XMLmind XML Editor - Online Help

Hussein Shafie
XMLmind Software
<xmleditor-support@xmlmind.com>

XMLmind XML Editor - Online Help

Hussein Shafie
XMLmind Software
<xmleditor-support@xmlmind.com>

Publication date June 30, 2026

Abstract

This online help contains, in addition to a short tutorial, the reference manual of all the menus, tool bars and dialog boxes of XMLmind XML Editor (**XXE** for short).

Table of Contents

I. Getting started 1
1. Install 3
1. Installing XMLmind XML Editor (**XXE** for short) 3
2. Contents of the installation directory 6
2. Getting started with XMLmind XML Editor 8
II. Reference 11
3. Menus 15
1. **File** menu 15
1.1. Folder menu 17
2. **Select** menu 18
2.1. The selection in **XXE** 18
2.1.1. Text selection 18
2.1.2. Node selection 19
2.1.3. Implicit element selection 20
2.2. **Select** menu items 20
3. **Edit** menu 21
3.1. Comment menu 26
3.2. Processing instruction menu 26
3.3. Reference menu 27
4. **Search** menu 29
4.1. The "**Find and Replace Element**" menu 30
4.2. The "**Web Search**" menu 31
5. **View** menu 32
5.1. The "**XML source**" menu item 34
6. **Tools** menu 36
6.1. **Remark** menu 39
6.2. **Revisions** menu 40
6.3. "Helper Application" menu 42
6.4. "Record Macro" menu 44
7. **XML** menu 46
8. **Options** menu 46
8.1. "Customize Configuration" menu 46
9. **Tab** menu 48
10. **Help** menu 48
4. The document view area 50
5. The main tool bar 52
1. Buttons which are common to all document types 52
2. Buttons which are specific to type of the document being edited 53
6. The node selection bar 54
1. The node path bar 54
1.1. Non-editable nodes 55
1.2. Elements which are in non-validating mode 55
1.3. Configuring the node path bar to display the value of some attributes 56
2. Navigation tool bar 57
7. Tools 59
1. Edit tool 59
1.1. Reducing the number of elements displayed by the **Edit** tool 61
2. Characters tool 62
2.1. The “**Favorites**” palette 63
3. Attributes tool 64

3.1. Using the attribute table 64
3.1.1. Reducing the number of attributes displayed by the attribute table 65
3.2. Using the attribute form 67
4. Search tool 69
5. Validity tool 71
6. The "**Browse Files**" tool 72
6.1. Opening and comparing revisions of a document 76
7. The "**Search Files**" tool 78
7.1. The "**Create or edit an advanced query**" dialog box 80
7.2. Syntax of search queries 83
8. The **Compare** tool 84
8.1. Why use the **Compare** tool? 84
8.2. Using the **Compare** tool 85
8. Status bar 88
1. Validity state 88
2. Document cache indicator 89
3. Status messages 89
4. Spell checker 89
5. Overwrite mode 89
6. Clipboard utilities 90
6.1. Clipboard content 90
9. Dialog boxes 91
1. The URL chooser dialog box 91
1.1. The "Document conversion parameters" panel 94
2. The "**Find and Replace Element**" dialog box 96
3. The "**Declare Namespace**" dialog box 101
4. The "**Attribute Value**" dialog box 103
5. The "**Check Spelling**" dialog box 103
6. The "**Preferences**" dialog box 106
6.1. New options 106
6.2. Open options 106
6.3. Save options 109
6.4. Print options 112
6.5. Edit options 113
6.6. Web search options 115
6.7. View options 116
6.7.1. "**CSS stylesheet defaults**" subsection 117
6.7.2. "**Tree view settings**" subsection 118
6.7.3. Preferences related to images 118
6.8. Tools options 119
6.8.1. Master document options 119
6.8.2. Validate options 119
6.8.3. Spell options 120
6.8.4. Revisions options 120
6.8.5. Helper applications options 121
6.9. Install add-ons options 121
6.10. General options 123
6.10.1. Features Options 124
6.11. Advanced options 126
6.11.1. Cached data options 126
6.11.2. Proxies options 128
6.12. Add-on options 130

6.12.1. Apache FOP options 130
6.12.2. RenderX XEP options 132
6.12.3. Text format options 133
6.12.4. JSON format options 134
7. The "**Tree view fonts and colors**" dialog box 135
8. The "**Helper Application Editor**" dialog box 136
9. The "**Save Document As Template**" dialog box 140
9.1. Organizing the content of the **File** → **New** dialog box 141
9.2. Removing a previously defined document template 142
10. The "**Save Selected Element as Template**" dialog box 143
10.1. Advanced options 144
10.2. Removing a previously defined element template 146
11. The "**Add Keyboard Shortcut**" dialog box 146
11.1. Removing a previously defined keyboard shortcut 150
12. The "**Customize CSS Stylesheet**" dialog box 152
12.1. Quickly customizing the style of selected element 153
13. The "**Customize Document Conversion Stylesheets**" dialog box 155
13.1. Introduction 155
13.2. Reference 157
13.3. The "XMLmind XSL Customizer" application 163
13.3.1. Introduction 163
13.3.2. Reference 165
14. The "**Install Add-ons**" dialog box 170
14.1. *Automatically* upgrading installed add-ons 173
14.2. How to "*manually*" install, uninstall or upgrade an add-on 174
15. Dialog box allowing to edit "text style" menu items 175
15.1. Dialog box allowing to edit a "text style" menu item 178
16. The media player dialog box 179
16.1. Adding audio or video to your document 179
16.2. Using the media player dialog box 181
16.2.1. The **Media** tab 182
16.2.2. The **Setup** tab 184
17. The "**Edit Image Map**" dialog box 185
17.1. Adding an image map to your document 185
17.2. Using the image map editor 187
17.2.1. The toolbar of the image map editor 188
17.2.2. Specifying the link target and optionally the link text of selected area 189
17.3. Creating and using an annotated variant of image file 190
A. Command line options 192
1. Command line 192
2. Preference keys 194
3. Environment variables 203
4. System properties 204

