON Tools** menu. This opens a dialog box where you can specify the location of the OpenAPI file that you want to test.

This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen JSON Editor presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen JSON Editor and the **OpenAPI Tester** action will invoke the tool.

Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** to quickly initiate the installation process:

Install

Manual Installation

To manually install the add-on, follow these instructions:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box.
2. Enter or paste <https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml> in the **Show add-ons from** field or select it from the drop-down menu.



Note:

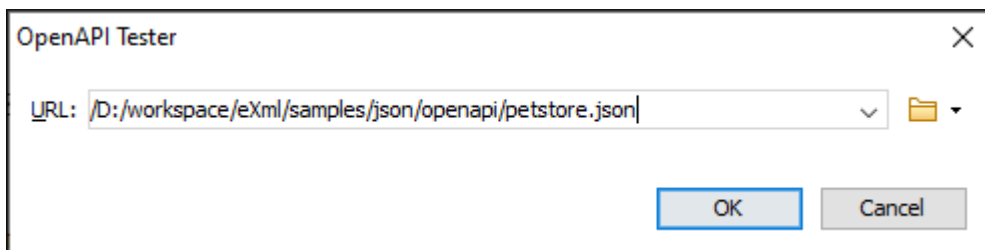
If you have issues connecting to the default update site, you can [download the add-on package](#), unzip it, then use the **Browse for local files** action in the **Install new add-ons** dialog box to locate the downloaded `addon.xml` file.

3. Select the **OpenAPI Tester** add-on and click **Next**.
4. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
5. Restart the application.

Result: The **OpenAPI Tester** dialog box is now available and can be selected from the **Tools > JSON Tools** menu.

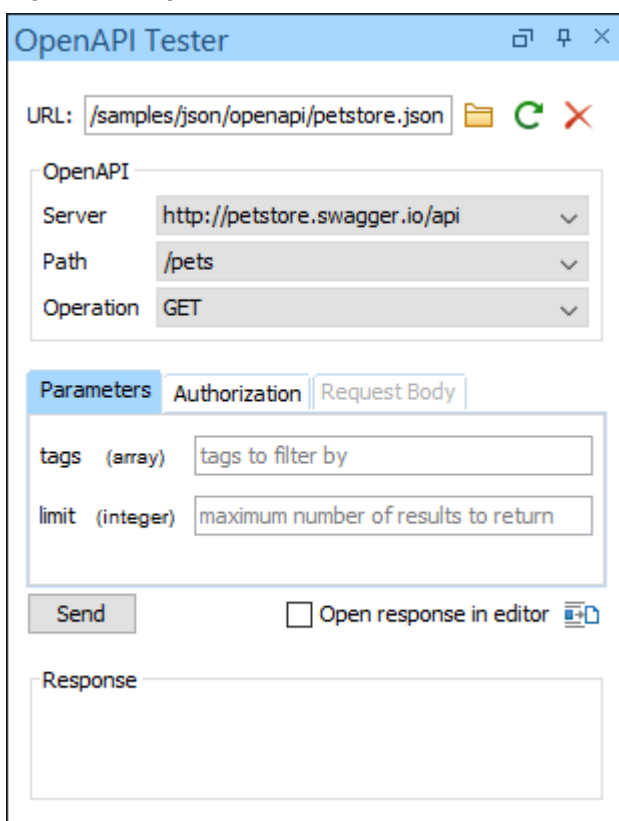
OpenAPI Tester Dialog Box

Figure 154. OpenAPI Tester Dialog Box



After selecting **OpenAPI Tester** from the **Tools > JSON Tools** menu, the **OpenAPI Tester** dialog box is displayed where you can select the URL for the OpenAPI document (either local or remote). After clicking **OK**, the **OpenAPI Tester** view becomes visible on the right side of the editor. The view can also be opened by selecting **OpenAPI Tester** from the **Window > Show View** menu.

Figure 155. OpenAPI Tester View



The tester loads the selected OpenAPI document and fills in the corresponding fields. The tester fields are as follows:

URL

The URL of the OpenAPI file. You can specify the path by using the text field or the **Browse** button. If you specify the path directly in the text field, you need to click the **Reload** button. You can use the **Clear** button if you want to remove all the content from the view.

Server

The list of servers defined by the OpenAPI document. The global "servers" array can be overridden on the path level or operation level. If it is not provided or is empty, the server URL defaults to "/".

Path

The list of individual endpoints (paths) in the API. The full request URL is constructed as `<server-url>/path`.

Operation

The list of HTTP operations supported by the selected path. OpenAPI 3.0 supports get, post, put, patch, delete, head, options, and trace.

Parameters tab

The definition of the parameters for the selected operation. OpenAPI 3.0 supports parameters passed via path, query string, headers, and cookies. The required parameters will be marked with a red asterisk. For array parameters, items must be separated by a comma.

Authorization tab

The list of available authentication types. The authentication data is persistent and can be removed from the contextual menu.

Request Body tab

The media type and the body of the request (if the selected operation allows it). For example, GET will not allow specifying a body since that HTTP method disallows it. Oxygen JSON Editor will create a sample request body based on the JSON Schemas defined in the OpenAPI document. Usually, you just need to change a few values for the request to be valid.

Variables tab

The list of variables used for server templating (if applicable). Variables are indicated by {curly brackets} in the server URL.

Send button

Executes the request by creating a HTTP client with all the information extracted from the view.

Open response in editor

If selected, the response will be opened in the main editing pane.



Generate scenario test

Creates a test scenario file in JSON format, based on the information extracted from the view. The file is then opened in the main editing pane, and can be used to [Run OpenAPI Test Scenario](#). (on page 570)

Response area

Initially this area is empty. After using the **Send** button, it presents the message received from the server in response to the request. The expected content type of the message is JSON. The

response status and possible errors that may occur are presented right below this area. The status has a green foreground for successful requests and a red foreground otherwise.

Resources

For more information about OpenAPI editing, testing, and documenting, watch our webinar:

<https://www.youtube.com/embed/gKdabeh49Qk>

Run OpenAPI Test Scenarios

Oxygen JSON Editor includes a testing tool for running *OpenAPI* test scenarios. The **Run OpenAPI Test Scenario** tool provides the ability to run a test suite for an OpenAPI document in JSON format. It performs the requests based on the specified OpenAPI document and the data entered in the test file, and then checks if the server responses are as expected.

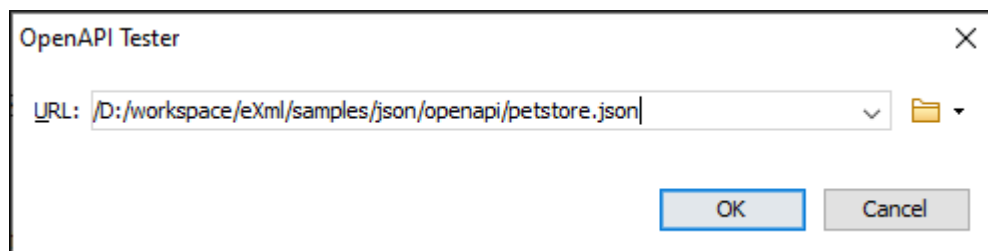


Attention:

This tool requires the **OpenAPI Tester add-on** ([on page 567](#)) (version 1.2.0 or newer) to be installed before it becomes available in the **JSON Tools** menu.

To use the tool, select **Run OpenAPI Test Scenario** from the **Tools > JSON Tools** menu. This opens a dialog box where you can specify the location of the test scenario file that you want to run.

Figure 156. Run OpenAPI Test Scenario Dialog Box



The scenario file must be valid according to the schema from here: [frameworks/json/schemas/openapi/scenario/openAPIScenario.jschema](#). There is a default scenario file template available when creating [new documents from templates](#) ([on page 225](#)) and it can be found in the **Framework Templates > OpenAPI Test Scenario**. The template will automatically be validated against the schema.

For the scenario file, you have to specify the path of the OpenAPI document and the server where the requests are made. Then, for each test, you need to enter valid data for the required fields "**path**", "**operation**", "**expectedResponse**", and the optional fields "**description**", "**parameters**", "**authorization**", or "**body**".

After successfully running the test scenario, the results are displayed in a new JSON file.



Tip:

Oxygen JSON Editor includes a [specialized framework for editing and working with OpenAPI test scenario files](#) ([on page 471](#)).

Resources

For more information about OpenAPI editing, testing, and documenting, watch our webinar:

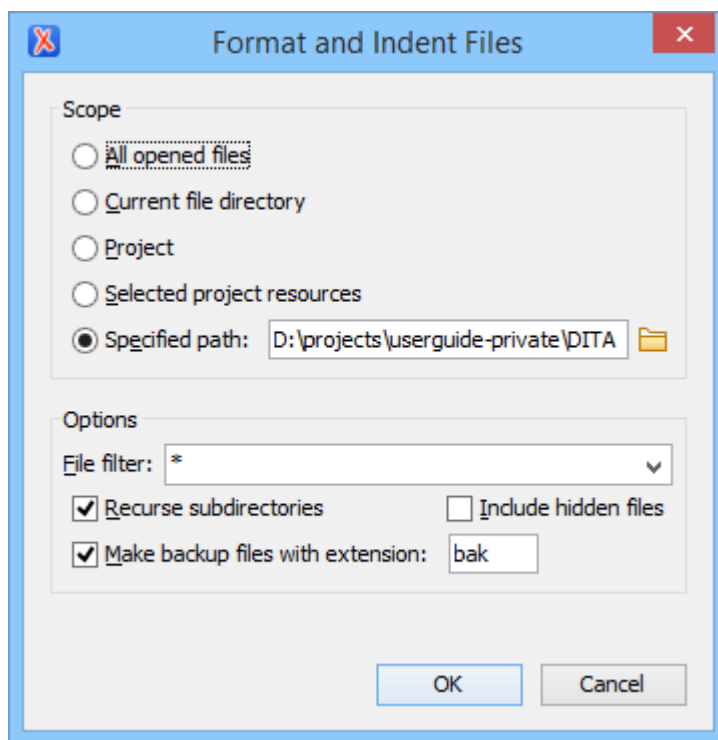
<https://www.youtube.com/embed/gKdabeh49Qk>

Format and Indent (Pretty-Print) Multiple Files

Oxygen JSON Editor provides support for formatting and indenting (*pretty-print (on page 640)*) multiple files at once. This action is available for 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 (on page 251)** or from the **Tools** menu. This opens the **Format and Indent Files** dialog box that allows you to configure options for the action.

Figure 157. Format and Indent Files Dialog Box



The **Scope** section allows you to choose from the following scopes:

- **All opened files** - The *pretty-print (on page 640)* is performed in all opened files.
- **Directory of the current file** - All the files in the folder of the currently edited file.
- **Project files** - All files from the current project.
- **Selected project files** - The selected files from the current project.
- **Specified path** - the *pretty-print (on page 640)* is performed in the files located at a specified path.

The **Options** section includes the following options:

- **File filter** - Allow you to filter the files from the selected scope.
- **Recurse subdirectories** - When selected, the [pretty-print \(on page 640\)](#) is performed recursively for the specified scope. The one exception is that this option is ignored if the scope is set to **All opened files**.
- **Include hidden files** - When selected, the [pretty-print \(on page 640\)](#) is also performed in the hidden files.
- **Make backup files with extension** - When selected, Oxygen JSON Editor makes backup files of the modified files. The default extension is **.bak**, but you can change the extension as you prefer.

Generate Documentation

Oxygen JSON Editor includes a tool for generating documentation for JSON schema, and OpenAPI documents.

Generating JSON Schema Documentation

Oxygen JSON Editor includes a tool for generating documentation for a JSON Schema file in HTML format. To generate JSON Schema documentation, select **JSON Schema Documentation** from the **Tools > Generate Documentation** menu. You can also open the tool by using the  **Generate Documentation** toolbar button. This opens a dialog box where you can specify the location of the JSON Schema file and HTML output file.

This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen JSON Editor presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen JSON Editor and the **JSON Schema Documentation** action will invoke the tool.

Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** to quickly initiate the installation process:

Install

Manual Installation

To manually install it the add-on, follow these instructions:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box.
2. Enter or paste <https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml> in the **Show add-ons from** field or select it from the drop-down menu.



Note:

If you have issues connecting to the default update site, you can [download the add-on package](#), unzip it, then use the **Browse for local files** action in the **Install new add-ons** dialog box to locate the downloaded **addon.xml** file.

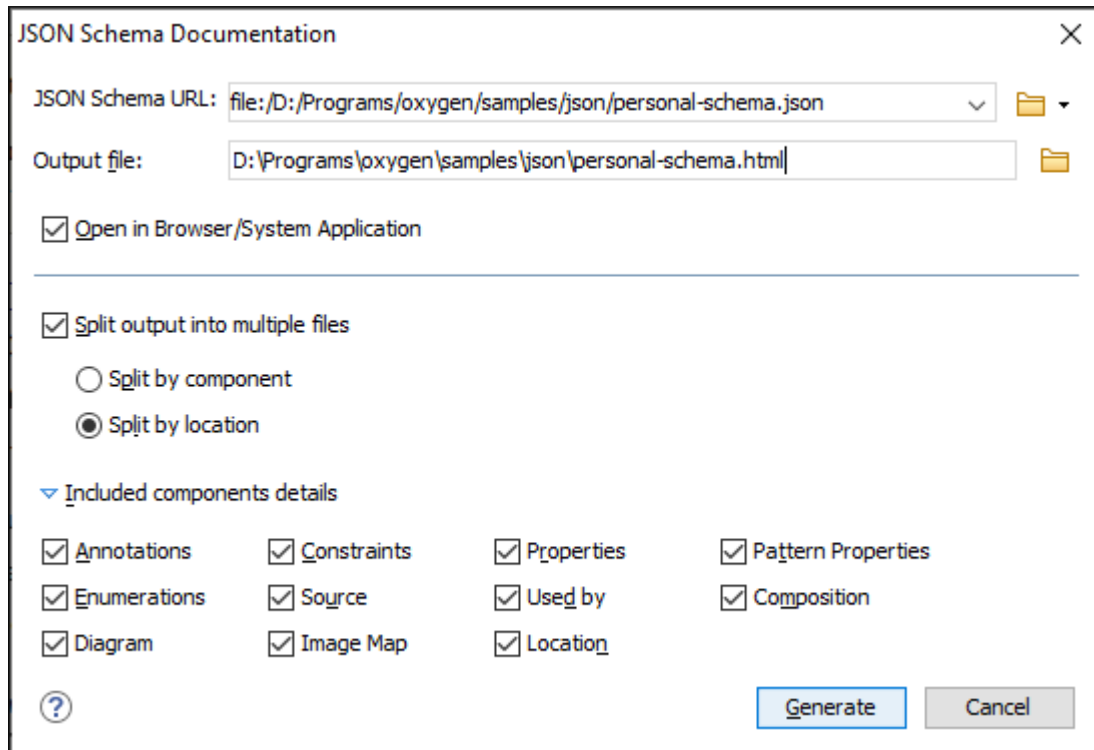
3. Select the **JSON Schema Documentation** add-on and click **Next**.

4. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
5. Restart the application.

Result: The **JSON Schema Documentation** dialog box is now available and can be selected from the **Tools > Generate Documentation** menu.

JSON Schema Documentation Dialog Box

Figure 158. JSON Schema Documentation Dialog Box



The **JSON Schema Documentation** dialog box includes the following fields and options:

JSON Schema URL

The URL of the JSON Schema file. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the **Browse** drop-down list. The tool supports schemas with versions *Draft 04, 06, 07* and, starting with version 4.0.0 of the add-on, *2019-09* and *2020-12*.

Output file

The path to the folder where the generated HTML file will be saved.

Split output into multiple files

If selected, the application splits the output into multiple files. You can choose between splitting them by **component** name or **location**.

Open in Browser/System Application

If selected, the generated result is opened in the system application associated with the output file type (HTML).

Included component details

This section can be used to specify whether or not the following components are shown in the generated documentation:

- **Annotations** - Displays the annotations (title, description) for each component (property or definition).
- **Constraints** - Displays the schema constraints for each component, according to its type.
- **Properties** - Displays the `properties` of an Object Schema.
- **Pattern Properties** - Displays the `patternProperties` of an Object Schema.
- **Enumerations** - Displays the enumerated values, if specified in the schema.
- **Source** - Displays the text schema source for each component.
- **Used By** - Displays the list of all the components that reference the current definition.
- **Composition** - Displays the `oneOf`, `anyOf`, and `allOf` compositors that are used for combining schemas.
- **Diagram** - Displays the diagram image for each component. The diagrams are generated according to the options specified in the [Schema Design preferences page \(on page 131\)](#). Diagrams are also generated for components within schemas from referenced files.
- **Image Map** - Diagrams will include hyperlinks that navigate to each particular component.
- **Location** - Displays the schema location for each component.

You can click **Generate** at any point to generate the JSON Schema documentation.

Generated JSON Schema Documentation in HTML Format

After generating the JSON Schema documentation, it is presented in a visual diagram style with various sections, hyperlinks, and options.

Figure 159. JSON Schema Documentation Example Opened in a Browser

Property personnel

Annotations

Title: The 'personnel' property

Description: Defines the personnel as a collection of person entries.

Diagram

The 'personnel' property
Defines the personnel as a collection of person entries.

Type object

Properties

Name	Occurrence
person	optional

Additional Properties false

Used by

Schema
#/schema

Source

```

{
  "personnel": {
    "type": "object",
    "additionalProperties": false,
    "title": "The 'personnel' property",
    "description": "Defines the personnel as a collection of person entries.",
    "properties": {
      "person": { "$ref": "#/definitions/personType" }
    }
  }
}

```

Location personal-schema.json#/properties/personnel

The generated documentation includes a Table of Contents on the left pane with links to particular sections in the right pane. You can collapse or expand details by using the **Showing** options or the **Collapse** or **Expand** buttons.

Generating OpenAPI Documentation

Oxygen JSON Editor includes a tool for generating documentation for *OpenAPI* 3.0, or 3.1 documents in either JSON or YAML format, including annotations and cross references. The documentation displays information about the servers, paths, components and tags defined in the OpenAPI documents and you can choose whether the output is presented in HTML (with various sections, hyperlinks, and filtering options), DITA, or PDF.

Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** to quickly initiate the installation process:

Install

Manual Installation

To manually install the add-on, follow these instructions:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box.
2. Enter or paste **<https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml>** in the **Show add-ons from** field or select it from the drop-down menu.

**Note:**

If you have issues connecting to the default update site, you can [download the add-on package](#), unzip it, then use the **Browse for local files** action in the **Install new add-ons** dialog box to locate the downloaded **addon.xml** file.

3. Select the **OpenAPI Documentation Generator** add-on and click **Next**.
4. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
5. Restart the application.

Result: The **OpenAPI Documentation** dialog box is now available and can be selected from the **Tools > Generate Documentation** menu.

Generating OpenAPI Documentation

To generate OpenAPI documentation, select **OpenAPI Documentation** from the **Tools > Generate Documentation** menu. This opens a dialog box where you can specify the location of the OpenAPI file and the output file, as well as the type of output to generate.

This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen JSON Editor presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen JSON Editor and the **OpenAPI Documentation** action will invoke the tool.

Figure 160. OpenAPI Documentation Dialog Box

The **OpenAPI Documentation** dialog box includes the following fields and options:

OpenAPI URL

The URL of the OpenAPI file (it can be in either JSON or YAML format). You can specify the path by using the text field or the browsing button (📁).

Output file

The path to the folder where the generated file(s) will be saved.

Generate HTML output

Choose this option to generate the output in HTML format.

Split output into multiple files

If selected, the application splits the output into multiple files. You can choose between splitting them by **component** name or **location**.

Included JSON schema component details

This section can be used to specify whether or not details about the following components that belong to internal or imported schemas are shown in the generated documentation:

- **Annotations** - Displays the annotations (title, description) for each component (property or definition).
- **Constraints** - Displays the schema constraints for each component, according to its type.
- **Source** - Displays the text schema source for each component.
- **Location** - Displays the schema location for each component.
- **Used By** - Displays the list of all the components that reference the current definition.
- **Properties** - Displays the `properties` of an Object Schema.
- **Pattern Properties** - Displays the `patternProperties` of an Object Schema.
- **Enumerations** - Displays the enumerated values, if specified in the schema.
- **Diagram** - Displays the diagram image for each component. The diagrams are generated according to the options specified in the [Schema Design preferences page \(on page 131\)](#). Diagrams are also generated for components within schemas from referenced files.
- **Composition** - Displays the `oneOf`, `anyOf`, and `allOf` compositors that are used for combining schemas.

Generate DITA output

Choose this option to generate the output in DITA format.

Create DITA map from OpenAPI document

If selected, the application generates a DITA map with referenced topics based on the input OpenAPI document.

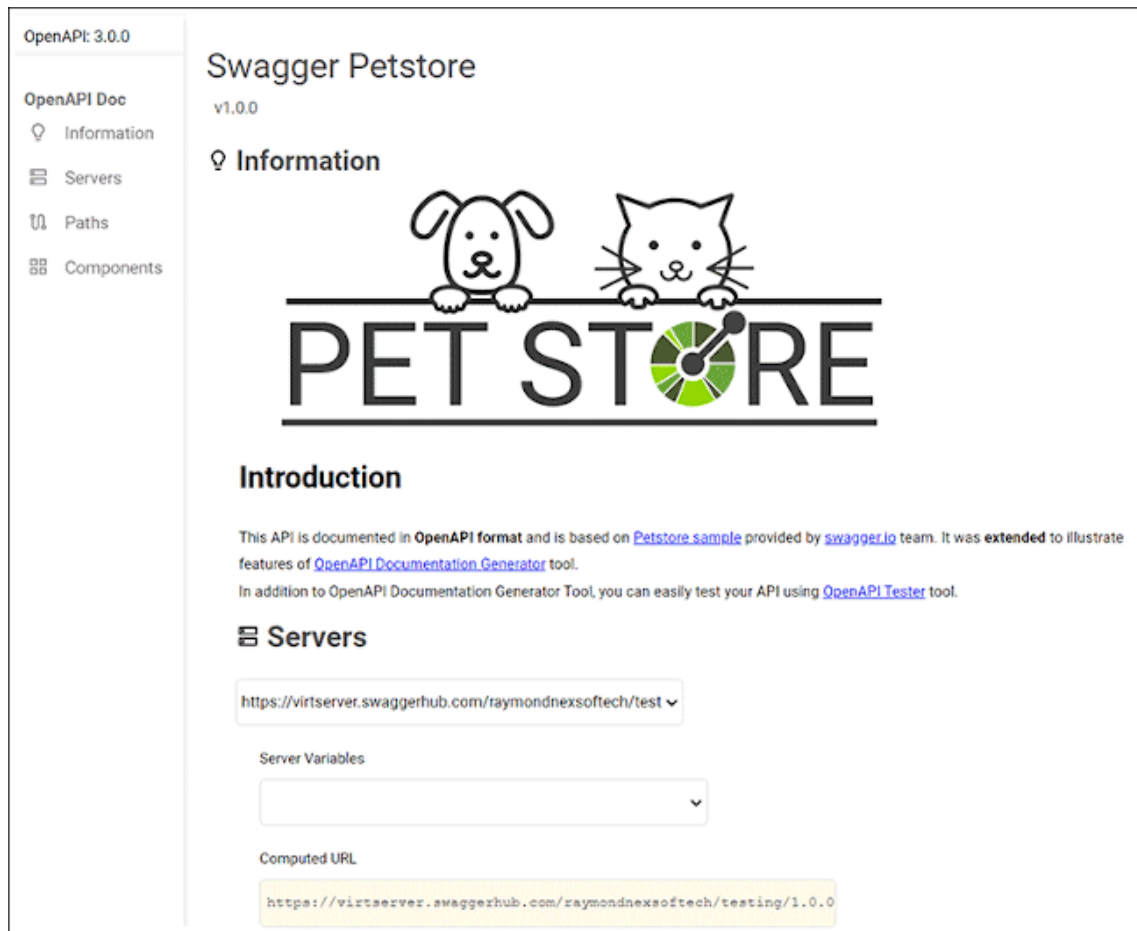
Generate PDF output

Choose this option to generate the output in PDF format.

Click the **Generate** button to generate the OpenAPI documentation.

Generated OpenAPI Documentation in HTML Format

When generating the OpenAPI documentation in HTML format, it is presented in the browser with various sections, hyperlinks, and options.

Figure 161. OpenAPI Documentation Example

The generated documentation includes a Table of Contents on the left pane with links to particular sections in the right pane. You can collapse or expand details by using the **Collapse** or **Expand** buttons.

Resources

For more information about OpenAPI editing, testing, and documenting, watch the following webinar:

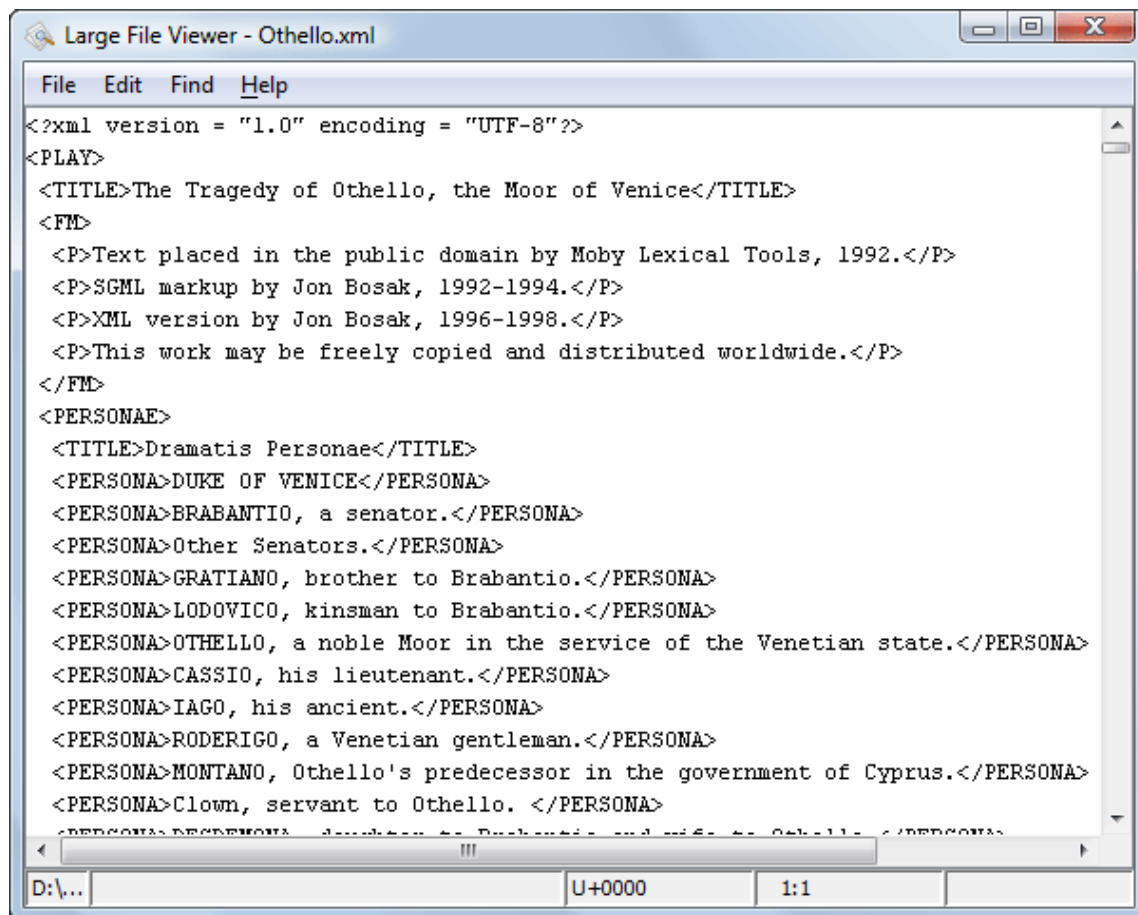
<https://www.youtube.com/embed/gKdabeh49Qk>

Large File Viewer

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 open 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 open 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 JSON Editor options. See [Encoding for non-XML files \(on page 114\)](#).

Figure 162. Large File Viewer



Large File Viewer components:

- The menu bar provides menu driven access to all the features and functions that are available in **Large File Viewer**:

File > Open

Opens files in the viewer (also available in the contextual menu).

File > Exit

Closes the viewer.

Edit > Copy

Copies the selected text to clipboard (also available in the contextual menu).

Find > Find

Opens a reduced **Find** dialog box that provides some basic search options, such as:

- **Case sensitive** - When selected, operations are case-sensitive.
- **Regular Expression** - When selected, allows you to use any regular expression in [Perl-like syntax \(on page 287\)](#).
- **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 file path, the Unicode representation of the character at the cursor position and the line and column in the open 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 \(on page 74\)](#).



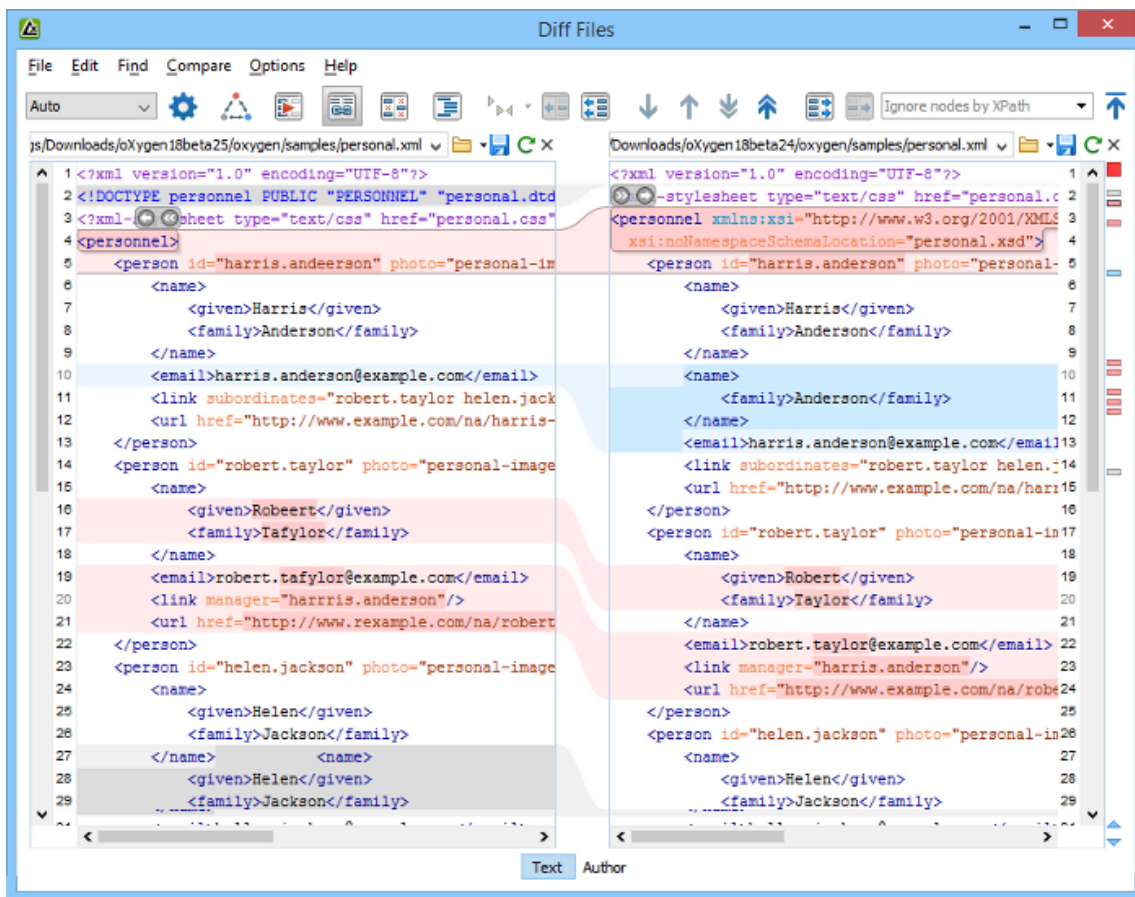
Tip:

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 164. 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 actions in the  **Browse** drop-down menu.

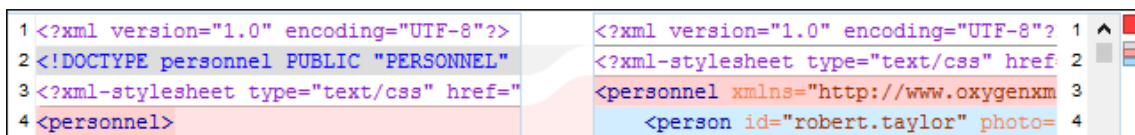
Step Result: The selected files are opened in the two side-by-side editors. A text editing mode is used to offer a better view of the differences.

2. To highlight the differences between the two files, click the  **Perform File Differencing** button from the toolbar.
3. You can use the drop-down menu on the left side of the toolbar to change the [algorithm \(on page 585\)](#) 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 165. 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 \(on page 168\)](#), 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.

Comparing Fragments (Copy/Paste)

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 macOS))** button, before pasting the fragments. Other notes for pasting fragments:

- As long as the fragment is more than 10 characters, the application will attempt to automatically detect the content type. It can detect the following types: XML, DTD, CSS, JSON, and Markdown (if it starts with #). If one of those content types is detected, the fragments will be displayed with syntax highlights.
- 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 \(on page 324\)](#) and in the [various menus \(on page 327\)](#) (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 JSON Editor offers the following two-way diff algorithms to compare files or fragments:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Characters** - Computes the differences at character level, meaning that it compares two files or fragments looking for identical characters. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in **Author** comparison mode.

- **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).