List of Figures

3.1. Perform a web search using Google by pressing **F3** and then g 32
3.2. A remark made by user `hussein` and its tooltip. 39
3.3. Remark editor #1 displayed when a new remark is to be inserted in the document 39
3.4. Remark editor #2 lets the user reply to a remark made by another user or modify her/his own remark or delete it 40
3.5. The "Open Revision" dialog box, here allowing to compare the document being edited (revision #4; latest one) to the initial revision (revision #1) 42
3.6. The "**Helper Application Not Found**" dialog box 43
4.1. Tab corresponding to the active document 50
4.2. Tab corresponding to the active document when it is in read-only mode 50
7.1. Menu when an XHTML5 document is opened in **XXE** (inserting an element after a `li`) 61
7.2. The **Characters** tool 62
7.3. The dialog box displayed by the "**Insert special character by name**" button 63
7.4. Menu when a DITA topic is opened in **XXE** 66
7.5. The "**Browse Files**" tool showing the source files of our DITA tutorial 73
7.6. The "List Revisions" dialog box 77
7.7. Search our DITA tutorial for topic files (`*.dita`) containing substring "`link`" 78
7.8. The "**Create or edit an advanced query**" dialog box 81
7.9. The **Compare** tool showing the differences existing between `doc.xml 1` (initial revision) and `doc.xml` (latest revision), displayed side by side 85
7.10. The **Compare** tool showing change #2 86
8.1. The clipboard viewer dialog box 90
9.1. An URL chooser dialog box having an expanded "**Document conversion parameters**" panel 94
9.2. The "**Find and Replace Element**" dialog box 96
9.3. The "**Find Element**" dialog box 97
9.4. The popup list displayed by the "**History**" button. 100
9.5. The "**Declare Namespace**" dialog box showing extra namespaces for an XHTML document 102
9.6. Composition area displayed below the caret 113
9.7. Dialog box allowing to add or edit a web search service specification 115
9.8. The "**Install add-ons**" preferences sheet 122
9.9. The "**FOP**" preferences sheet 130
9.10. The "**Choose a TTF font and specify its aliases**" dialog box 131
9.11. The "**Text format**" preferences sheet 133
9.12. The "**Customize CSS Stylesheet**" dialog box 152
9.13. Right-click popup menu of the "**Customize CSS Stylesheet**" dialog box 153
9.14. An easy to use style editor 154
9.15. The "**CSS Source**" tab of the style editor 155
9.16. Converting an XML document to an XSL-FO based format 156
9.17. Converting an XML document to an HTML-based format 157
9.18. The "**Customize Document Conversion Stylesheets**" dialog box showing the stylesheets used when a DITA map is converted to single-page XHTML 157
9.19. "XMLmind XSL Customizer" main window 164
9.20. The **Add**/**Edit** attribute dialog box 167
9.21. The **Add**/**Edit** parameter dialog box 168
9.22. The "**Install Add-ons**" dialog box 171
9.23. The **Uninstall** tab 172
9.24. Toggles found at the beginning of the DocBook tool bar 176
9.25. Customizing the items of the DocBook' “italic text style” menu 176

9.26. The menu item editor 178
9.27. Two embedded media players, one for video and the other for audio, both having controls 180
9.28. The right-click popup menu of a media player (audio only, no controls) 180
9.29. The **Media** tab of the media player dialog box 182
9.30. The **Setup** tab of the media player dialog box 184
9.31. The right-click, contextual, popup menu has an "**Edit Image Map**" entry 186
9.32. The "**Edit Image Map**" dialog box 187
9.33. The "**Edit image map area**" dialog box 189
9.34. The popup menu displayed by clicking the "**Create and use an annotated variant of image file**" toolbar button 190
9.35. Example of annotated image preceding an XHTML `map` 190

List of Tables

9.1. Lock Types 108
9.2. Substituted Variables 112

List of Examples

3.1. A use case for command **Force Deletion** 23
3.2. Example: book referencing chapters 28
3.3. Example explaining the enhanced cross-reference creation/validation provided by a master document 37
9.1. DocBook example: change the ID of an element *and* update all links to this element 100
9.2. XHTML example: add attribute `target="_blank"` to all external links 101
9.3. Default language versus actual language example 104
9.4. When converting a DITA map to PDF, use a 12pt base font size and give a light gray background to all the topic titles 159
9.5. When converting a DITA map to single-page XHTML, use a 12pt base font size and give a light gray background to all the topic titles 160
9.6. When converting a DITA map to PDF, give a light gray background to all the topic titles 169
9.7. When converting a DITA map to PDF, use a 12pt base font size 170

Part I. Getting started

Table of Contents

1. Install 3
    1. Installing XMLmind XML Editor (**XXE** for short) 3
    2. Contents of the installation directory 6
2. Getting started with XMLmind XML Editor 8

Chapter 1. Install

1. Installing XMLmind XML Editor (XXE for short)

Requirements

XXE should run fine on any platform supporting Java 11+. However it is officially supported only on:

- Windows version 8.1 to version 11, 32-bit or 64-bit, with the Windows Look & Feel (**LAF**).
- macOS 26.x (Tahoe) and 15.x (Sequoia), Intel® or ARM® (**AKA** M1 **AKA** Apple® Silicon) processor, with the Macintosh **LAF**.
- Linux, with the Metal **LAF**.
- The add-on called "*FlatLaf Look and Feel*" containing a clean, simple and elegant “Look & Feel” for **XXE** is supported on all platforms. All FlatLaf themes are officially supported.