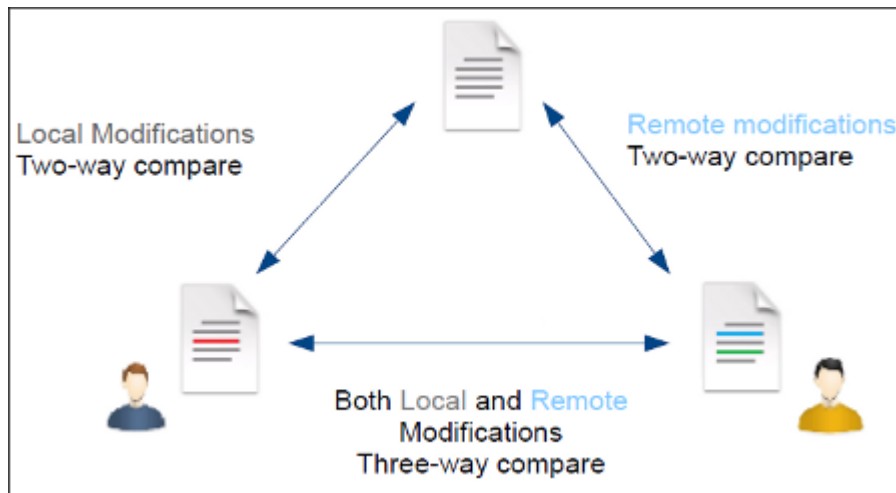
For example:

- 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 JSON Editor also includes a three-way comparison feature to help you solve conflicts and merge changes between multiple modifications. It is especially helpful for teams who have multiple authors editing and committing the same documents. It provides a comparison between a local change, another change, and the original base revision. Some additional advantages include:

- Visualize and merge content that was modified by you and another member of your team.
- Marks differences correctly even when the document structure is rearranged.
- Allows you to merge XML-relevant modifications.

Figure 166. 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 actions in the ▾ **Browse** drop-down menu.

Step Result: The selected files are opened in the two side-by-side editors. A text editing mode is used to offer a better view of the differences.

2. Click the **Three-Way Comparison** button on the toolbar and select the base (original) file in the **Base** field. You can specify the path by using the text field, the history drop-down, or the browsing actions 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 \(on page 585\)](#) 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 (original) file. The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes.

Figure 167. Three-Way Differences

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

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 \(on page 168\)](#), 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 \(on page 324\)](#) and in the [various menus \(on page 327\)](#) (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

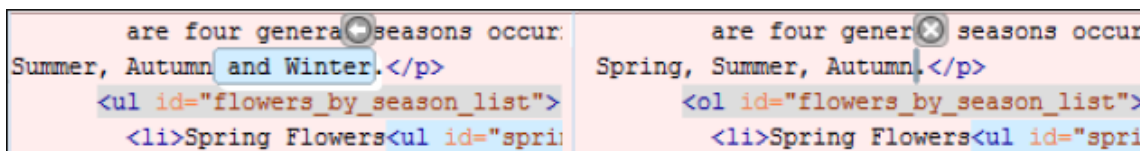
Oxygen JSON Editor offers the following three-way diff algorithms to compare files:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Second-Level Comparisons

For both two-way and three-way comparisons, Oxygen JSON Editor automatically performs a second-level comparison for the **Lines**, **XML Fast**, and **XML Accurate** algorithms. After the first comparison is finished, the second-level comparison for the **Lines** algorithm is processed on text nodes using a word level comparison, meaning that it looks for identical words. For the **XML Fast** and **XML Accurate** algorithms, the second-level comparison is processed using a [syntax-aware comparison \(on page 586\)](#), meaning that it looks for identical *tokens*. This second-level comparison makes it easier to spot precise differences and you can merge or reject the precise modifications.

Figure 168. Second-Level Diff Comparison

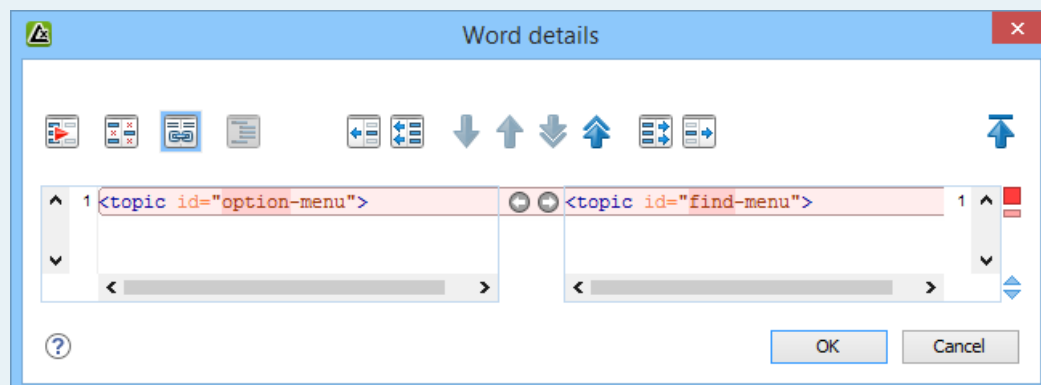


Note:

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.

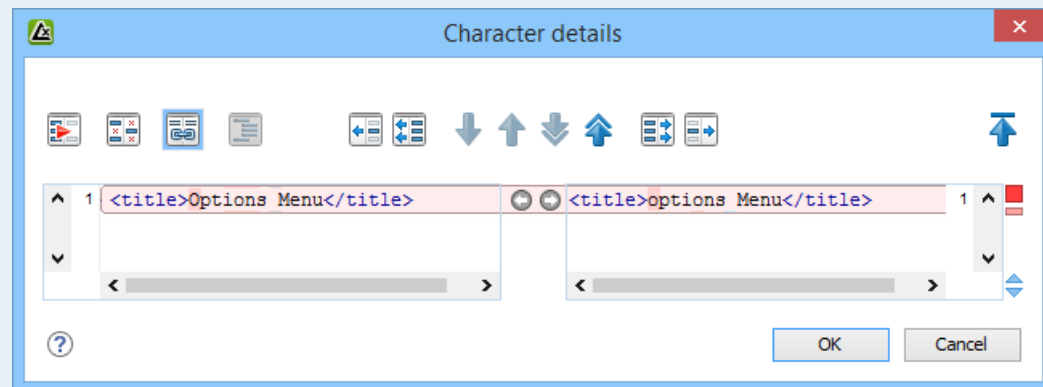
Figure 169. Word Level Comparison





To do a character level comparison, select **Show Character Level details** from the contextual menu or **Compare** menu.

Figure 170. Character Level Comparison



Related information

[Files Comparison Preferences Page \(on page 166\)](#)

[Compare Directories Tool \(on page 332\)](#)

Starting File Comparison Tool from a Command Line

The file comparison tool can be started by using command-line arguments. In the installation folder there is an executable shell (**diffFiles.bat** on Windows, **diffFiles.sh** on macOS and Linux). To specify the files to compare, you can pass command-line arguments using the following construct: `diffFiles.bat/diffFiles.sh [path to left file] [path to right file] [path to 3-way base file]`.

If three files are specified, the tool will start in the [3-way comparison mode \(on page 316\)](#). If only two files are specified, the tool will start in the [2-way comparison mode \(on page 313\)](#). The first specified file will be added to the left panel in the comparison tool, the second file to the right panel, and the optional third file will be the base (ancestor) file used for a 3-way comparison. If you pass only one argument, you are prompted to manually choose another file.

If you want to launch the file comparison tool from an external application with specified files and you want the file browsing buttons at the top of both panels to be hidden, you should use the `-ext` argument as the first command. There are some additional arguments that are allowed and to see all the details for the command-line construct, type `diffFiles.bat --help` in the command line.

Example:

To do a 3-way comparison, the command line might look like this:

Windows

```
diffFiles.bat "c:\docs\file 1" "c:\docs\file 2" c:\docs\basefile
```

**Tip:**

If there are spaces in the path names, surround the paths with quotes.

Linux

```
diffFiles.sh home/file1 home/file2 home/basefile
```

macOS

```
diffFiles.sh documents/file1 documents/file2 documents/basefile
```

How to Integrate the File Comparison Tool with Git

The file comparison tool can be integrated with Git clients. It requires that you configure your `.gitconfig` file and then you can simply start the tool from the command line.

To integrate the **Compare Files** tool with your Git client, follow this procedure:

1. Use one of the following methods to instruct your Git client to use the *Oxygen Compare Files* tool:
 - **Manual Configuration** - Locate your Git user-specific configuration file (`.gitconfig`) and edit it with a text editor (for example, in Windows, the `.gitconfig` file is most likely located in your user home directory). Add (or replace) the following lines:

```
[diff]
    tool = oxygendiff

[merge]
    tool = oxygendiff

[difftool "oxygendiff"]
    cmd = '[pathToOxygenInstallDir]/diffFiles.exe' -ext $REMOTE $LOCAL $LOCAL

[mergetool "oxygendiff"]
    cmd = '[pathToOxygenInstallDir]/diffFiles.exe' -ext $LOCAL $REMOTE $BASE $MERGED
    trustExitCode = true

[difftool]
    prompt = false
```

**Note:**

For macOS, the `cmd` lines would start with something like: `sh "/Applications/Oxygen XML Editor/diffFiles.sh"`. For Linux, the `cmd` lines would start with something like: `sh "/Oxygen XML Editor/diffFiles.sh"`.

**Tip:**

On Redhat 7, the following command would work, where the whole command is quoted and then inside that, the path to `diffFiles.sh` is quoted:

**[difftool "oxygendiff"]**

```
cmd = '/home/user/Oxygen XML Editor 21/diffFiles.sh' -ext $REMOTE $LOCAL
$LOCAL
```

[mergetool "oxygendiff"]

```
cmd = '/home/user/Oxygen XML Editor 21/diffFiles.sh' -ext $LOCAL $REMOTE
$BASE
```

```
$MERGED trustExitCode = true
```

- **Command Line Configuration** - To automatically configure the `.gitconfig` file, you can run the following commands from a command line:

```
git config --global diff.tool oxygendiff
git config --global difftool.oxygendiff.cmd '[Oxygen install dir]/diffFiles.exe -ext
$REMOTE $LOCAL $LOCAL'
git config --global merge.tool oxygendiff
git config --global mergetool.oxygendiff.cmd '[Oxygen install dir]/diffFiles.exe
-ext $LOCAL $REMOTE $BASE $MERGED'
git config --global mergetool.oxygendiff.trustExitCode true
```

**Note:**

For macOS, the *Oxygen* file comparison tool would be specified in the second and fourth commands with something like: `sh "/Applications/Oxygen XML Editor/diffFiles.sh"`. For Linux, it would be something like: `sh "/Oxygen XML Editor/diffFiles.sh"`.

2. To start the **Compare Files** tool and see a comparison of changes for a particular file, run the following command from a command line:

```
git difftool [PathToFile]
```

**Tip:**

If the file you want to compare has conflicts, you can start the **Compare Files** tool as a *merge conflict resolution* tool by running the following command:

```
git mergetool [PathToFile]
```

For more information about the Git *difftool* syntax, see <https://git-scm.com/docs/git-difftool>.

For more information about the Git *mergetool* syntax, see <https://git-scm.com/docs/git-mergetool>.

How to Integrate the File Comparison Tool with Sourcetree

The file comparison tool can be integrated with Sourcetree so that you can use it to compare changes. The advantages of doing this include:

- The *Oxygen Compare Files* tool presents the files side-by-side and makes it much easier to determine real changes.
- The *Oxygen Compare Files* tool includes XML comparison algorithms.
- The *Oxygen Compare Files* tool includes various options for configuring the comparison.
- The *Oxygen Compare Files* tool allows you to navigate through changes.

To integrate the **Compare Files** tool with Sourcetree, follow this procedure, depending on your operating system:

Windows

1. In Sourcetree, go to **Tools > Options**.
2. Go to the **Diff** tab.
3. In the **External Diff/Merge** section, configure the settings as follows:
 - **External Diff Tool** - Select **Custom**.
 - **Diff Command** - Enter the path of the *Oxygen diffFile.exe* file (for example: `c:\Programs\Oxygen XML Editor\diffFiles.exe`).
 - **Arguments** - Enter `-ext $REMOTE $LOCAL $LOCAL`.
 - **Merge Tool** - Select **Custom**.
 - **Diff Command** - Enter the path of the *Oxygen diffFile.exe* file (for example: `c:\Programs\Oxygen XML Editor\diffFiles.exe`).
 - **Arguments** - Enter `-ext $LOCAL $REMOTE $BASE $MERGED`.
4. Click **OK**.

Result: In Sourcetree, you can now compare file changes with the *Oxygen Compare Files* tool by simply selecting **External Diff** from the contextual menu, **Actions** menu, or **Ctrl+D**.

macOS

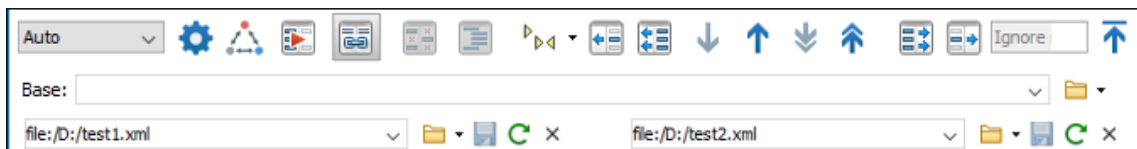
1. In Sourcetree, go to **Sourcetree > Preferences**.
2. Go to the **Diff** tab.
3. In the **External Diff/Merge** section, configure the settings as follows:
 - **External Diff Tool** - Select **Custom**.
 - **Diff Command** - Enter a command-line argument to launch the *Oxygen diffFiles.sh* file (for example: `sh "/Applications/Oxygen XML Editor/diffFiles.sh"`).
 - **Arguments** - Enter `-ext $REMOTE $LOCAL $LOCAL`.
 - **Merge Tool** - Select **Custom**.
 - **Diff Command** - Enter a command-line argument to launch the *Oxygen diffFiles.sh* file (for example: `sh "/Applications/Oxygen XML Editor/diffFiles.sh"`).
 - **Arguments** - Enter `-ext $LOCAL $REMOTE $BASE $MERGED`.
4. Close the preferences dialog box.

Result: In Sourcetree, you can now compare file changes with the *Oxygen Compare Files* tool by simply selecting **External Diff** from the contextual menu or **Actions** menu.

Toolbar and Contextual Menu Actions of the Compare Files Tool

The toolbar of the **Compare Files** tool contains operations that can be performed on the source and target files or XML fragments. Many of the actions are also available in the contextual menu.

Figure 171. 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. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **Syntax Aware** - Computes differences for the file types or fragments known by Oxygen JSON Editor, taking the syntax (the specific types of tokens) into consideration. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **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 \(on page 166\)](#) where you can configure various options.

Three-Way Comparison

Toggle action that allows you to perform a three-way comparison between the two files displayed in the two editing panes and a base (ancestor) file.



Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.



Synchronized scrolling

Toggles synchronized scrolling on or off so that a selected difference can be seen on both sides of the application window. This option is on by default.



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. This option is not available when in the **Author** comparison mode.



Format and Indent Both Files (**Ctrl + Shift + P** (**Command + Shift + P** on macOS))

Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.



Note:

When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.



Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.



Copy All Changes from Right to Left

Copies all changes from the file in the right panel to the file in the left panel.



Next Block of Changes (**Ctrl + Period** (**Command + Period** on macOS))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.



Note:

A change block groups one or more consecutive lines that contain at least one change.



Previous Block of Changes (**Ctrl + Comma** (**Command + Comma** on macOS))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.



Next Change (**Ctrl + Shift + Period** (**Command + Shift + Period** on macOS))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.



Previous Change (Ctrl + Shift + Comma (Command + Shift + M on macOS))

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.



Copy All Changes from Left to Right

Copies all changes from the file in the left panel to the file in the right panel.



Copy Change from Left to Right

Copies the selected difference from the file in the left panel to the file in the right panel.

Ignore Nodes by XPath

You can use this text field to enter an [XPath expression \(on page 474\)](#) to ignore certain nodes from the comparison. It will be processed as XPath version 2.0. You can also enter the name of the node to ignore all nodes with the specified name (for example, if you want to ignore all ID attributes from the document, you could simply enter `@id`). This field is only available when comparing XML documents using the **XML Fast** or **XML Accurate** algorithms.



Note:

If an XPath expression is specified in the **Ignore nodes by XPath** option ([on page 168](#)) 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 macOS))

Jumps to the first change.

Base

Available for [three-way comparisons \(on page 316\)](#). It is the base file that will be compared with the files opened in the left and right editors. You can specify the path to the file by using the text field, its history drop-down, or the browsing actions in the **Browse** drop-down menu.

Left-Side (Source) File

You can specify the path to the file to be compared on the left side (source) by using the text field, its history drop-down, or the browsing actions in the **Browse** drop-down menu.



Save

Saves the changes made in the source (left-side) file.

 Reload

Reloads the source (left-side) file.

 Close

Closes the source (left-side) file.

Right-Side (Target) File

You can specify the path to the file to be compared on the right side (target) by using the text field, its history drop-down, or the browsing actions in the  **Browse** drop-down menu.

 Save

Saves the target (right-side) file.

 Reload

Reloads the target (right-side) file.

 Close

Closes the target (right-side) file.

Compare Files Tool Menus

The menus in the **Compare Files** tool contain some of the same actions that are on the toolbar, as well as some common actions that are identical to the same actions in the Oxygen JSON Editor menus. The menu actions include:

File Menu

Source > Open

Browses for a file that will be displayed in the left panel.

Source > Open URL

Browses for a remote file that will be displayed in the left panel.

Source > Open File from Archive

Browses an archive for a file that will be displayed in the left panel.

Source > Reload

Reloads the file in the left panel.

Source > Save

Saves the changes made to the file in the left panel.

Source > Save As

Allows you to choose a destination to save the file in the left panel.

Source > Close

Closes the file in the left panel.

Target > Open

Browses for a file that will be displayed in the right panel.

Target > Open URL

Browses for a remote file that will be displayed in the right panel.

Target > Open File from Archive

Browses an archive for a file that will be displayed in the right panel.

Target > Reload

Reloads the file in the right panel.

Target > Save

Saves the changes made to the file in the right panel.