It is possible to use **XXE** on other platforms and/or with other **LAF**s, but without support from XMLmind Software.

XXE should work fine on Macs having Retina® screens and on Personal Computers having UHD (“4K”) screens. On Windows, all DPI scale factors —100%, 125%, 150%, 200%, etc— are supported. However for this to work, please make sure to install latest Oracle® Java™ runtime and to use the system, default, Look and Feel.

Install on Windows

1. Download the `setup.exe` file.

   This `setup.exe` files includes a very recent, generally the most recent, Oracle® *private* Java™ runtime. Therefore, you don't need to install Java on your computer. Moreover, if you have Java already installed on your computer, then your public Java runtime will be ignored by **XXE**.

2. Double-click on the `setup.exe` file to launch the installer.

   Requires administrative privileges if the application is installed for all users. Does *not* require administrative privileges if the application is installed only for the current user.

3. Follow the instructions of the installer.

About Java on Windows

The `setup.exe` distribution includes a very recent —generally the most recent— *private* OpenJDK Java runtime.

If you prefer to run **XXE** using a different version of Java™, you'll have to first delete folder *`XXE_INSTALL_DIR`*`\bin\jre64\` in order to force **XXE** to use the version of Java installed on your computer.

Note that *`XXE_INSTALL_DIR`*`\bin\jre64\` contains a 64-bit version of the Java runtime which cannot be used on a 32-bit version of Windows. This means that, on a 32-bit version of Windows, you'll still have to download and install a 32-bit Java™ 8+ runtime on your computer in order to use **XXE**.

Install on the Mac

1. Download the `.dmg` file.
2. Double-click the downloaded `.dmg` file to open it in the Finder.
3. Copy the `XMLmind.app` folder, an application bundle represented by icon , anywhere you want. For example, drag&drop this icon to the `/Applications` folder or to your desktop.
4. Start XMLmind XML Editor by double-clicking on the icon (or use the Launchpad).
5. The first time XMLmind XML Editor is started, your Mac will generally ask you to confirm that you actually want to open an application downloaded from the Internet. Click **Open** to confirm.

   Don't worry, XMLmind XML Editor has been digitally signed using a certificate issued by Apple itself. This confirmation is required for any digitally signed application not coming from the App Store.
6. Move the downloaded `.dmg` file to the Trash.

About Java on the Mac

The `.dmg` distribution includes a very recent —generally the most recent— *private* OpenJDK Java runtime.

If you prefer to run **XXE** using a different version of Java, you'll have to first delete folder `XMLmind.app/Contents/Resources/xxe/bin/jre/` (`jrea/` for a Mac having an ARM processor) in order to force **XXE** to use the version of Java installed on your computer.

Install on Linux or *alternative* install on the Mac

On the Mac, it is strongly recommended to download and install the `.dmg` file.

1. Make sure that the Java™ `bin/` directory is referenced in the $PATH and, at the same time, check that the Java™ runtime in the $PATH has the right version:

```
~$ java -version
openjdk version "26.0.1" 2026-04-21
OpenJDK Runtime Environment (build 26.0.1+8-34)
OpenJDK 64-Bit Server VM (build 26.0.1+8-34, mixed mode)
```

2. Unpack the **XXE** distribution inside any directory you want:

```
$ cd
$ unzip xxe-pro-11_2_0.zip
$ ls xxe-pro-11_2_0
addon/
bin/
```

```
demo/
doc/
...
```

3. **XXE** is intended to be used directly from the `xxe-pro-11_2_0/` directory. That is, you can start **XXE** by simply executing:

```
$ xxe-pro-11_2_0/bin/xxe &
```

After that, you may want to add `xxe-pro-11_2_0/bin/` to your `$PATH`.

Manual install on Windows

On Windows, it is strongly recommended to download and install the auto-installable `setup.exe` file.

1. Make sure that you have a Java™ 8+ runtime installed on your machine. To check this, open a command window and type "`java -version`" followed by Enter. You should get something looking like this:

```
C:\> java -version
openjdk version "26.0.1" 2026-04-21
OpenJDK Runtime Environment (build 26.0.1+8-34)
OpenJDK 64-Bit Server VM (build 26.0.1+8-34, mixed mode)
```

2. Unzip the **XXE** distribution inside any directory you want:

```
C:\> mkdir XMLmind
C:\> cd XMLmind
C:\XMLmind> unzip xxe-pro-11_2_0.zip
C:\XMLmind> dir xxe-pro-11_2_0
... <DIR> addon
... <DIR> bin
... <DIR> demo
... <DIR> doc
...
```

3. **XXE** is intended to be used directly from the `xxe-pro-11_2_0/` directory. That is, you can start **XXE** by simply executing:

```
C:\XMLmind> xxe-pro-11_2_0\bin\xxe.exe
```

After that, you may want to add a shortcut to "`C:\XMLmind\xxe-pro-11_2_0\bin\xxe.exe`" on your desktop.

Note that the `bin/` directory contains not only `xxe.exe`, but also `xxe-c.bat`, which starts **XXE** with a *console*. On Windows, a console is needed to be able to see low-level error messages, for example, when developing custom configurations or extensions.

2. Contents of the installation directory

File layout when you install the `.dmg` distribution on the Mac

All the files described below are found in folder `XMLmind.app/Contents/Resources/xxe/`.

If you want to see the contents of the `XMLmind.app` folder, an application bundle, please open a Finder window, right-click (or **Ctrl**+click) on the XMLmind.app icon and select "**Show Package Contents**" from the popup menu.

`addon/`

This `addon/` directory contains a number of add-ons which are bundled with **XXE**.

`addon/config/`

Contains configuration files for a number of document types: DocBook, DITA, XHTML, etc.

The content of a configuration file, which specifies a customization of **XXE** for a specific XML application, is described in detail in XMLmind XML Editor - Configuration and Deployment.

`bin/`

Contains **XXE** code (`.jar` files) and many scripts used to start **XXE** and its associated utilities.

`bin/xxe.exe`, `xxe.jstart`

File `xxe.exe` is used to start **XXE** on Windows. This `.exe` file is a home-made launcher parameterized by `xxe.jstart`, an UTF-8 encoded, plain text file.

`bin/xxe`, `xxe-c.bat`

Scripts used to start **XXE**. Use `xxe` on Linux and on the Mac. Use `xxe-c.bat` on Windows, but only when you need to start **XXE** with a *console*. On Windows, a console is needed to be able to see low-level error messages, for example, when developing custom configurations or extensions.

`bin/xxeconvert, xxeconvert.bat`

Scripts used to run **xxeconvert**. This command-line utility can be used convert XML documents to other formats (HTML, PDF, RTF, etc) from within a script or a makefile in the exact same way as in the "**Convert Document**" submenus of **XXE**.

See The **xxeconvert** command-line utility for more information about this tool.

`bin/xmltool`, `xmltool.bat`

Scripts used to run **xmltool**. See The **xmltool** command-line utility for more information about this tool.

`bin/authvalue`, `autvalue.bat`

Scripts allowing to generate an authorization token for use by **XXE**. See command-line option `-auth` [193] for more information about these authorization tokens.

`bin/*.jar`

All the Java™ class libraries needed to run **XXE**:

- `xxe.jar` contain the code of **XXE**.

  Substantial parts of `xsdregex.jar`, James Clark's XSD to Java Regular Expression Translator, have been directly added to `xxe.jar` (which is why file `xsdregex.jar` is not included in the distribution). Download original package from http://www.thaiopensource.com/download/.

  Package `com.jclark.xsl.expr` contains the implementation of XPath 1.0 used by XT, James Clark's XSLT engine. A modified version of this package, renamed `com.xmlmind.xml.xpath`, has been directly added to `xxe.jar` (which is why file `xt.jar` is not included in the distribution). Download full XT from http://www.jclark.com/xml/xt-old.html or from http://www.blnz.com/xt/index.html.

- `xerces.jar` contains XML parser Apache Xerces2 Java to version 2.12.2. (The version included in the Java™ runtime has bugs which have been fixed in bundled version.)

- `xmlresolver.jar` contains XMLResolver, an enhanced XML resolver with XML Catalog support.

- `relaxng.jar` is Jing version 20030619, James Clark's RELAX NG validator, slightly modified for use in **XXE**. The details of the modifications are found in `relaxng.README`.

- `saxon.jar` is Michael H. Kay's XSLT 1 engine. See http://saxon.sourceforge.net/.

- `saxon12.jar` is Michael H. Kay's XSLT 3.0 engine. See https://www.saxonica.com/.

Some of the above JAR files have not been developed by XMLmind. Copyright information is contained in the corresponding `legal/`*`xxx`*`.LICENSE` file. Read the corresponding `legal/`*`xxx`*`.README` file to have more details about these excellent Java class libraries.

`bin/icons/`

Contains desktop icons for **XXE**.

`demo/`

Contains XML documents that can be opened in **XXE** to demo some of its features.

`doc/`

Contains XMLmind XML Editor documentation in HTML and PDF (Acrobat) formats.

`legal/`, `legal.txt`

Contains legal information about **XXE** and about third-party components used in **XXE**.

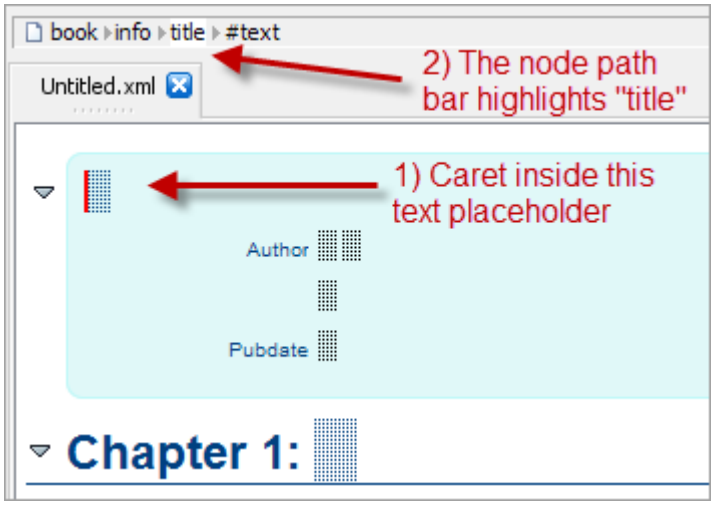
Chapter 2. Getting started with XMLmind XML Editor

There are a few things you need to learn before starting to use XMLmind XML Editor (**XXE** for short).

XXE is quite straightforward to use but you simply cannot guess how it works. While what you see resembles your typical word processor, **XXE** does not work like a word processor, nor like a text editor, neither like other XML editors. Therefore this chapter is really a must read.

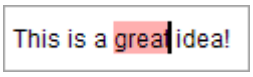
- Patterns looking like this are *text placeholders*. You can click into (or tab to) such placeholders and start typing.