Target > Save As

Allows you to choose a destination to save the file in the right panel.

Target > Close

Closes the file in the right panel.

Base > Open

Browses for a file that will be compared with both files in a [three-way comparison \(on page 316\)](#).

Base > Open URL

Browses for a remote file that will be compared with both files in a [three-way comparison \(on page 316\)](#).

Base > Open File from Archive

Browses an archive for a file that will be compared with both files in a [three-way comparison \(on page 316\)](#).

** Save Results as HTML (Available in Text mode only)**

Generates an HTML file that contains detailed information about the comparison result.

Save Comparison as Document with Tracked Changes (Available for two-way comparison in Author mode only)

Allows you to merge two compared documents based on the differences detected and save the results as a specified file that includes the special change tracking marks. You can load the resulting file in **Oxygen's Author** mode to review the changes that resulted from the merge process and you can accept or reject them. Note that if the documents to be compared already contain tracked changes, they will be automatically accepted before generating the output file.

Close (Ctrl + W (Command + W on macOS))

Closes the application.

Edit Menu



Cut

Cut the selection from the currently focused editor panel to the clipboard.



Copy

Copy the selection from the currently focused editor panel to the clipboard.



Paste

Paste content from the clipboard into the currently focused editor panel.

Select all

Selects all content in the currently focused editor panel.



Undo

Undo changes in the currently focused editor panel.



Redo

Redo changes in the currently focused editor panel.

Find Menu



Find/Replace

Perform *find/replace* operations in the currently focused editor panel.

Find Next

Go to the next match using the same options as the last *find* operation. This action runs in both editor panels.

Find Previous

Go to the previous match using the same options as the last *find* operation. This action runs in both editor panels.

Compare Menu



Three-Way Comparison

Toggle action that allows you to perform a three-way comparison between the two files displayed in the two editing panes and a base (ancestor) file.



Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.



Next Block of Changes (Ctrl + Period (Command + Period on macOS))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.



Note:

A change block groups one or more consecutive lines that contain at least one change.



Previous Block of Changes (Ctrl + Comma (Command + Comma on macOS))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.



Next Change (Ctrl + Shift + Period (Command + Shift + Period on macOS))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.



Previous Change (Ctrl + Shift + Comma (Command + Shift + M on macOS))

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.



Last Change (Ctrl + E (Command + E on macOS))

Jumps to the last change.



First Change (Ctrl + B (Command + B on macOS))

Jumps to the first change.



Copy All Changes from Right to Left

Copies all changes from the file in the right panel to the file in the left panel.



Copy All Changes from Left to Right

Copies all changes from the file in the left panel to the file in the right panel.



Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.



Copy Change from Left to Right

Copies the selected difference from the file in the left panel to the file in the right panel.



Show Word Level Details

Provides a word-level comparison of the selected change.



Show Character Level Details

Provides a character-level comparison of the selected change.



Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on macOS))

Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.

**Note:**

When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.

Options Menu

Preferences

Opens the preferences dialog box that includes numerous pages of options that can be configured.

Menu Shortcut Keys

Opens the **Menu Shortcut Keys** option page where you can configure keyboard shortcuts available for menu items.

Reset Global Options

Resets options to their default values. Note that this option appears only when the tool is executed as a stand-alone application.

Import Global Options

Allows you to import an options set that you have previously exported.

Export Global Options

Allows you to export the current options set to a file.

Help Menu

Help (F1)

Opens a **Help** dialog box that displays the User Manual at a section that is appropriate for the context of the current cursor position.

Use Online Help

If this option is selected, when you select Help or press F1 while hovering over any part of the interface, Oxygen JSON Editor attempts to open the help documentation in online mode. If this option is not selected or an internet connection fails, the help documentation is opened in offline mode.

Report problem

Opens a dialog box that allows the user to write the description of a problem that was encountered while using the application. You can change the URL where the reported problem is sent by using the `com.oxygenxml.report.problems.url` system property. The report is sent in XML format through the `report` parameter of the POST HTTP method.

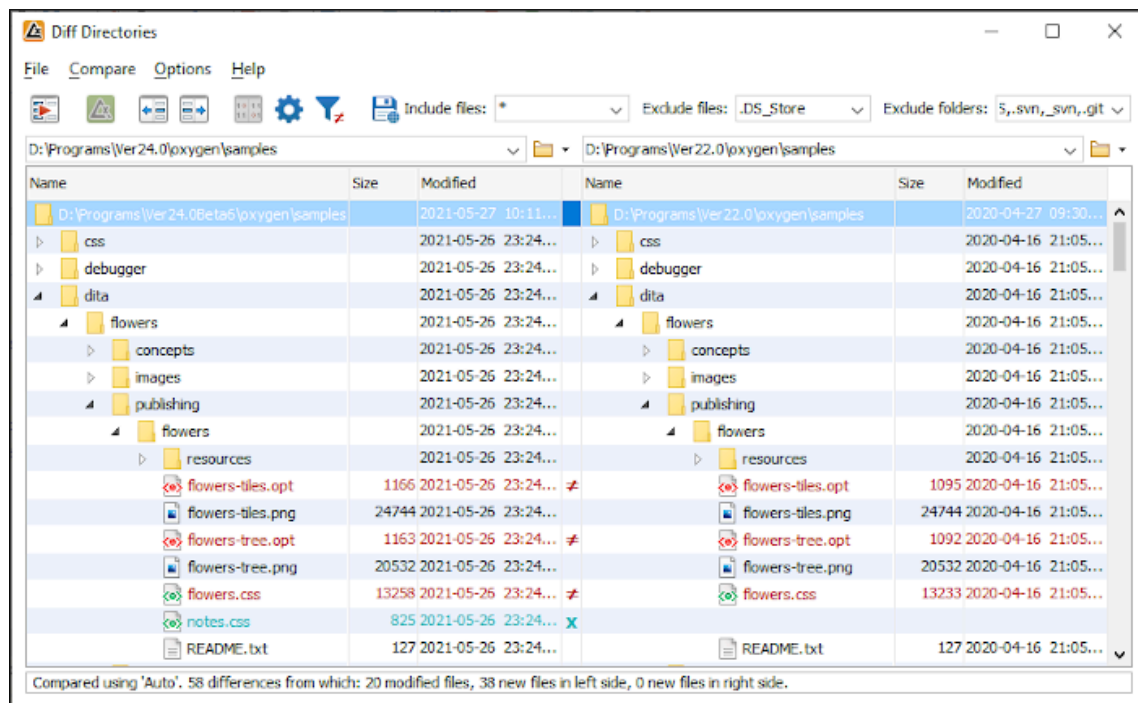
Support Center

Opens the Oxygen JSON Editor Support Center web page in a browser.

Compare Directories Tool

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 JSON Editor installation folder (**diffDirs.exe**).

Figure 172. Diff Directories Dialog Box



Starting the Tool from a Command Line

The directory comparison tool can also be started by using command-line arguments. In the installation folder there is an executable shell (**diffDirs.bat** on Windows, **diffDirs.sh** on macOS and Linux). To specify the directories to compare, you can pass command-line arguments using the following construct: `diffDirs.bat/diffDirs.sh [directory path 1] [directory path 2]`.

If you pass only one argument, you are prompted to manually choose the second directory or archive.

Example:

To do a comparison between two directories, the command line would look like this:

Windows

```
diffDirs.bat "c:\documents new" "c:\documents old"
```



Tip:

If there are spaces in the path names, surround the paths with quotes.

Linux

```
diffDirs.sh home/documents1 home/documents2
```

macOS

```
diffDirs.sh documents1 documents2
```

Directory Comparisons

To perform a directory comparison, follow these steps:

1. Select a folder in the left panel and the folder you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the **Browse for local directory** action in the  **Browse** drop-down menu.

Step Result: The selected directory structures are opened in the two side-by-side panels.

2. To highlight the differences between the two folders, click the  **Perform Directories Differencing** button from the toolbar.
3. You can also use the  **Diff Options** button to access the [Directories Comparison preferences page \(on page 169\)](#) 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 JSON Editor treat supported archives as directories, select the [Look in archives option \(on page 170\)](#) 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 \(on page 170\)](#). You can double-click lines marked with the **#** symbol to open a **Compare Files** window, which shows the differences between the two files.

The directories that contain files that differ are expanded automatically so that you can focus directly on the differences. You can merge the contents of the directories by using the copy actions. If you double-click (or

press **Enter**) on a line with a pair of files, Oxygen JSON Editor starts a [file comparison \(on page 313\)](#) between the two files, using the **Compare Files** tool.

Related information

[Compare Files Tool \(on page 313\)](#)

Toolbar and Contextual Menu Actions of the Compare Directories Tool

The toolbar of the **Compare Directories** tool contains operations that can be performed on the compared directory structure. Some of the toolbar actions are also available in the contextual menu.

Figure 173. Compare toolbar



Toolbar Actions



Perform Directories Differencing

Looks for differences between the two directories displayed in the left and right side of the application window.



Perform Files Differencing

Opens the [Compare Files tool \(on page 313\)](#) 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 \(on page 169\)](#) where you can configure various options.



Show Only Modifications

Displays a more uncluttered file structure by hiding all identical files.



Save Results as HTML

Generates an HTML file that contains detailed information about the comparison result.

File and folder filters

Differences can be filtered using three combo boxes: **Include files**, **Exclude files**, and **Exclude folders**. They come with predefined values and are editable to allow custom values. All of them accept multiple comma-separated values and the * and ? wildcards. For example, to filter out all JPEG and GIF image files, edit the **Exclude files** filter box to read *.jpeg, *.png. Each filter includes a drop-down menu with the latest 15 filters applied.

Contextual Menu Actions



Perform Files Differencing

Opens the **Compare Files** tool ([on page 313](#)) that allows you to compare the currently selected files.



Binary Compare

Performs a byte-level comparison on the selected files.



Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.



Copy Change from Left to Right

Copies the selected difference from the file in the left panel to the file in the right panel.

Open

If the action is invoked on a file, the selected file is opened in Oxygen JSON Editor. If the action is invoked on a directory, the selected directory is opened in the default file browser for your particular operating system.

Open in System Application

Opens the selected file in the system application that is associated with that type of file. The action is available when launching the **Compare Directories** tool from the **Tools** menu in Oxygen JSON Editor.

Show in Explorer

Opens the default file browser for your particular operating system with the selected file highlighted.

Compare Directories Tool Menus

The menus in the **Compare Directories** tool contain some of the same actions that are on the toolbar, as well as some common actions that are identical to the same actions in the Oxygen JSON Editor menus. The menu actions include:

File Menu



Save Results as HTML

Generates an HTML file that contains detailed information about the comparison result.

Close (Ctrl + W (Command + W on macOS))

Closes the application.

Compare Menu



Perform Directories Differencing

Looks for differences between the two directories displayed in the left and right side of the application window.



Perform Files Differencing

Opens the [Compare Files tool \(on page 313\)](#) that allows you to compare the currently selected files.



Copy Change from Right to Left

Copies the selected change from the right side to the left side (if there is no file/folder in the right side, the left file/folder is deleted).



Copy Change from Left to Right

Copies the selected change from the left side to the right side (if there is no file/folder in the left side, the right file/folder is deleted).

Options Menu

Preferences

Opens the preferences dialog box that includes numerous pages of options that can be configured.

Menu Shortcut Keys

Opens the **Menu Shortcut Keys** option page where you can configure keyboard shortcuts available for menu items.

Reset Global Options

Resets options to their default values. Note that this option appears only when the tool is executed as a stand-alone application.

Import Global Options

Allows you to import an options set that you have previously exported.

Export Global Options

Allows you to export the current options set to a file.

Help Menu

Help (F1)

Opens a **Help** dialog box that displays the User Manual at a section that is appropriate for the context of the current cursor position.

Use Online Help

If this option is selected, when you select Help or press F1 while hovering over any part of the interface, Oxygen JSON Editor attempts to open the help documentation in online mode. If this option is not selected or an internet connection fails, the help documentation is opened in offline mode.

Report problem

Opens a dialog box that allows the user to write the description of a problem that was encountered while using the application. You can change the URL where the reported problem is sent by using the `com.oxygenxml.report.problems.url` system property. The report is sent in XML format through the `report` parameter of the POST HTTP method.

Support Center

Opens the Oxygen JSON Editor Support Center web page in a browser.

Compare Images

You can use the **Compare Directories** tool to compare images. If you double-click a line that contains two different images, the **Compare images** window is displayed. This dialog box presents the images in the left and right sides, scaled to fit the available view area. You can use the contextual menu actions to scale the images to their original size or scale them down to fit in the view area.

The supported image types are: *GIF*, *JPG*, *JPEG*, *PNG*, and *BMP*.

Compare Directories Against a Base (3-Way) Tool

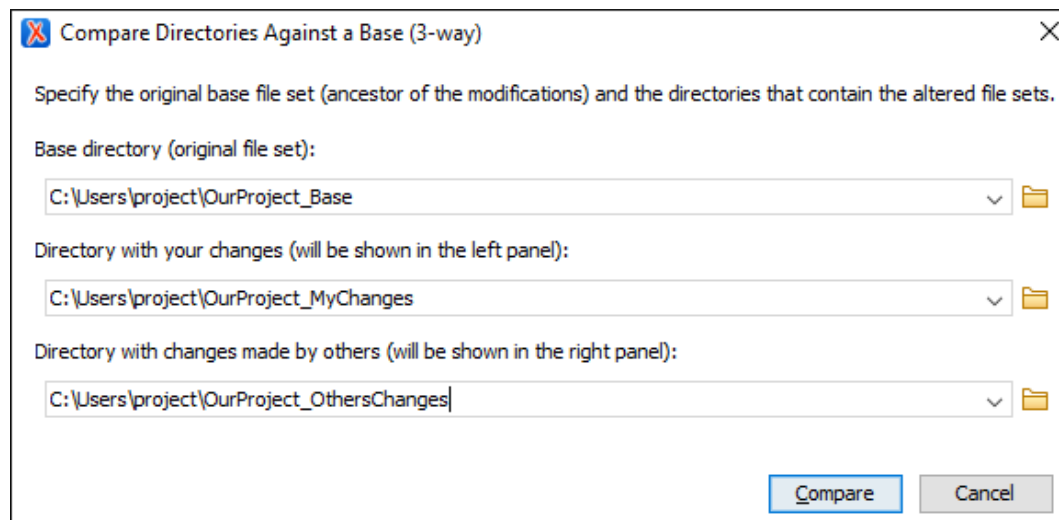
The **Compare Directories Against a Base (3-way)** tool allows you to perform three-way comparisons on directories to help you identify and merge changes between multiple modifications of the same directory structure. It is especially helpful for teams that have multiple authors contributing documents to the same directory system. It offers information about conflicts and changes, and includes actions to easily merge, accept, overwrite, or ignore changes to the directory system.

How to Perform 3-Way Directory Comparisons

To perform a 3-way directories comparison, follow these steps:

1. Select  **Compare Directories Against a Base (3-way)** from the **Tools** menu.

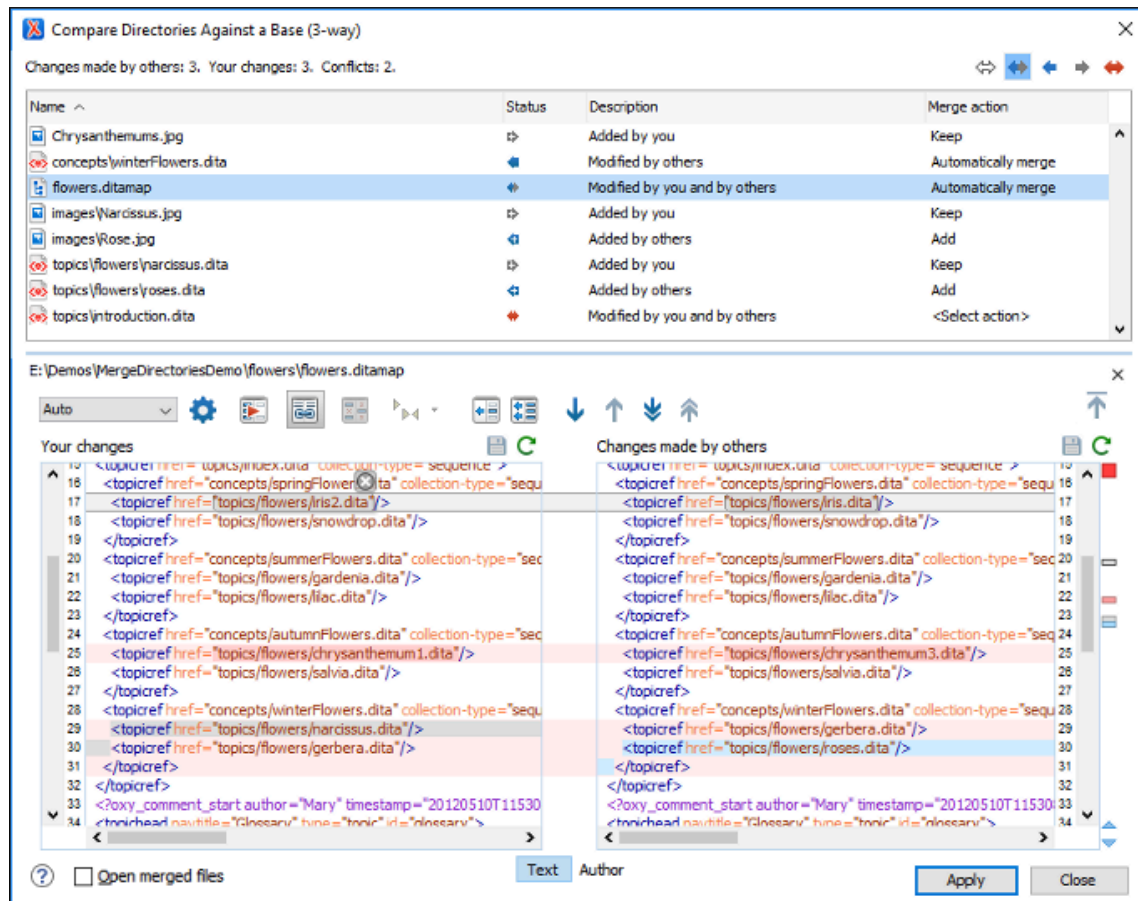
Step Result: This opens a dialog box that allows you to select the 3 file sets that will be used for the comparison.

Figure 174. Compare Directories Against a Base File Set Chooser

2. Select the file sets to be compared:
 - **Base directory** - This is the original (base) file set before any modifications were made by you or others.
 - **Directory with your changes** - This is the file set with changes that you have made. This file set will be displayed in the left panel in the comparison tool.
 - **Directory with changes made by others** - This is the file set with changes made by others that you want to merge with your changes. This file set will be displayed in the right panel in the comparison tool.
3. Click the **Compare** button to compare the file sets and open the comparison and merge tool.
4. Use the features and actions described in the next section to identify and merge the changes.

3-Way Directory Comparison and Merge Tool

Figure 175. Comparison and Merge Tool



The 3-way directory comparison and merge tool includes the following information, features, and actions:

Number of Changes and Conflicts

The first thing you see in the top-left corner of the tool is the grand total of all the changes made by others, changes made by you, and the number of conflicts.

Filter Buttons

In the top-right corner you can use the toggle buttons to filter the list of modifications:

↔ Show all files

Use this button to show all modified and unmodified files, as well as conflicts.

↔ Show only files modified by you and others

Filters the list to show all files that have been modified, including conflicts.

← Show only files modified by others

Filters the list to only show the files that were modified by others.

→ Show only files modified by you

Filters the list to only show the files that were modified by you.

Show only conflicting files

Filters the list to only show files that contain conflicts.

List of Files Panel

This panel shows the list of files in the compared file sets based upon the filter button that is selected. This panel includes the following sortable columns:

- **Name** - The file names.
- **Status** - An icon that represents the file status. Red icons indicate some sort of conflict. Gray icons indicate modifications made by you. Blue icons indicate modifications made by others.
- **Description** - A description of the file status.
- **Merge Action** - This column provides a drop-down menu for each file that allows you to choose some merge actions depending upon its status. A default action is always set to **Automatically merge** the changes made by others with your changes. If there is a conflict, the default is **<Select action>** and you are required to make a selection. Click this column to access the drop-down menu where you can make a selection. The same actions are available in the contextual menu.



Tip:

If the solution proposed in the **Merge Action** column for any particular file is not satisfactory, you can change it directly in that column (even if that file is not selected) without automatically re-triggering the comparison (except for in certain cases where re-triggering the comparison is necessary).

You can click a file to open it in the file comparison panel (the file from your file set is shown in the left panel while the file from the file set with changes made by others is shown in the right panel). For image files, the comparison panel shows a preview of the image. For other binary files, a preview is not available and you will just see its status.

File Comparison Panels

If you click a file in the top panel, the file is opened in this file comparison section. The file from your file set is shown in the left panel and the file from the other file set is shown in the right panel.



Note:

If Oxygen JSON Editor does not recognize the file type, a dialog box will be displayed that allows you to select the type of editor you want it to be associated with for this comparison (if you want Oxygen JSON Editor to remember this association, you can select the **Associate file type with editor** option at the bottom of the dialog box).

This panel includes the following information and toolbar actions:

File Path

The first thing you see in this panel is the file path where merge actions will be applied if you make changes.

✕ Close

Closes the file comparison panel.

Algorithm Drop-down Menu

This drop-down menu allows you to select one of the following diff algorithms to be used for file comparisons:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in **Author** comparison mode.
- **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 \(on page 166\)](#) where you can configure various options.



Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.



Synchronized scrolling

Toggles synchronized scrolling. When toggled on, a selected difference can be seen in both panels.



Ignore Whitespaces

Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces. This option is not available when the file comparison is in **Author** mode.



Tags Display Mode

Allows you to select the amount of markup to be displayed in the **Author** visual mode. You can choose between: **Full Tags with Attributes**, **Full Tags**,

Block Tags,  **Block Tags without Element Names,**  **Inline Tags,**  **Partial Tags,** or  **No Tags.**



Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.



Copy All Changes from Right to Left

Copies all changes from the file in the right panel to the file in the left panel.



Next Block of Changes (Ctrl + Period (Command + Period on macOS))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.



Note:

A change block groups one or more consecutive lines that contain at least one change.



Previous Block of Changes (Ctrl + Comma (Command + Comma on macOS))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.



Next Change (Ctrl + Shift + Period (Command + Shift + Period on macOS))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.



Previous Change (Ctrl + Shift + Comma (Command + Shift + M on macOS))

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.



First Change (Ctrl + B (Command + B on macOS))

Jumps to the first change.

Left-Side File (Your changes)

Above the panel you can see the file path and the following two buttons:



Save

Saves changes made to the file.



Reload

Reloads the file.

Right-Side File (Changes made by others)

Above the panel you can see the file path and the following two buttons:



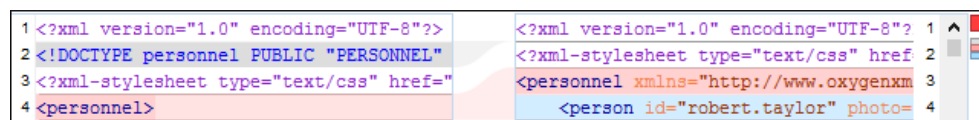
Reload

Reloads the file.

Displaying Changes in the File Comparison Panels

The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes.

Figure 176. File Comparison Panels



The differences are also highlighted in several colors, depending on the type of change, and dynamic lines connect the compared fragments in the middle section between the two panes. The highlighting colors can be customized in the [Files Comparison / Appearance preferences page \(on page 168\)](#), but the default colors and their shades mean the following:

- **Pink** - Identifies modifications on either side.
- **Gray** - Identifies an addition of a node in the left side (your outgoing changes).
- **Blue** - Identifies an addition of a node in the right side (incoming changes).
- **Lighter Shade** - Identifies blocks of changes that can be merged in their entirety.
- **Darker Shade** - Identifies specific changes within the blocks that can be merged more precisely.

Direct Editing Actions in the File Comparison Panels

In addition to selecting merge actions from the drop-down menus in the **Merge Action** column in the top panel, you can also edit the files directly in the left pane (your local changes). The two editors are constantly synchronized and the differences are refreshed when you save the modified document ( **Save** button or **Ctrl+S**) or when you click the  **Perform File Differencing** button.

A variety of actions are available in the contextual menu in both editing panes. The tool also includes some inline actions to help you merge, copy, or remove changes. When you select a change, the following inline action widgets are available, depending on the type of change:



Append right change to left

Copies the content of the selected change from the right side and appends it on the left side.

 Copy change from right to left

Replaces the content of a change in the left side with the content of the change in the right side.

 Remove change

Removes the change from the left side.

Anytime you save manual changes ( **Save** button or **Ctrl+S**), the selection in the **Merge Action** column in the top panel automatically changes to **Use merged** and a copy of the original file is kept so that you can revert to the original file if necessary. To discard your manual changes and revert to your original changes, select a different action in the **Merge Action** drop-down menu.

Open Merged Files

If you select this option, all the files that will be modified by the merge operation will be opened in the editor after the operation is finished.

Applying Changes

When you click the **Apply** button, all the merge actions you have selected and the changes you have made will be processed.

If there are unresolved conflicts (conflicts where no merge action is selected in the **Merge Action** drop-down menu), a dialog box will be displayed that allows you to choose how to solve the conflicts. You can choose between the following:

- **Keep your changes** - If you select this option and then click **Apply**, your local changes will be preserved for the unresolved conflicts.
- **Overwrite your changes** - If you select this option and then click **Apply**, your local changes will be overwritten with the changes made by others, for the unresolved conflicts.
- **Cancel** - You can click the **Cancel** button to go back to the merge tool to resolve the conflicts individually.

Canceling Changes

If you click the **Cancel** button at the bottom of the merge tool, all the changes you made in the tool will be lost.

Related information

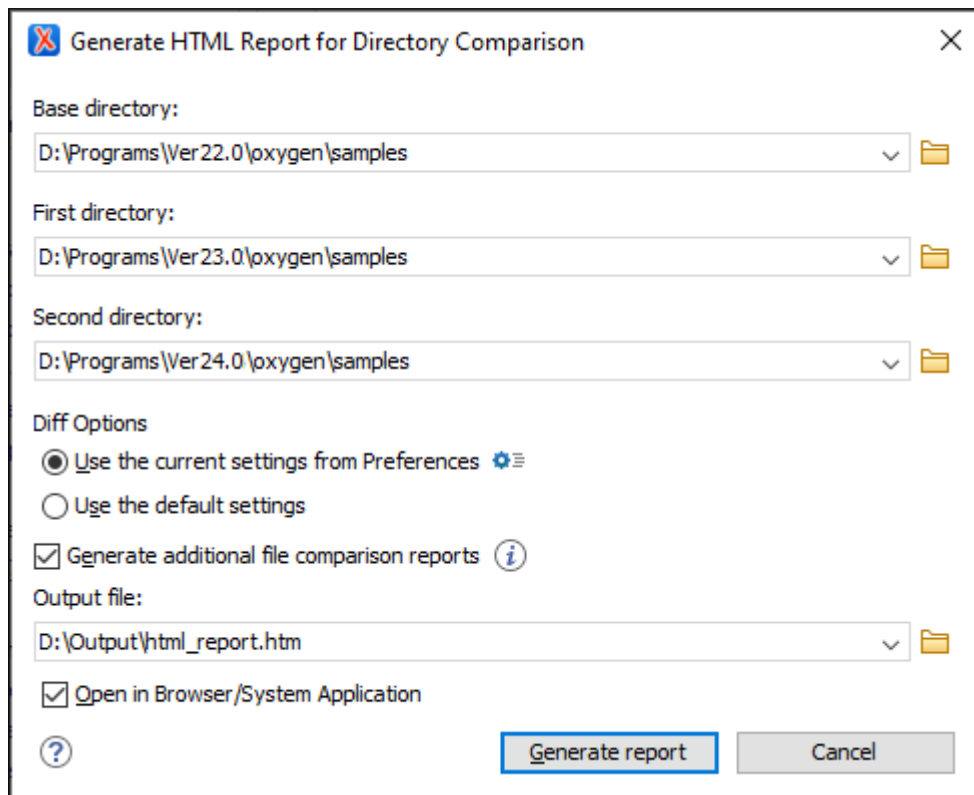
[Compare Directories Tool \(on page 332\)](#)

[Compare Files Tool \(on page 313\)](#)

Generate HTML Report for Directory Comparison

The **Generate HTML report for directory comparison** tool can be used to generate a report in the form of an HTML file that contains the results of a directory comparison (for either 2-way or 3-way comparisons). The **Generate HTML report for directory comparison** action for invoking the tool can be found in the **Tools** menu. It opens a dialog box where you can specify the directories to compare as well as some other options.

Figure 177. Generate HTML Report for Directory Comparison Dialog Box



The **Generate HTML report for directory comparison** dialog box contains the following options:

Base directory

Specifies the path of the base directory that the other two directories will be compared against in a 3-way comparison. This field should be left empty for 2-way comparisons.

First directory

Specifies the path of the first directory to be included in the comparison.

Second directory

Specifies the path of the second directory to be included in the comparison.

Diff options

Specifies which option set to use for generating the comparison report. If you choose **Use the current settings from Preferences**, the options set in the **Directories Comparison** preferences page (*on page 169*) and the *include/exclude filter options* in the **Compare Directories** tool (*on page 335*) are taken into account when generating the comparison result. You can also click the  **Diff options** button to open the **Directories Comparison** preferences page where you can see

or modify the current settings. If you choose **Use the default settings**, the default values for all settings are used.

Generate additional file comparison reports

Generates further comparison reports for all non-binary modified file pairs and provides links to them in the main report (in the middle cells of the results table). [See the example below \(on page 617\)](#). These additional file comparison reports are saved to a directory that will have the same parent directory and the same name as the output file provided, suffixed by **"-OXY-FC-REPORTS"**. The links created in the main report are relative to this directory. If the main HTML report is later copied or moved to another location, to retain full functionality in the browser, the directory with the additional file comparison reports must also be copied/moved to the same location.



Note:

Generating additional file comparison reports could significantly increase the execution time. A progress tracker for the whole operation is available.



Tip:

An XPath expression specified in the **Ignore nodes by XPath** text field within the [Files Comparison preferences page \(on page 166\)](#) is now taken into account if you enable the **Generate additional file comparison reports** option.

Output file

Specifies the path for an output file to save the comparison results file.

Open in Browser/System Application

Opens the comparison results file in the browser or system application that is associated with HTML files.

After clicking the **Generate report** button, a report in the form of an HTML file is generated with details about the comparison results.

Figure 178. HTML Report for Directory Comparison

Differences: 13

Comparison details: ☒ all differences (13) ☐ outgoing (5) ☐ incoming (5) ☐ conflicts (3)

Base folder: D:/Sample1/dita-flowers/flowers-base/

Folder 1: D:/Sample1/dita-flowers/flowers-by-John/

Folder 2: D:/Sample1/dita-flowers/flowers-by-Mary/

File name	Size	Modified		File name	Size	Modified
concepts/autumnFlowers.dita	1151	2021-07-06 01:49:12	* *	concepts/autumnFlowers.dita	1143	2021-07-16 06:52:21
concepts/glossaryGenus.dita	571	2021-07-16 06:54:17	*	concepts/glossaryGenus.dita	577	2021-07-06 01:49:12
concepts/glossaryPanicle.dita	483	2021-07-15 06:53:34	*	concepts/glossaryPanicle.dita	495	2021-07-06 01:49:12
images/Gerbera.jpg	10134	2021-07-06 01:49:12	*	images/Gerbera.jpg	22776	2021-07-16 05:55:25
			+	publishing/flowers/resources/images/flower_logo.png	6178	2021-07-06 01:49:12
			+	publishing/flowers/READ-ME.txt	127	2021-07-06 01:49:12
publishing/flowers/README.txt	127	2021-07-06 01:49:12	-			
tasks/gardenPreparation.dita	2275	2021-07-06 01:49:12	*	tasks/gardenPreparation.dita	2291	2021-07-15 12:21:02
topics/flowers/chrysanthemum.dita	2949	2021-07-15 07:48:25	*	topics/flowers/chrysanthemum.dita	2932	2021-07-06 01:49:12
topics/flowers/gerbera.dita	2432	2021-07-16 06:18:28	* *	topics/flowers/gerbera.dita	2456	2021-07-16 06:25:13
topics/flowers/snowdrop.dita	2883	2021-07-15 13:57:59	*	topics/flowers/snowdrop.dita	2806	2021-07-06 01:49:12
topics/test/		2021-07-15 12:36:56	+			
topics/introduction.dita	770	2021-07-16 06:58:21	* *	topics/introduction.dita	758	2021-07-15 12:12:46

Figure 179. Example of an Additional File Comparison Report

← → ↻ ⓘ File | D:/Output/html_report-OXY-FC-REPORTS/fc-36.html

Differences: 5 difference blocks, 8 differences in total

Comparison details by difference blocks: ☒ all (5) ☐ incoming (3) ☐ outgoing (2)

Base file: D:/Sample1/dita-flowers/flowers-base/topics/flowers/gerbera.dita

File 1: D:/Sample1/dita-flowers/flowers-by-John/ topics/flowers/gerbera.dita		File 2: D:/Sample1/dita-flowers/flowers-by-Mary/ topics/flowers/gerbera.dita
8 is a genus of ornamental plants from the daisy family (Asteraceae). It was named in 9 honor of the German naturalist Traugott Gerber (1710-1743) who travelled extensively in Russia and 10 was a friend of Carl Linnaeus </p>	+ -	8 is a genus of ornamental plants from the sunflower family (Asteraceae). It was named in 9 honor of the German naturalist Traugott Gerber </p>
12 13 <!--Maybe we can add more pictures here....--> 14 15 <p>It has approximately 30 species in the wild, extending to South America, Africa and tropical	+ -	11 <p>It has approximately 30 species in the wild, extending to South America, Africa and tropical
18 also known as Transvaal daisy or Barberton Daisy. 19 Gerbera species bear a large capitulum with striking, two-lipped ray florets in yellow,	- +	14 also known as Transvaal daisy or Barberton Daisy. 15 16 17 Gerbera species bear a large capitulum with striking, two-lipped ray florets in yellow,
25 The domesticated <xref keyref="cultivar" format="dita">cultivars</xref> are mostly a result of a	- +	24 The domesticated <xref format="dita" keyref="cultivar">cultivars</xref> are mostly a result of a

Resources

For more information about how to generate HTML comparison reports, watch our video demonstration:

<https://www.youtube.com/embed/6jPccHKUNNk>

Related information[Compare Directories Tool \(on page 332\)](#)[Compare Directories Against a Base \(3-Way\) Tool \(on page 338\)](#)[Compare Files Tool \(on page 313\)](#)

External Tools

A command-line tool can be started in the Oxygen JSON Editor user interface as if from the command line of the operating system shell. Oxygen JSON Editor offers you the option of integrating such a tool by specifying just the command line for starting the executable file and its working directory. To integrate such a tool, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#) and go to [External Tools \(on page 172\)](#) (or select **Configure** from the **Tools > External Tools** menu).