- The *node path bar*, found above the document view, indicates what is selected or when there is no explicit selection, the element containing the caret.

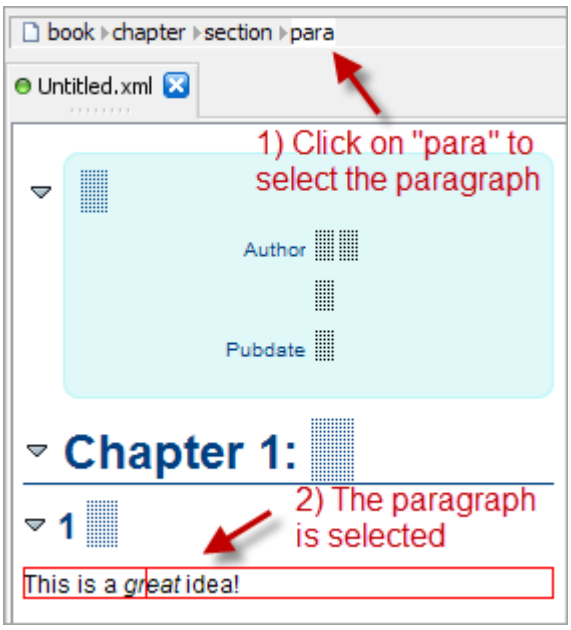


- **XXE** has 2 different kinds of selection, the *text selection* and the *node selection*.

  a. The text selection, which is given a pink background color, works like in any word processor or text editor. Few editing commands apply to the text selection: **Edit** → **Paste**, **Edit** → **Convert**. For example, you can convert the text selection to a bold or italic inline element.

  b. The node selection looks different than the text selection: a thin red frame is drawn around the selected nodes. The simplest way to select a node is to click on its name in the node path bar.



Most editing commands apply to the node selection.

The node selection may comprise several sibling nodes, for example, two contiguous paragraphs. The easiest way to extend the node selection is to Shift-click before or after the thin red frame.

- The element directly containing the caret is always *implicitly selected*: no thin red frame around it.

  For example if you want to insert a table before a paragraph, first click inside this paragraph (but not inside any of its child elements, as this would implicitly select the child element) and then use **Edit** → **Insert Before**.

  However, if you want to insert a list after a table, you cannot do that using the implicit element selection because a table cannot directly contain some text. You'll have to first explicitly select this table (e.g. by clicking on "`table`" in the node path bar) and then use **Edit** → **Insert After**.

- Notice that the two editing commands mentioned above are **Edit** → **Insert Before**, **Edit** → **Insert After**. **XXE** also has less commonly used **Edit** → **Insert** editing command.
  Edit → **Insert Before**

  Insert an element (or text node) *just before the node selection*.

  Edit → **Insert**

  Insert an element *right here, at caret position*.

  Edit → **Insert After**

  Insert an element (or text node) *just after the node selection*.

Note that ***XXE** does not work like other XML editors*. In other XML editors,
- **Insert Before** often means "Insert *somewhere* before selection".
- **Insert** often means "Insert *somewhere* inside selection".
- **Insert After** often means "Insert *somewhere* after selection".

- For the same reasons, **XXE** has 3 paste commands and not just one: **Edit** → **Paste Before**, **Edit** → **Paste**, **Edit** → **Paste After**. Unlike **Edit** → **Insert**, **Edit** → **Paste** is commonly used and works as expected by replacing the text or node selection with the contents of the clipboard.

- When **XXE**, which is a strictly validating XML editor, does not allow you to perform the editing command you want, it's almost always because you didn't select the right object and/or you are not using the right editing command.

  For example, you have clicked inside a paragraph and attempt to use **Edit** → **Insert** to add a section after it. This cannot work because a paragraph section contain a chapter. Instead, first select the section which is the ancestor of the paragraph (e.g. click "`section`" in the node path bar) then use **Edit** → **Insert After** (not **Edit** → **Insert**) and select "`section`" from the list.

Watch the screencast corresponding to what's explained above.

Part II. Reference

Table of Contents

3. Menus 15
1. **File** menu 15
1.1. Folder menu 17
2. **Select** menu 18
2.1. The selection in **XXE** 18
2.1.1. Text selection 18
2.1.2. Node selection 19
2.1.3. Implicit element selection 20
2.2. **Select** menu items 20
3. **Edit** menu 21
3.1. Comment menu 26
3.2. Processing instruction menu 26
3.3. Reference menu 27
4. **Search** menu 29
4.1. The "**Find and Replace Element**" menu 30
4.2. The "**Web Search**" menu 31
5. **View** menu 32
5.1. The "**XML source**" menu item 34
6. **Tools** menu 36
6.1. **Remark** menu 39
6.2. **Revisions** menu 40
6.3. "Helper Application" menu 42
6.4. "Record Macro" menu 44
7. **XML** menu 46
8. **Options** menu 46
8.1. "Customize Configuration" menu 46
9. **Tab** menu 48
10. **Help** menu 48
4. The document view area 50
5. The main tool bar 52
1. Buttons which are common to all document types 52
2. Buttons which are specific to type of the document being edited 53
6. The node selection bar 54
1. The node path bar 54
1.1. Non-editable nodes 55
1.2. Elements which are in non-validating mode 55
1.3. Configuring the node path bar to display the value of some attributes 56
2. Navigation tool bar 57
7. Tools 59
1. Edit tool 59
1.1. Reducing the number of elements displayed by the **Edit** tool 61
2. Characters tool 62
2.1. The “**Favorites**” palette 63
3. Attributes tool 64
3.1. Using the attribute table 64
3.1.1. Reducing the number of attributes displayed by the attribute table 65
3.2. Using the attribute form 67
4. Search tool 69
5. Validity tool 71
6. The "**Browse Files**" tool 72