The [External Tools preferences page \(on page 172\)](#) presents a list of the external tools that have been configured. Once [a tool has been configured \(on page 172\)](#), you can open it by selecting it from the **Tools > External Tools** menu or from the  **External Tools** drop-down menu on the toolbar (the **Tools** toolbar needs to be selected in the [Configure Toolbars dialog box \(on page 222\)](#)). You can also [assign a keyboard shortcut \(on page 174\)](#) to be used to launch the tool.

If the external tool is applied on one of the files opened in Oxygen JSON Editor, the [Save all files before calling external tools option \(on page 135\)](#) (in the **Save** preference page) should be selected so that all edited files are automatically saved when an external tool is applied.

When an external tool is launched, the icon on the toolbar changes to a stop icon () and you can use this button to stop the tool. When the tool has finished running (or you close it), the icon changes back to the original icon ().

**Note:**

Even though you can stop the external tool by invoking the stop action while it is running, that does not necessarily mean it will also stop the processes spawned by that external tool. For instance, if you stop an external tool that runs a batch file, the batch may be stopped but without actually stopping the processes that the batch was running at that time.

Example: Integrating the Ant Tool

This is an example procedure for integrating [the Ant build tool](#) into Oxygen JSON Editor:

1. [Download](#) and [install Apache Ant \(on page 637\)](#) 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**) (*on page 74*) and go to **External Tools** (or select **Configure** from the **Tools > External Tools** menu).
4. Click the **New** button to create a new external tool entry and enter the following information:
 - **Name** - For example, `Ant tool`.
 - **Working directory** - For example, `C:\projects\XMLproject`.
 - **Command line** - For example, `"C:\projects\XMLproject\ant.bat" clean`.
5. Click **OK** to add the new tool to the list of external tools.
6. Run the tool from **Tools > External Tools > Ant tool**. You can see the output in the system console:

```
Started: "C:\projects\XMLproject\ant.bat" clean
Buildfile: build.xml

clean:
[echo] Delete output files.
[delete] Deleting 5 files from C:\projects\XMLproject

BUILD SUCCESSFUL
Total time: 1 second
```

12.

Troubleshooting

This section provides a collection of common performance and other types of problems that might be encountered when using Oxygen JSON Editor, along with their possible solutions.

Performance Problems and Solutions

This section contains solutions for some common performance problems that may appear when running Oxygen JSON Editor.

Display Problems on Linux or Solaris

Problem

I experience display problems (such as screen freezes) on Linux or Solaris.

Cause

This is possibly a rendering problem with the off-screen pixmap support.

Solution

Add the following startup parameter (*on page 211*): **-Dsun.java2d.pmoftscreen=false**.

Too many nested apply-templates calls Error When Running a Transformation

Problem

I'm getting the error message **Too many nested apply-templates calls** when I try to transform my DocBook file to HTML using default Oxygen JSON Editor DocBook to HTML transformation scenario.

Cause

Most likely, this is the result of a masked stack overflow error.

Solution

Try setting a new VM option with the value **-Xss4m**. You can also try to slowly increase this to larger values (e.g. **-Xss5m** or **-Xss6m**). Note that this consumes memory on a per thread basis (Oxygen JSON Editor can have tens of threads), so using a very large value here can backfire and leave Oxygen JSON Editor without memory.

Related Information:

[Setting a Java Virtual Machine Parameter when Launching Oxygen JSON Editor \(on page 211\)](#)

Performance Issues with Large Documents

Problem

The performance of the application slows down considerably over time when working with large documents.

Cause

A possible cause is that the application needs more memory to run properly.

Solutions

- You can increase the maximum amount of memory available to Oxygen JSON Editor by [setting the `-Xmx` parameter in a configuration file \(on page 211\)](#) that is specific to the platform that runs the application.

**Attention:**

The maximum amount of memory should be less than 75% of the physical amount of memory available on the machine. Otherwise, the operating system and other applications will have no memory available.

- When installed on a multiple user environment, each instance of Oxygen JSON Editor will be allocated the amount stipulated in the memory value. To avoid degrading the general performance of the host system, ensure that the amount of memory available is optimally apportioned for each of the expected instances.

**Note:**

When starting Oxygen JSON Editor from the icon created on the Start menu or Desktop in Windows (or from the shortcut created on the Linux desktop), the default maximum memory available to the application is set to 40% of the amount of physical RAM (but not more than 1 GB for 32-bit distributions or 4 GB for 64-bit distributions).

When starting Oxygen JSON Editor from a command-line script, the default maximum memory is 1 GB for 32-bit distributions or 4 GB for 64-bit distributions.

Performance Issues when Using Oxygen JSON Editor with Remote Desktop

Problem

When trying to run Oxygen JSON Editor in a Remote Desktop Protocol (RDP) environment, the performance is slow and choppy.

Cause

Running a standalone version of Oxygen JSON Editor over a slow RDP connection may result in performance issues.

Solution

As a workaround, you try to run Oxygen JSON Editor as an Eclipse plugin when working with a slow RDP connection.

Misc Problems and Solutions

This chapter presents common problems that may appear when running the application along with solutions for these problems.

Address Family Not Supported by Protocol Family

Problem

I have experienced the following error: *"Address Family Not Supported by Protocol Family; Connect"*. How do I solve it?

Cause

This seems to be an IPv6 connectivity problem. By default, the Java runtime used by Oxygen JSON Editor prefers to create connections via IPv6, if the support is available. However, even though it is available in appearance, IPv6 sometimes happens to be configured incorrectly on some systems.

Solution

A quick solution for this problem is to set the `java.net.preferIPv4Stack` Java property to `true` (`java.net.preferIPv4Stack=true`), by following this procedure:

1. Create a file named `custom_commons.vmoptions` and on a single line, add `-Djava.net.preferIPv4Stack=true`. Then save the file and copy it to the Oxygen JSON Editor installation folder (may need admin access).
2. Restart Oxygen JSON Editor.
3. Make sure the procedure was successful by going to **Help > About > System properties** and check that the value of the `java.net.preferIPv4Stack` property is `true`.

Application Reports Errors During Startup After Installing a New Version

Problem

Sometimes, after installing a new version of Oxygen JSON Editor, various errors are reported when the application starts.

Cause

This problem may occur if you install the application in a folder where an older version of the application was previously installed. Especially on macOS, there is a possibility for older resources and libraries from the previous application to remain in the installation folder and break the functionality of the newer version of the application.

Solution

Close the application and completely [uninstall it \(on page 72\)](#), then install it again. The user-specific settings are saved in a separate folder in the user home directory so they will not be lost.

- On macOS, you can move the entire application installation folder to the **Trash**, then re-install.
- On Linux and Windows, you can [uninstall using the facilities provided by the installer \(on page 72\)](#), then re-install.

If you intentionally want to load extra Java libraries with Oxygen JSON Editor, you have the following choices:

- If the libraries are necessary for XSLT transformations, each XSLT transformation scenario has an **Extensions** button that allows you to reference the libraries directly from the transformation scenario.
- If the libraries are necessary for database connections, you can configure them when you define the data sources.
- You can [add a plugin](#) in Oxygen JSON Editor that contributes libraries to the global libraries list. The plugin can be distributed as an add-on. An example of such a plugin can be found here: <https://github.com/oxygenxml/oxygenxml.xlsx.import>.
- In the Oxygen JSON Editor **lib** folder, there is a file called **libraries.list**. You can edit that file and add the names of the extra libraries present in the folder. You can also choose to delete that **libraries.list** completely if you want to inhibit the libraries checking completely.

Application Takes Several Minutes to Start

Problem

Oxygen JSON Editor seems to take an abnormally long amount of time to start.

Cause

Some anti-virus software can cause Java applications, such as Oxygen JSON Editor, to start very slowly due to scanning compressed archives (such as the *JAR* libraries that all Java applications use).

Solution

A possible solution is to add the Oxygen JSON Editor folder to the list of exceptions in the anti-virus software settings.

Archive Distribution Fails to Run on macOS 10.12 (Sierra)

Problem

For versions prior to 18.1, the classic archive distributions of Oxygen JSON Editor (`.zip` or `.tar.gz`) fail to run on macOS 10.12 (Sierra).

Cause

This happens because the archives get quarantined and macOS 10.12 (Sierra) treats quarantined apps differently than older versions and isolates them from their parent folder at launch. If Oxygen JSON Editor was already installed when you upgraded to macOS 10.12 (Sierra), there are no problems.

Solution

If Oxygen JSON Editor was not already installed, or you need to reinstall an older version (prior to version 18.1), the quarantine flag must be removed for the entire content of the Oxygen JSON Editor installation directory or the individual applications. To resolve this issue, follow these steps:

1. Open a *Terminal* window and change the directory to the folder where Oxygen JSON Editor is located.
2. Run the following command:

```
xattr -dr com.apple.quarantine oxygen/
```

where "oxygen" is the actual name of the Oxygen JSON Editor directory.

If Oxygen JSON Editor is in a location that requires administrator privileges for write access, you need to run the command from an administrator account and prefix the command with `sudo`. You will then be prompted to enter your password.

Blank Window is Shown When Starting the App Over an RDP Connection on Linux

Problem

When starting Oxygen JSON Editor or its installer on *Linux*, a blank window is displayed when started over an RDP connection.

Cause

Oxygen JSON Editor and its installer are Java Swing apps that require a 24 bit color depth from the X server.

Solution

1. If you are using *xrdp*, find the `/etc/xrdp/xrdp.ini` file.
2. Uncomment the `xserverbpp=24` line.
3. Save your files and close all the apps (the subsequent step will terminate your remote session so you could lose your progress if you do not save your files first).

4. Restart the *xrdp* service:

```
sudo systemctl restart xrdp.service
```

**Note:**

Alternatively, you can try setting `max_bpp=24` in the same `/etc/xrdp/xrdp.ini` file.

Cannot Open Files from Desktop/Downloads/OneDrive on macOS

Problem

When using Oxygen JSON Editor on *macOS*, the application cannot open files from Desktop/Downloads/OneDrive.

Cause

Sometimes, macOS shows a popup about allowing the application access to some special folders (e.g. Downloads or Desktop), and unless you explicitly agree, it will leave it unchecked (the app does not allowed access). The popup can go unnoticed and disappears after a while, so it is easy to overlook it and the application will not have access to that folder.

Solution

Go to the macOS **System preferences > Security & Privacy > Privacy tab > Files and Folders**. You should find *Oxygen* in that list and the folders you were prompted to access. Check the box for the folder (i.e. Downloads or Desktop), if there is one unchecked.

**Note:**

If that does not work, look at "Full Disk Access" from the same Privacy tab. Add *Oxygen* there so that it has full access. However, only use this method as a last resort.

Cannot Uninstall Oxygen JSON Editor in Windows

Problem

When I try to uninstall Oxygen JSON Editor in Windows, I get an error that says it cannot find the `files.log` file.

Cause

The *install4j* installer that is used by Oxygen JSON Editor creates the `files.log` file during the installation process. If you cannot uninstall the product, then most likely something went wrong with this file during the installation process.

Solution

To solve this, simply reinstall the software in the same directory as the current installation. This will automatically uninstall the old version or overwrite it with a clean install. You should then be able to uninstall this new installation.

Compatibility Issue Between Java and Certain Graphics Card Drivers

Problem

Under certain settings, a compatibility issues can appear between Java and some graphics card drivers, which results in the text from the editor (in **Author** or **Text** mode) being displayed garbled.

Solution

If you encounter this problem, update your graphics card driver. Another possible workaround is, [open the Preferences dialog box \(Options > Preferences\) \(on page 74\)](#), go to **Appearance > Fonts**, and set the value of the **Text antialiasing** option [\(on page 84\)](#) to ON.



Note:

If this workaround does not resolve the problem, set the **Text antialiasing** option [\(on page 84\)](#) to other values.

Crash at Startup on Windows with an Error About the nvoglv32.dll File

Problem

I try to start Oxygen JSON Editor on Windows but it crashed with an error message about “*Fault Module Name: nvoglv32.dll*”. What is the problem?

Cause

It is most likely an OpenGL driver issue.

Solution

This can be avoided by creating an empty file called **opengl32.dll** in the Oxygen JSON Editor install folder (if you start Oxygen JSON Editor with the shortcut created by the installer on the Start menu or on Desktop) or in the subfolder **bin** of the home folder of the Java virtual machine that runs Oxygen JSON Editor (if you start Oxygen JSON Editor with the **oxygen.bat** script). The home folder of the Java virtual machine that runs Oxygen JSON Editor is the value of the `java.home` property that is available in the **System properties** tab of the **About** dialog box (**Help > About**).

Damaged File Associations on macOS

Problem

After upgrading macOS and Oxygen JSON Editor, it is no longer associated to the appropriate file types (such as XML, XSL, XSD). How can I re-create the file associations?

Cause

The upgrade damaged the file associations in the LaunchService Database on your macOS machine.

Solution

You can rebuild the LaunchService Database with the following procedure. This will reset all file associations and rescan the entire file system searching for applications that declare file associations and collect them in a database used by Finder.

1. Find all the Oxygen JSON Editor installations on your hard drive.
2. Delete them by dragging them to the Trash.
3. Clear the Trash.
4. Unpack the Oxygen JSON Editor installation kit on your desktop.
5. Copy the contents of the archive into the folder **/Applications/Oxygen**.
6. Run the following command in a Terminal:

```
/System/Library/Frameworks/CoreServices.framework/Versions/A/Frameworks/  
LaunchServices.framework/Versions/A/Support/lsregister -kill -r -domain local -domain system  
-domain user
```

7. Restart Finder with the following command:

```
killall Finder
```

8. Create an XML or XSD file on your desktop. It should have the Oxygen JSON Editor icon.
9. Double-click the file.
10. Accept the confirmation.

Result: When you start Oxygen JSON Editor, the file associations should work correctly.

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

Problem

What details should I add to my request for technical support on the online form in the product website?

Solution

When completing a request for Technical Support using the online form, include as many details as possible about your problem. For problems where a simple explanation may not be enough for the Technical Support

team to reproduce or address the issue (such as server connection errors, unexpected delays while editing a document, an application crash, etc.), you should generate log files and attach them 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 the Oxygen JSON Editor log files, follow these steps:

1. Create a text file called **logback.xml** in the application installation folder, with the following content:

```
<configuration>
  <appender name="R2" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <file>${user.home}/Desktop/oxygenLog/oxygen.log</file>
    <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
      <fileNamePattern>${user.home}/Desktop/oxygenLog/oxygen%i.log.gz</fileNamePattern>
      <minIndex>1</minIndex>
      <maxIndex>20</maxIndex>
    </rollingPolicy>
    <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
      <maxFileSize>12MB</maxFileSize>
    </triggeringPolicy>
    <encoder>
      <pattern>%r %marker %p [ %t ] %c - %m%n</pattern>
    </encoder>
  </appender>

  <root level="debug">
    <appender-ref ref="R2" />
  </root>
</configuration>
```

2. Restart the application.
3. Reproduce the error.
4. Close the application.
5. Delete the **logback.xml** 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, do not forget to delete the **logback.xml** file when you are done with the procedure.

Result: The resulting log files are named **oxygen.log** and **oxygen#.log.gz** (for example, **oxygen.log**, **oxygen1.log.gz**, **oxygen2.log.gz**, etc.) and are located in the **Desktop\oxygenLog** folder.

Dialog Boxes Cannot Be Resized on Mac

Problem

When using Oxygen JSON Editor on *macOS Big Sur*, dialog boxes (for example the **Find/Replace** or **Preferences** dialog box) cannot be resized.

Cause

This is caused by an issue with resizing dialog boxes in Oxygen JSON Editor on *macOS Big Sur* (and possibly later versions) causing crashes if the main application is in full screen mode.

Solution

Until this limitation is resolved in a future Oxygen JSON Editor version, dialog boxes cannot be maximized or resized on *macOS Big Sur*.

Fonts Installed in Windows Do Not Appear in Fonts Preferences Page

Problem

I installed a font in Windows and it does not appear in the list of available fonts in the [Preferences > Fonts](#) page ([on page 83](#)) (in the **Editor** section).

Cause

Oxygen JSON Editor is a Java application and Java does not detect fonts that are installed at the user level. This most likely occurred because you installed the font via the **Install for me** option in Windows.

Solution

Reinstall the font using the **Install for all users** option in Windows. You will need Administrator privileges to access this option.

Handshake Failure Error When Accessing an HTTPS (SSL) Resource

Problem

When attempting to access an HTTPS (SSL) resource, I receive a **handshake_failure** error.

Cause

The issue is most likely due to the limitation of Java cryptography capabilities.

Solution

A solution might be to download and deploy [Java Cryptography Extension \(JCE\) Unlimited Strength Jurisdiction Policy Files 8 \(for Java 11\)](#).

**Warning:**

It is possible that this may not be legal in your country. Be advised that you bare legal responsibility for activating unlimited strength Java cryptography capabilities, so you must have the legal right to use such cryptography (consult your export/import control counsel or attorney to determine the exact requirements for your jurisdiction).

To deploy it in Oxygen JSON Editor, unpack the downloaded zip archive and move the two jar files (**local_policy.jar** and **US_export_policy.jar**) from **UnlimitedJCEPolicyJDK8** to the following directory, overwriting existing files:

- **Windows** - `[OXYGEN_INSTALL_DIR]/jre/lib/security`
- **Linux** - `[OXYGEN_INSTALL_DIR]/jre/lib/security`
- **macOS** - `[OXYGEN_INSTALL_DIR]/jre.bundle/Contents/Home/jre/lib/security`

Hunspell Spell Checker is Unusable on Your Platform Error

Problem

When trying to use the **Check Spelling** option, I receive the error **Hunspell spell checker is unusable on your platform. It has crashed the application in a previous session.**

Cause

There are instances where Oxygen JSON Editor determines that an internal component (such as the spell checker) has crashed the application and disables that component from running in the future (to prevent a possible future crash).

Solution

To re-enable the spell checker component, follow these steps:

1. Close Oxygen JSON Editor.
2. Open the `%APPDATA%\com.oxygenxml` folder and look for a file called something like **HunspellCrashGuard*.txt**. Delete that file.
3. Restart Oxygen JSON Editor.

High Resolution Scaling Issues

Problem

I encounter scaling detection issues in a high resolution display (for example, some GUI components are too small).

Cause

This sometimes happens when using multiple displays with different resolutions because the application cannot detect the correct scaling setting.

Solution

You can use the `sun.java2d.uiScale` Java system property to instruct Java to use a particular scaling factor:

```
-Dsun.java2d.uiScale=1.5
```

High Resolution Scaling Issues on Linux

Problem

On Linux bundled with Oracle OpenJDK 11, Oxygen JSON Editor does not automatically scale high-resolution images when using the system's scaling settings.

Cause

This happens because Java 11 does not detect the system scaling setting for HiDPI displays on Linux operating system.

Solution

In the Oxygen JSON Editor installation folder, create a new file named **custom_commons.vmoptions**. Inside the file, manually add `-Dsun.java2d.uiScale=2`. This command instructs Java to use 2x (200%) scaling.

Java Virtual Machine (JVM) Crash on macOS

Problem

Oxygen JSON Editor crashed the Java virtual machine or it could not start up on my macOS computer due to a JVM crash.

Cause

Usually it is an incompatibility between the JVM and a native library of the host system. More details are available in the crash log file generated by macOS and reported in the crash error message.

Solution

If this happens, it is best to send a copy of the logs via email to support@oxygenxml.com. Usually, the operating system will offer a prompt that allows you to see the logs. If not, you should be able to find the logs in the Console app (**Applications > Utilities**, under `~/Library/Logs/DiagnosticReports`). They are usually named `JavaApplicationStub*.crash/.hang`.

Keyboard Language Resets to Default on Windows

Problem

In Windows, I have set a specific language for my keyboard and while using Oxygen JSON Editor, it keeps getting reset to the default language.

Cause

When multiple languages are enabled in Windows, the default shortcut key combination for switching the input language is **Left Alt + Shift**. Trying to use various shortcuts in Oxygen JSON Editor that involves combinations of those two keys is probably resetting your input language to the default setting if you press those two keys without a third combination.

Solution

You can change the Windows shortcut keys that are assigned to the input language by going to the control panel and searching for the **Switch input languages** option. For example, in Windows, go to **Control Panel > Language > Advanced Setting**. In the **Switching input methods** section, click on **Change language bar hot keys**. Click the **Change Key Sequence** button. This opens a dialog box that allows you to switch the shortcut keys for the input language or keyboard layout.

Keyboard Shortcuts Do Not Work At All

Problem

The keyboard shortcuts listed in the [Menu Shortcut Keys preferences page \(on page 174\)](#) do not work.

Cause

Usually this happens when a special keyboard layout is set (in the operating system) that generates non-standard characters. For example, an extended Greek layout will generate special characters that are not present in the default Greek layout. This causes the Java virtual machine that runs the application to receive unexpected key codes.

Solution

Reset the keyboard layout to the standard layout for your particular language.

Keyboard Shortcuts Result in Unexpected Behavior

Problem

In some rare cases, using certain keyboard shortcuts listed in the [Menu Shortcut Keys preferences page \(on page 174\)](#) result in something different than what is listed in that preferences page.

Cause

This is usually caused by the operating system intercepting the keyboard shortcut. For example, certain applications or hardware drivers intercept certain keyboard shortcuts by default. Another example is if you have multiple input sources configured, the operating system might intercept shortcuts if they match the ones used to change between the input sources.

Solution

Assign a different keyboard shortcut for the particular action in the [Menu Shortcut Keys preferences page \(on page 174\)](#) or refer to documentation for your operating system or hardware equipment to see if there are options to change the behavior.

Mac Touch Bar Function Keys Do Not Work

Problem

I am using a Mac that has a Touch Bar but its function keys do not work in Oxygen JSON Editor.

Causes

By default, the Touch Bar function keys are not enabled for Oxygen JSON Editor.

Solution

To enable the Touch Bar function keys for Oxygen JSON Editor, follow these steps:

1. Go to **System Preferences** and select **Keyboard**.
2. Click **Shortcuts**.
3. From the left sidebar, select **Function Keys**.
4. Click the **+** symbol, select **Oxygen** from the list of apps, and click **Add**.

Server Signature Mismatch Error

Problem

I receive an error indicating that *the current license was already activated on a License Server* or that the *License Server's Signature does not match*.

During the license activation process, the license key becomes bound to a particular license server deployment. This means that a code that uniquely identifies your license server deployment (called *Server Signature*) is sent to the **Oxygen** servers, which in turn will sign the license key. The *Server Signature* is computed from the list of network interfaces of the server where you deployed the license.

When starting the license server, if you receive an error stating that your *Server Signature* does not match, there are several possible causes:

Possible Cause 1

The license key was moved to a new server that hosts your license server.

Solution

Revert to your previous configuration.

Possible Cause 2

A new network interface was changed, added, or activated in the server that hosts your license server.



Note:

A specific example of when this could happen is if the Bluetooth or the WiFi module is activated/deactivated.

Solution

If reverting is not possible, contact the [Oxygen support team](#).

Possible Cause 3

The license server was restarted from a different location as the previous restart. For example, some server configurations will have the Apache Tomcat server installed in a versioned folder (`/usr/local/apache-tomcat-V.V.V`) with a symbolic link to the typical folder (`/usr/local/tomcat`). The server can be restarted from either location, but it is recommended to always restart from the typical folder (`/usr/local/tomcat`) and always restart from the same location.

Solution

The server simply needs to always be restarted from the same location.

Out Of Memory Error When Opening Large Documents

Problem

I am trying to open a file larger than 100 MB in Oxygen JSON Editor, but it runs out of memory (`OutOfMemoryError`).

Solution

You should make sure that the value of the [Optimize loading in the Text edit mode for files over](#) option ([on page 133](#)) is less than the size of your document. This 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 JSON Editor](#) ([on page 621](#)).

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 \(on page 117\)](#). 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 \(on page 579\)](#).

Scroll Function of my Notebook Trackpad is Not Working

Problem

I got a new notebook (Lenovo Thinkpad™ with Windows) and noticed that the scroll function of my trackpad is not working in Oxygen JSON Editor.

Cause

It is most likely a problem with the Synaptics™ trackpads.

Solution

Try adding the following lines to the `C:\Program Files\Synaptics\SynTP\TP4table.dat` file (depending on your version of Oxygen JSON Editor). For example:

```
* , *, oxygen20.1.exe, *, *, *, WheelStd, 1, 9
* , *, oxygenAuthor20.1.exe, *, *, *, WheelStd, 1, 9
* , *, oxygenDeveloper20.1.exe, *, *, *, WheelStd, 1, 9
* , *, syncroSVNClient.exe, *, *, *, WheelStd, 1, 9
* , *, diffDirs.exe, *, *, *, WheelStd, 1, 9
* , *, diffFiles.exe, *, *, *, WheelStd, 1, 9
```

Special Characters are Replaced with a Square

Problem

My file was created with another application and it contains special characters (such as é, ©, ®, etc.) but Oxygen JSON Editor displays a square for these characters.

Solution

You must set a font that is able to render the special characters in the **Font preferences (on page 83)**. If it is a text file, you must also set [the encoding used for non XML files \(on page 114\)](#). If you want to set a font that is installed on your computer but that font is not accessible in the **Font** preferences, this 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**.

Text on Buttons and Labels is Invisible for Linux Installer

Problem

After starting the Linux installer, the text on buttons and labels is invisible.

Cause

This seems to be a font issue between Oracle Java SE 8 (bundled with the installer) and Fedora/Gnome.

Solution

There are two possible workarounds:

- Run the installer with the default (non-native) Java L&F by using the `-Dinstall4j.nolaf=true` argument. For example:

```
oxygen-64bit-openjdk.sh -Dinstall4j.nolaf=true
```

- Run the installer in console mode using the `-c` argument. For example:

```
oxygen-64bit-openjdk.sh -c
```

Text Rendering Issues on macOS

Problem

On macOS, I sometimes encounter issues where text is not rendered properly. For example, when tags are displayed in **Author** mode, sometimes the tag icon is rendered over the top of text (hiding the text) and sometimes text flows outside of code blocks.

Cause

This is an uncommon error that cannot be fixed in current versions.

Solution

Open the **Preferences** dialog box (**Options > Preferences**) (*on page 74*), go to **Editor > Edit modes > Author**, and deselect the **Fast text layout** option (*on page 121*).

13.

Glossary

Active Cell

Active cell refers to the selected cell where data is entered when you begin typing. Only one cell is active at a time. The *active cell* is bounded by a heavy border.

Alternate CSS Style

The **Alternate CSS Style** refers to the choices in the bottom half of **Styles** drop-down menu (on the toolbar) that makes it easy to apply style changes to your documents as they appear in **Author** mode and the output without having to edit the CSS stylesheets. By default, the *alternate styles* are applied like layers, they are merged sequentially with the *main CSS style* (on page 639), and you can activate any number of them. However, if you deselect the **Enable multiple selection of alternate CSSs option** (on page 94) in the **CSS** subtab of the **Document Type** configuration dialog box (on page 89), the *alternate styles* are treated like *main CSS styles* (on page 639) and you can only select one at a time.

Apache Ant

Apache Ant (Another Neat Tool) is a software tool for automating software build processes.

Block Element

A **block element** is intended to be visually separated from its siblings, usually vertically. For instance, paragraphs and list items are *block elements*. It is distinct from a *inline element*, which has no such separation.

Bookmap

A **bookmap** is a specialized *DITA map* used for creating books. A *bookmap* supports book divisions such as chapters and book lists such as indexes.

Canonicalize

To **canonicalize** something means to convert it to a standard format that everyone generally uses. When using the term with regard to XML, it refers to the process of converting data that has more than one possible representations into a standardization that conforms to the specification of an XML document or document subset. It is helpful for applications that require the ability to test whether or not the content of an XML document or subset has been changed.

Content Completion Assistant

The **Content Completion Assistant** refers to a very helpful mechanism in Oxygen JSON Editor that offers a list of proposed items that could be inserted at the current location, depending on the current context, editing mode, and type of document. It also tries to determine the most logical choice in the current editing context and displays that proposal at the beginning of the list.

For more information about this feature and how to invoke it, see the following:

Dockable

A **Dockable** window is one that can be moved and resized, and either floated or pinned to a location, allowing you to configure the workspace according to your preferences.

Document Type Association

In general terms, a **Document Type Association** is a set of rules that associate a document type with a [framework \(on page 639\)](#). In Oxygen JSON Editor, **Document Type Association** also specifically refers to a [preferences page \(on page 87\)](#) where you can create new custom *frameworks* or edit existing ones. Note that *frameworks* (document types) that come built-in with Oxygen JSON Editor are read-only, but you can **Extend** ([on page 88](#)) or **Duplicate** ([on page 87](#)) them to configure them as custom *frameworks*.

DITA Map

A **DITA map** is a component of the DITA [framework \(on page 639\)](#) that provides the means for a hierarchical collection of DITA topics that can be processed to form an output. Maps do not contain the content of topics, but only references to them. These are known as topic references. Usually, the maps are saved on disk or in a CMS with the extension **.ditamap**.

Maps can also contain relationship tables that establish relationships between the topics contained within the map. Relationship tables are also used to generate links in your published document.

You can use your map or [bookmap \(on page 637\)](#) to generate a deliverable using an output type such as XHTML, PDF, HTML Help, or Eclipse Help.

Foldable Element

A **foldable element** refers to elements that can be collapsed and expanded in Oxygen JSON Editor.

Foldable elements are marked with a small triangle (▾ / ▸) on the left side of the editor panel and you can use that triangle to quickly collapse or expand them. This feature is helpful when you are working with large documents and you want to temporarily hide blocks of content. You can right-click the triangle to access additional collapse and expand actions (**Collapse Other Folds**, **Collapse Child Folds**, **Expand Child Folds**, **Expand All**).

Framework

A **framework** refers to a package that contains resources and configuration information to provide ready-to-use support for a vocabulary or document type. A *framework* is associated to a document type according to a set of rules. It also includes a variety of settings that improve editing capabilities for its particular file type.

Global Options

Global Options refers to the [storage option \(on page 187\)](#) in the Oxygen JSON Editor preference pages (**Options > Preferences**). If you select **Global Options (on page 188)**, the options in that particular preferences page are stored locally on your computer and are not accessible to other users (unless you [export them into an XML options file \(on page 189\)](#) that can then be shared).

IDML

IDML is an abbreviation for Adobe InDesign Markup files.

Inline Element

An **inline element** is intended to be displayed in the same line of text as its siblings or the surrounding text. For instance, strong and emphasis in HTML are *inline elements*. It is distinct from a [block element](#), which is visually separated from its siblings.

Java Archive

Java Archive (JAR) is an archive file format. JAR files are built on the ZIP file format and have the **.jar** file extension. Computer users can create or extract JAR files using the `jar` command or an archive tool.

Keystore

A **Keystore** is an encrypted file that contains private keys and certificates. There are two types of *keystores* that are supported in Oxygen JSON Editor:

- **Java Key Store (JKS)**
- **Public-Key Cryptography Standards version 12 (PKCS-12)**

Main CSS Style

The **Main CSS Style** refers to the selection in the top half of the **Styles** drop-down menu (on the toolbar) that makes it easy to quickly change the look of your documents as they appear in **Author** mode and the output without having to edit the CSS stylesheets. The *main CSS* applies to the whole document and you can also select one or more [alternate styles \(on page 637\)](#) (listed in the bottom half of the drop-down menu) that behave like layers and are merged sequentially with the *main CSS style*.

Plugin

In Oxygen JSON Editor, a **plugin** is a component that adds extended functionality using a series of extension points and can be installed as an *add-on*. For more information, along with a full list of *add-ons* that are officially supported for Oxygen JSON Editor, see [Oxygen XML Add-on Repositories](#).

Pretty-Print

Pretty-print refers to formatting and indenting the source code in **Text** mode to make the content easier to view and analyze. The formatting actions that are available in Oxygen JSON Editor include:

-  **Format and Indent Element** - Available in the **Source** submenu of the contextual menu for the current element.
-  **Format and Indent** - Available on the toolbar for the entire current document.
-  **Format and Indent Files** - Available in the contextual menu of the [view \(on page 251\)](#) for one or more selected files.

Project Options

Project Options refers to the [storage option \(on page 187\)](#) in the Oxygen JSON Editor preference pages (**Options > Preferences**). If you select **Project Options (on page 188)**, the options in that particular preferences page are stored at project level in the project file (**.xpr**), which can easily be [shared with other users \(on page 188\)](#).

Quick Assist

The **Quick Assist** feature gives you easy access to some of the most commonly used actions for the specific type of document you are editing. If one or more actions are available in the current context, they are accessible via a yellow bulb help (💡) placed at the current line in the stripe on the left side of the editor in **Text** mode. You can also invoke the quick assist menu by using the (on macOS) keyboard shortcuts.

Working Set

A **Working Set** refers to a set of files that will be used for the scope of search and refactoring operations. Many of the search and refactoring wizards include a step where you can specify the scope for the operation and you can choose one or more *working sets* to restrict the scope to that specified set of files.

XML Catalog

An **XML Catalog** maps a system ID or a URI reference for a resource (stored either remotely or locally) to a local copy of the same resource. Whenever XML processing relies on external resources (such as referenced schemas and stylesheets), the use of an *XML Catalog* becomes a necessity when Internet access is not available or the connection is slow.

Oxygen JSON Editor includes default global catalogs as well as default catalogs for each of the built-in *frameworks (on page 639)*, and you can also create your own. Oxygen JSON Editor uses these *XML Catalogs* to resolve references for document validation and transformations.