6.1. Opening and comparing revisions of a document 76
7. The "**Search Files**" tool 78
7.1. The "**Create or edit an advanced query**" dialog box 80
7.2. Syntax of search queries 83
8. The **Compare** tool 84
8.1. Why use the **Compare** tool? 84
8.2. Using the **Compare** tool 85
8. Status bar 88
1. Validity state 88
2. Document cache indicator 89
3. Status messages 89
4. Spell checker 89
5. Overwrite mode 89
6. Clipboard utilities 90
6.1. Clipboard content 90
9. Dialog boxes 91
1. The URL chooser dialog box 91
1.1. The "Document conversion parameters" panel 94
2. The "**Find and Replace Element**" dialog box 96
3. The "**Declare Namespace**" dialog box 101
4. The "**Attribute Value**" dialog box 103
5. The "**Check Spelling**" dialog box 103
6. The "**Preferences**" dialog box 106
6.1. New options 106
6.2. Open options 106
6.3. Save options 109
6.4. Print options 112
6.5. Edit options 113
6.6. Web search options 115
6.7. View options 116
6.7.1. "**CSS stylesheet defaults**" subsection 117
6.7.2. "**Tree view settings**" subsection 118
6.7.3. Preferences related to images 118
6.8. Tools options 119
6.8.1. Master document options 119
6.8.2. Validate options 119
6.8.3. Spell options 120
6.8.4. Revisions options 120
6.8.5. Helper applications options 121
6.9. Install add-ons options 121
6.10. General options 123
6.10.1. Features Options 124
6.11. Advanced options 126
6.11.1. Cached data options 126
6.11.2. Proxies options 128
6.12. Add-on options 130
6.12.1. Apache FOP options 130
6.12.2. RenderX XEP options 132
6.12.3. Text format options 133
6.12.4. JSON format options 134
7. The "**Tree view fonts and colors**" dialog box 135
8. The "**Helper Application Editor**" dialog box 136

9. The "**Save Document As Template**" dialog box 140
9.1. Organizing the content of the **File** → **New** dialog box 141
9.2. Removing a previously defined document template 142
10. The "**Save Selected Element as Template**" dialog box 143
10.1. Advanced options 144
10.2. Removing a previously defined element template 146
11. The "**Add Keyboard Shortcut**" dialog box 146
11.1. Removing a previously defined keyboard shortcut 150
12. The "**Customize CSS Stylesheet**" dialog box 152
12.1. Quickly customizing the style of selected element 153
13. The "**Customize Document Conversion Stylesheets**" dialog box 155
13.1. Introduction 155
13.2. Reference 157
13.3. The "XMLmind XSL Customizer" application 163
13.3.1. Introduction 163
13.3.2. Reference 165
14. The "**Install Add-ons**" dialog box 170
14.1. *Automatically* upgrading installed add-ons 173
14.2. How to "*manually*" install, uninstall or upgrade an add-on 174
15. Dialog box allowing to edit "text style" menu items 175
15.1. Dialog box allowing to edit a "text style" menu item 178
16. The media player dialog box 179
16.1. Adding audio or video to your document 179
16.2. Using the media player dialog box 181
16.2.1. The **Media** tab 182
16.2.2. The **Setup** tab 184
17. The "**Edit Image Map**" dialog box 185
17.1. Adding an image map to your document 185
17.2. Using the image map editor 187
17.2.1. The toolbar of the image map editor 188
17.2.2. Specifying the link target and optionally the link text of selected area 189
17.3. Creating and using an annotated variant of image file 190

Chapter 3. Menus

1. File menu

New Window

Opens a new main window identical to the initial main window of XMLmind XML Editor. See also menu item **Move Document to New Window** [50] found in the right-click popup menu of a document tab.

Note that using **File** → **Quit** will only close the corresponding main window. The XMLmind XML Editor application is actually terminated when its last main window has been closed using **File** → **Quit**.

You may need to enable this menu item by checking "**Enable the "Window|New Window" menu item**" in **Options** → **Preferences**, **General|Features** section.

New

Creates a new document which is a copy of a named template.

This dialog box displayed by this command contains a list of named document templates (example: Article). A document template is listed below the name of the document type (example: DocBook) to which it belongs.

Use the URL Chooser

If this toggle is checked, XXE will display an URL chooser dialog box [91] rather than the standard file chooser dialog box each time the user needs to specify a file name.

Default: not checked.

Open

Opens an existing document. Displays standard file chooser dialog box or advanced URL chooser [91] dialog box depending on the state of the "**Use the URL Chooser**" checkbox [15].

- It is also possible to open a document by dropping its icon anywhere inside **XXE** main window (except onto “drop sites” such as images in a document view). More information in Drag and drop in XMLmind XML Editor [17].
- Shift-click on this menu item to open a document in read-only mode.

Opening plain text files in XMLmind XML Editor

This feature is available only when the "**Edit source**" add-on has been installed.

The "**Edit source**" add-on adds to **XXE** the capability to open, edit and save all kinds of text files (`.css`, `.js`, `.dtd`, etc), including "not well-formed" XML files. This is done using an advanced text editor featuring syntax highlighting, function folding, etc.

Which files are directly opened as text files (i.e. not as XML files first) can be specified using the **Options** → **Preferences**, "**Add-on|Text format**" preferences form.

Why open text files in **XXE**?

1. This is handy to fix "not well-formed" XML files.
2. After downloading and installing the corresponding virtual drive plug-ins, **XXE** can open, edit and save text files stored on remote servers (WebDAV, FTP, FTPS, SFTP, Google Drive).
3. **XXE** can lock local text files and text files stored on a WebDAV server.

Save

Saves the current document, but only if it has been modified.

Behaves like **Save As** if the current document is a newly created document or if the current document has not the write permission for the user.

At least for XXE, you do not have to save XML documents with a `.xml` extension. You can use more meaningful extensions.

Save As

Saves the current document to a different location. Displays standard file chooser dialog box or advanced URL chooser [91] dialog box depending on the state of the "**Use the URL Chooser**" checkbox [15].

Save All

Saves all modified documents.

Page Setup

Selects the page size and margins used for printing. Displays standard **Page Setup** dialog box.

Print

Prints the current view of the current document and this, no matter what is being selected. Displays standard **Print** dialog box.

More text on printed pages

Temporarily decrease the size of the text using **View** → **Text Size** → **Smaller** [33] (or use your mouse wheel to do that) if you want to see more text on the printed pages.

Print Selected Element

Allows to print a subset of current document. Requires one or more nodes to be explicitly selected (see Select menu [20]). If several nodes are selected, it is the parent element of these nodes which is printed.

Quit

Quit **XXE**.

Below the above menu items, a menu item is added for the last nine documents which have been opened in **XXE**. Selecting such menu item:

- Loads the document into **XXE** if it is not currently loaded.

- Reloads the document into **XXE**, closing all existing views, if it is currently loaded. If the document has been modified, the user will have to confirm that changes are to be discarded.

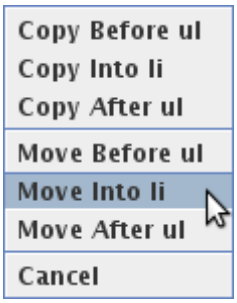
- Shift-click on a menu item to reopen the corresponding document in read-only mode.
- Ctrl-click or right-click[1] on a menu item to display a file chooser dialog box. This dialog box allows to quickly and easily choose a file found in the same directory as the recently opened file clicked upon.

Drag and drop in XMLmind XML Editor

It is possible to drag the text or node selection in order to drop it elsewhere, by dragging it while keeping the left button and the **Alt** key pressed.

The action associated to drop depends on the nature of the object being dropped and on the drop location:

- Dropping an URL or file onto an image element changes the source file of this image.
- Dropping an URL onto an external link element changes the target of this link.
- Dropping an URL or file outside a document view or inside a document view but on elements other than images and links opens the corresponding document in XMLmind XML Editor.
- Otherwise, dropping some text or some XML nodes inside a document view displays a popup menu allowing to copy or move the data being dropped.



Note that only the current text or node selection can be moved within a document. Any data other than the current text or node selection can be copied, but not moved. Moreover, the current text or node selection cannot be moved between two different documents.

1.1. Folder menu

You may need to enable this submenu by checking "**Enable the 'File|Folder' Submenu**" in **Options** → **Preferences**, **General|Features** section.

Open Current Folder

Open the folder[2] containing the last opened document.

[1]Mac users: make sure to press **Ctrl** and not Cmd which clicking. Also note that right-clicking with a 3-button mouse will not work.
[2]A folder is also called a *directory* on some platforms.

Open Home Folder

Open the Home folder of the user. For example, opens `/Users/john` on the Mac.

Open Folder

Select a folder[2]. Displays standard file chooser dialog box or advanced URL chooser [91] dialog box depending on the state of the "**Use the URL Chooser**" checkbox [15].

After the folder is selected, XXE displays the "**Browse Files**" tool [72] in order to let the user interact with this folder.

2. Select menu

2.1. The selection in XXE

There are three types of selection in **XXE**:

- text selection (also called character selection),
- node selection (a node is either a named element or an anonymous text node or an anonymous comment node or an anonymous processing instruction node),
- implicit element selection.

2.1.1. Text selection

The text selection is the type of selection supported by all text editors and word processors.

It is, of course, possible to select text across document nodes.

Selected text is displayed as characters drawn over a pink background.

The text selection can be seen as a way to specify a range of characters and descendant nodes contained *in a common ancestor node*. The copy, cut, delete, paste, convert commands will copy, delete or replace the specified characters and descendant nodes.

- Example 1: "`<p>This is our <img src="logo.png"> new logo.</p>`". Selected text starts at "`This`" and ends after "`new`" and is to be deleted.

  This delete operation gives "`<p> logo.</p>`".

- Example 2: "`<p>This is <em>great</em>, really!</p>`". Selected text starts at "`This`" and ends after "`great`" and is to be converted to `<strong>`.

  This convert operation gives "`<p><strong>This is <em>great</em></strong>, really!</p>`".

How to select text in XXE:

Using the mouse	Using the keyboard
• Click at the beginning of the text selection and drag the mouse to the end of selection.	• **Shift**+Left or **Shift**+Right extends the text selection by one character.

Using the mouse	Using the keyboard
	• **Ctrl**+**Shift**+Left or **Ctrl**+**Shift**+Right extends the text selection by one word. • **Ctrl**+**Shift**+Home extends the text selection to the beginning of the line. • **Ctrl**+**Shift**+End extends the text selection to the end of the line. • **Shift**+Down or **Shift**+Up extends the text selection by one line.

As expected, `Shift-click` may be used to extend the text selection. However if there is a node selection (see below), `Shift-click` will extend the node selection instead.

2.1.2. Node selection

The node selection is unique to the XML editor.

In **XXE**, you can even select several nodes at the same time if these nodes are consecutive children of the same parent (contiguous range of child nodes).

Selected nodes are displayed with a thin red border around them.