Index

A

- Add-on
 - 640
- Add-on preferences
 - 86
- Add-ons
 - 486, 640
 - Batch converter
 - 500
 - Emmett
 - 532
 - Git support
 - 486
 - Terminology checker
 - 507
 - Vale linter
 - 516
- Addons
 - 486
- Annotation preferences
 - 143
- Appearance preferences
 - 79
- Archive preferences
 - 170
- Archives
 - 480
 - Browse
 - 480
 - EPUB
 - 483
 - File browser
 - 480
 - Migrate OOXML to DITA
 - 485
 - Migrate OOXML to TEI
 - 485
 - Modify
 - 480
 - ODF
 - 483
 - OOXML
 - 483
- Associate JSON schema in a framework configuration
 - 377
- Associate schema directly in JAML documents
 - 428
- Associate schema directly in JSON documents
 - 376
- Associate schema through a validation scenario
 - 373
- Associate schema to a JSON document
 - 373
- AsyncAPI documents
 - Content completion
 - 472
 - Editing features
 - 472
 - Validation
 - 472
- Author Action dialog box
 - 96
- Author mode preferences
 - 120
- Auto recovery
 - 235
- AutoCorrect
 - 300
 - Add dictionaries
 - 302
- AutoCorrect preferences
 - 128
- Automatic Spell Check
 - 297
- Automatic validation in JSON
 - 364
- Automatically correct misspelled words
 - 300
- Available memory

215

B

Backup file

235

Batch converter add-on

500

C

Change file permissions on remote FTP server

240

Change language for interface

210

Character Map dialog box

306

Check Spelling

288

Check Spelling in multiple Files

299

Checking Well-Formedness in JSON

363

Close file

236

Code template preferences

144

Color preferences

81

Common problems and solutions

622

Compare Directories Against a Base tool

338, 607

Compare Directories tool

332, 602

Compare images

337, 607

Menus

336, 605

Toolbar

334, 604

Compare Files tool

313, 582

Command-line arguments

320, 590

Integrate with Git

321, 591

Integrate with Sourcetree

323, 592

Menus

327, 597

Toolbar

324, 594

Comparing files and directories

312

Compile LESS to CSS

358

Configuring options

186

Configuring Oxygen

74

Configuring Toolbars

194, 222

content

517, 528

Content completion in HTML

446

Content completion in JSON

371

Content completion in YAML

429

Content completion preferences

139

Contextual actions in YAML documents

434

Contextual menu actions in JSON

388

Contextual menu of current editor tab

245

Convert JSON to XML

381, 556

Convert JSON to YAML

387, 433, 554

Convert XML to JSON

384, 559

Convert XSD to JSON Schema

536, 562

Convert YAML to JSON

- 387, 433, 555
- Create JSON schema from learned document structure
 - 377
- Creating Markdown documents
 - 450
- Creating new documents
 - 224
- CSS stylesheets
 - Content completion
 - 353
 - Custom CSS properties
 - 353
 - Editing features
 - 352
 - Folding
 - 355
 - Format and indent (pretty print)
 - 355
 - Minifying
 - 355
 - Outline view
 - 354
 - Syntax highlighting
 - 354
 - Validation
 - 352
- CSS validation preferences
 - 160
- Current license was already activated
 - 633
- Cursor navigation preferences
 - 124
- Custom editor variables
 - 180
- Custom grammar checker
 - 507
- Custom system properties
 - 205
- Custom validation engine preferences
 - 153
- Customize document templates

- 230
- D**
- Design editing mode
 - Navigation
 - 398
- Design mode preferences
 - 131
- Diff Directories tool
 - 332, 602
- Diff Files tool
 - 313, 582
- Diff tool
 - 313, 582
- Directory comparison appearance preferences
 - 170
- Directory comparison preferences
 - 169
- Document template preferences
 - 114
- Document templates
 - Creating
 - 229
 - Customizing
 - 230
 - Sharing
 - 234
- Document type association preferences
 - 87
- Document type configuration dialog box
 - 89
 - Association rules tab
 - 90
 - Author tab
 - 94
 - Actions subtab
 - 95
 - Content Completion subtab
 - 107
 - Contextual Menu subtab
 - 105
 - CSS subtab
 - 94

- Menu subtab
 - 104
- Toolbar subtab
 - 107
- Catalogs tab
 - 111
- Classpath tab
 - 93
- Extensions tab
 - 113
- Schema tab
 - 92
- Templates tab
 - 110
- Validation tab
 - 112
- Document Types
 - 470
- Documents with long lines
 - 310

E

- Edit mode preferences
 - 117
- Editor preferences
 - 115
- Editor variables
 - 197
- Emmet add-on
 - 532
- Encoding preferences
 - 114
- Error: Cannot find files.log file
 - 625
- Error: OutOfMemory
 - 621
- Error: OutofMemoryError
 - 634
- Error: stack overflow
 - 620
- Error: Startup crash - Fault Module Name
nvoglv32.dll
 - 626

- Export actions
 - 95
- Export color themes
 - 79
- Export Global Options
 - 189
- Export Global Validation Scenarios
 - 197
- Export Layout
 - 190, 217
- Exporting Markdown documents
 - 449
- External tool configuration
 - 172
- External tool preferences
 - 172
- External Tools
 - 618

F

- File comparison appearance preferences
 - 168
- File comparison preferences
 - 166
- File properties
 - 246
- File type preferences
 - 177
- Find actions
 - 284
- Find All Elements action
 - 283
- Find All Elements dialog box
 - 283
- Find and invoke action
 - 284
- Find/Replace action
 - 273
- Find/Replace dialog box
 - 273
- Find/Replace in Files action
 - 277
- Find/Replace in multiple files

277

Finding/Replacing text

Find All Elements dialog box

283

Find/Replace dialog box

273

Find/Replace in multiple files

277

Navigating Find/Replace matches

287

Quick Find toolbar

286

Floating license servers

Server signature does not match

633

Folding in YAML documents

430

Font preferences

83

Format and Indent Files tool

571

Format and indent in YAML documents

430

Format/Indent multiple files

571

Formatting preferences

135

Framework directory locations

88

Frameworks

470

FTP connection settings

183

G

generate

517, 528

Generate Documentation tools

551, 572

Generate HTML report for directory comparison

tool

345, 615

Generate JSON Schema documentation

540, 572

Generate JSON Schema tool

417, 553

Generate OpenAPI documentation

546, 575

Generate Sample JSON Files tool

419, 551

Getting familiar with the interface

14, 216

Getting Started

13

Getting Started with Oxygen

Help

17

Interface

14, 216

Resources to help you with Oxygen

16

Shortcut keys

20

Supported document types

15

Git addon

486

Git client add-on

486

Global Options

187

Global preferences

76

Grid mode preferences

119

H

Help

Support related resources

17

Using the Help menu

17

Hex Viewer tool

581

Hide file tabs

243

Highlights
348

HTML documents

- Content completion
446
- Editing features
444
- Folding
447
- Minification
447
- Minifying
447
- Outline view
448
- Syntax highlighting
447
- Validation
445

HTML grammar checker
516

HTTP authentication schemes
243

HTTP connection settings
182

HTTPS troubleshooting
241

Huge file editor
310

Huge files
310

I

Ignored validation problems preferences
155

Import color themes
79

Import Global Options
189

Import Global Validation Scenarios
197

Increase memory
215

Installed fonts do not appear in Fonts preferences
629

Installing Oxygen
53

- Group deployment
68
- Linux Installation
59
- Linux/Unix Server Installation
66
- macOS Installation
58
- Upgrading
70
- Windows Installation
53
- Windows Server Installation
64

Integrating external tools
618

Introduction
12

J

Java system properties
205

Java VM parameters
211

JavaScript documents
439

- Content Completion
441
- Editing features
439
- Outline view
443
- Syntax highlighting
442
- Validation
441

JSON documents
358

Associate JSON schema in a framework configuration	365
377	Resolving references to remote schemas
Associate schema	371
373	Sharing scenarios
Associate schema directly in JSON documents	370
376	Validation scenarios
Associate schema through a validation scenario	367
373	XML to JSON converter
Content completion	384, 559
371	XSD to JSON Schema converter
Contextual menu actions	536, 562
388	YAML to JSON converter
Flatten schema	387, 433, 555
423	JSON Lines documents
Folding	Content completion
378	423
JSON to XML converter	Editing features
381, 556	423
JSON to YAML converter	Validation
387, 433, 554	423
Navigating references	JSON schema
360	364, 421
Outline view	JSON Schema
378	395
Presenting validation errors	JSON Schema Converter tool
365	420, 566
Schema annotations	JSON Schema Design mode
372	396
Syntax highlighting	Components
378, 423	405, 405
Validating JSON schema	Contextual menu actions
421	401
Validation	Validation
363	416
Against a schema	JSON Schema Design Mode
364	Editing actions
Automatic validation	400
364	JSON Schema diagram editor
Check Well-formedness	396
363	Components
Manual validation	405, 405
	Contextual menu actions

- 401
- Editing actions
- 400
- Validation
- 416
- JSON Schema Diagram Editor
- Navigation
- 398
- Palette view
- 398
- JSON Schemas
- Design mode editing
- Navigation
- 398
- Palette view
- 398
- JSON to XML tool
- 381, 556
- JSON to YAML tool
- 387, 433, 554
- JSON validation scenarios
- 367
- JSON-LD documents
- Content completion
- 473
- Editing features
- 473
- Validation
- 473

K

- Kerberos
- 243

L

- Large documents
- 308
- Large File Viewer tool
- 579
- Large files
- 309
- Layout configuration
- 190, 217
- Layout preferences

- 85
- Learn document structure
- 377
- LESS stylesheets
- Compile to CSS
- 358
- Content completion
- 357
- Editing features
- 356
- Syntax highlighting
- 357
- Validation
- 357
- Licenses
- 69
- Licensing
- 69, 69
- Load Layout
- 190, 217
- Localizing the user interface
- 210

M

- Mac Function Keys
- 633
- Mac Touch Bar
- 633
- Manual validation in JSON
- 365
- Markdown documents
- 449
- Actions in Markdown Editor
- 450
- Creating Markdown documents
- 450
- Markdown Editor
- 449
- Rules and specifications
- 457
- Syntax highlighting
- 456
- Markdown Editor

449
Markdown grammar checker
516
Memory availability
215
Menu Shortcut Keys
174
Message preferences
185
Move file tabs
243

N

Network connection preferences
180
New document from templates
229
New Document Wizard
225
New from Templates Wizard
229
New project
248
Non-XML files
247
NTLM
243

O

Open file in system application
234
Open preferences
132
Open remote document
236
Open URL dialog box
237
Open/Find Resource
In content
270
In file paths
272
In reviews
272

Open/Find Resource dialog box
267
Open/Find resource preferences
178
Open/Find Resource view
264
OpenAPI document type
470
OpenAPI test scenario documents
Content completion
471
Editing features
471
Validation
471
OpenAPI tester
543, 567
Opening file
234
Options Menu
Preferences
74
Out Of Memory error
579
oxy:allows-child-element function
100
oxy:allows-global-element function
102
oxy:current-selected-element function
103
oxy:is-required-element function
104, 104
oxy:platform function
104
oxy:selected-elements function
103

P

Palette view in JSON Design mode
398
Performance preferences
132
Plugin preferences

171	87
Preferences	Document type configuration dialog box
74	89
Add-ons	Document Type locations
86	88
Appearance	Document Validation
79	153
Application Layout	Edit Modes
85	117
Archive	Editor
170	115
Author mode	Encoding
120	114
AutoCorrect	External Tools
128	172
AutoCorrect Dictionaries	File Types
130	177
Code Templates	Files Comparison
144	166
Colors	Files Comparison Appearance
81	168
Content Completion	Fonts
139	83
Content Completion annotations	Format
143	135
CSS formatting	FTP/SFTP
138	183
CSS Validator	Global
160	76
Cursor Navigation	Grid mode
124	119
Custom Editor Variables	HTTP(S)/WebDAV
180	182
Custom Validation Engines	Ignored validation problems
153	155
Directories Comparison	JavaScript Content Completion
169	142
Directories Comparison Appearance	JavaScript formatting
170	139
Document Templates	JSON Content Completion
114	142
Document Type Association	Mark Occurrences

152	117
Menu Shortcut Keys	Trusted Hosts
174	184
Messages	XPath Content Completion
185	141
Network Connection Settings	YAML Content Completion
180	143
Open	Presenting validation errors in JSON
132	365
Open/Find Resource	Print documents
178	349
Plugins	Print preferences
171	160
Print	Print Preview dialog box
160	349
Project Level Settings	Printing
86	349
Proxy	Problems and solutions
181	622
Save	Project Level Settings preferences
134	86
Schema Design mode	Project Options
131	187
Schema-Aware	Project view
125	251
Spell Check	Projects
156	248
Spell Check Dictionaries	Adding items to projects
158	248
SSH	Creating new projects
184	248
SVN	Managing resources
161	251
SVN Diff	Move resources
164	259
SVN Messages	Project view
165	251
SVN Working Copy	Rename resources
163	259
Syntax Highlight	Sharing
151	261
Text mode	Proxy preferences

181	
Q	
Quick Find toolbar	
286	
R	
RDP performance issues	
621	
Read-only files	
311	
Registering subscription license	
70	
Regular expressions syntax	
287	
Remote Desktop performance issues	
621	
Reset Global Options	
189	
Reset Layout	
190, 217	
Run OpenAPI test scenario tool	
570	
S	
Save file	
235	
Save preferences	
134	
Save remote document	
236	
Schema annotations in JSON	
372	
Scratch Buffer	
311	
Search dialog box	
273	
Search multiple files	
277	
Search preferences	
178	
Searching documents	
263	
Open/Find Resource	
In content	
	270
	In file paths
	272
	In reviews
	272
	Open/Find Resource dialog box
	267
	Open/Find Resource view
	264
	Server signature does not match
	633
	SFTP connection settings
	183
	Sharing document templates
	234
	Sharing JSON validation scenarios
	370
	Sharing project level options
	188
	Sharing project options file
	261
	Minimize differences between versions
	263
	Sharing settings
	188
	Sharing transformation scenarios
	261
	Sharing validation scenarios
	261
	Shortcut key configuration
	174
	Shortcut Keys
	20
	Simple text editor
	247
	Single sign-on
	243
	Smart paste preferences
	125
	Special character preferences
	132
	Special characters

- 302
 - Character Map
 - 306
 - Fallback Font Support
 - 305
 - Inserting symbols
 - 306
 - Unrecognized characters
 - 304
 - Unsupported characters
 - 304
- Spell check preferences
 - 156
- Spell checking
 - 288
 - Add dictionaries
 - 292
 - Add term lists
 - 294
 - Custom list of terms
 - 294
 - Customizing dictionaries and term lists
 - 291
 - Dictionaries and term lists
 - 290
 - Forbidden words
 - 294
 - Ignored words
 - 297
 - Learned words
 - 296
 - Multiple files
 - 299
 - Replace dictionaries
 - 295
- Spell Checking
 - Automatic spell check
 - 297
- SSH connection settings
 - 184
- StackOverflowException error
 - 155

- Startup parameter
 - Application launcher parameters
 - 211
 - Command-line script parameters
 - 214
 - Custom startup parameters file
 - 214
- Startup parameters
 - 211
- Status information
 - 348
- Storing global options
 - 187
- Storing project level options
 - 187
- Subscription licenses
 - 70
- Supported frameworks
 - OpenAPI
 - 470
- SVN message preferences
 - 165
- SVN preferences
 - 161
- Swagger
 - 470
- Switch file tabs
 - 243
- Syntax highlight preferences
 - 151
- System properties
 - 205

T

- Tags hide text
 - 636
- Terminology checker
 - 507
- Text editing mode
 - Format and indent multiple files
 - 571
- Text flows out of code blocks
 - 636

Text invisible issue
636

Text mode preferences
117

Text rendering issues
636

Three-way directory comparison and merge tool
338, 607

Three-way file comparison
313, 582

Toolbar configuration
194, 222

Touch Bar on Mac
633

Trackpad scroll function doesn't work on Lenovo
Thinkpad
635

Trusted hosts connection settings
184

Two-way file comparison
313, 582

U

Unicode support
303

Uninstalling Oxygen
72

Upgrading Oxygen
70

UTF-8 BOM handling
114

V

Validating JSON Documents
Automatic validation
364

Check Well-formedness
363

Manual validation
365

Resolving references to remote schemas
371

Sharing scenarios
370

Validating JSON Documents against a schema
364

Validating JSON schema
421

Validation engine 'StackOverflowException' error
155

Validation preferences
153

W

WebDAV connection settings
182

WebDAV over HTTPS
240

X

XML to JSON tool
384, 559

XPath Expressions
474

Catalogs
478

XPath Builder view
474

XPath Results view
477

XSD to JSON Schema tool
536, 562

Y

YAML documents
424

Associate schema directly in YAML documents
428

Content completion
429

Outline view
431

Syntax highlighting
430

Validation
425

Validation scenarios
426

YAML editor

424

YAML to JSON tool

387, 433, 555

YAML validation scenarios

426

Copyright

Oxygen JSON Editor User Manual

Syncro Soft SRL.

Copyright © 2002-2023 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 Websites referenced elsewhere within this documentation, and Syncro Soft SRL does not necessarily endorse the products, services, or information described or offered within them. Syncro Soft cannot guarantee

that these links will work all the time and has 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 JSON Editor End-User License Agreement, as well as information regarding support for this product, while under warranty, is available through the [Oxygen JSON Editor End-User License Agreement](#).

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 [Third-party License Agreements webpage](#).

Terms and conditions: For the terms and conditions for using Oxygen JSON Editor, see [Oxygen JSON Editor End-User License Agreement](#).

Documentation: For the most current versions of documentation, see the [Oxygen JSON Editor User Manual](#).

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 support webpage](#).