How to explicitly select a *single node* in **XXE**:

Using the mouse	Using the keyboard
• Click on the node. If some text is near the place where you have clicked, it will ``attract'' the caret and the node will not be selected. In such case, you are forced to Ctrl-click instead of simply clicking. • Or Ctrl-click on the node. Ctrl-click several times (without moving the mouse) if needed to. Each Ctrl-click will select the parent of currently selected node. Do not Ctrl-click several times too fast otherwise the editor will think you are double-clicking or triple-clicking and therefore, selecting elements that way would not work. • Or click on generated content such as a list bullet or a section number. • Or Ctrl-click in the blank space found at the right of the text of a paragraph.	• Click inside the node to move the caret there. Type **Ctrl**+Up until you reach the node you want to select. Keep an eye on the Node Path bar [54] while doing this. • Or use the Select menu [20] or any of the corresponding keyboard shortcuts **Ctrl**+Up, **Ctrl**+Down, **Shift**+**Ctrl**+Up or **Shift**+**Ctrl**+Down.

Using the mouse	Using the keyboard
• Or click on the name of the node you want to select in the Node Path bar [54].	

Procedure for selecting a *node range* using the *mouse*:

1. Select first node using any of the methods described above.

   Make sure that you have selected the right node by looking at the Node Path bar [54], otherwise extending the node selection will not work.

2. `Shift-click` on the last node of the selection. If there is no ambiguity, you can even `Shift-click` *anywhere* past the last node of the selection.

Procedure for selecting a *node range* using the *keyboard*:

- **Esc**+**Down** selects all child nodes of explicitly or implicitly selected element.
- OR

  1. Select first node using any of the methods described above.

     Make sure that you have selected the right node by looking at the Node Path bar [54], otherwise extending the node selection will not work.

  2. Adjust selected node range: **Esc**+**Right** extends node selection to following sibling and **Esc**+**Left** extends node selection to preceding sibling.

  Note **Esc**+**Right** (and **Esc**+**Left**) will first select element containing caret if there is no explicit node selection, therefore typing **Esc**+**Right** several times is often the quickest way to select a node range.

2.1.3. Implicit element selection

The implicitly selected element is simply the element containing the *caret* (also called the insertion cursor).

Being implicitly selected, this element is not drawn with a red border around it but you know it because it is the element which is displayed in the Node Path bar [54].

All editing commands except the most generic form of Split [25] and Join [25] can be applied to the *implicitly selected element*.

Almost all editing commands do not require you to explicitly select the element you want to act upon. *This makes **XXE** at the same time efficient and easy to use.*

2.2. Select menu items

Clicking anywhere in the document view or any caret movement has the side effect to cancel the selection. It is also possible to explicitly do so by typing **Esc**+**Esc**.

Select Parent

Selects parent of currently selected node.

If there is no currently selected node, selects the text, comment or processing instruction node containing the caret.

Select Child

Selects previously selected child of currently selected node.

If there is no such child, selects first child node of currently selected element.

If currently selected node is a text, comment or processing instruction node, cancels the selection.

Select Preceding Sibling

Selects preceding sibling of currently selected node, if any.

Select Following Sibling

Selects following sibling of currently selected node, if any.

Extend Selection to Preceding Sibling

Adds preceding sibling, if any, of currently selected node to the node selection.

If there is no currently selected node, selects the element containing the caret.

Keyboard shortcut (not displayed in menu): **Esc**+**Left**.

Extend Selection to Following Sibling

Adds following sibling, if any, of currently selected node to the node selection.

If there is no currently selected node, selects the element containing the caret.

Keyboard shortcut (not displayed in menu): **Esc**+**Right**.

Select All Children

Selects all children of currently selected element.

Keyboard shortcut (not displayed in menu): **Esc**+**Down**.

3. Edit menu

A simplified, contextual, version of the **Edit** menu is displayed if you right-click in a document view.

Also note that a right-click may be used to select the element clicked upon prior to displaying the popup menu. More precisely:

- If you right-click outside the existing text or node selection or if there is no selection, this selects (explicitly if really needed to, otherwise implicitly) the *element* clicked upon.
- If you right-click inside the existing text or node selection, the selection is not changed.

Undo

Undo last command.

Redo

Redo last undone command.

Repeat

Repeats last repeatable command.

Commands requiring the user to specify an argument (e.g. Replace [24], Insert Before [24], Insert [24], Insert After [24], Convert [24], Wrap [25], Change Processing Instruction Target [27], etc) are repeatable.

Command History

Displays a dialog box listing last ten repeatable commands from newest to oldest.

Cut

Cuts
- text selection
- OR explicitly selected node or node range
- OR implicitly selected element

to system clipboard.

It is possible to cut and paste nodes between two instances of XXE (of course if the DTD or schema allows it).

Copy

Copies
- text selection
- OR explicitly selected node or node range
- OR implicitly selected element

to the system clipboard.

Selected characters are automatically copied as system selection on platforms supporting system selection (X-Window) and automatically copied to an internal clipboard on other platforms.

Copy as Text

Copies *as plain text* explicit text or node selection to the system clipboard.

Paste Before

Pastes the content of system clipboard before
- explicitly selected node or node range
- OR implicitly selected element.

The system clipboard may contain one or several XML nodes or just plain text.

Paste

Pastes the content of system clipboard replacing
- text selection
- OR explicitly selected node or node range,

OR if there is no explicit selection, pastes the content of system clipboard into
- element containing caret, at caret position.

The system clipboard may contain one or several XML nodes or just plain text.

Clicking with mouse button #2 (middle button or mouse wheel) can be used to paste the content of system selection on platforms supporting system selection and can be used to paste the content of an internal clipboard on other platforms (if allowed by grammar constraining the document, of course).

By default, this very handing feature is not enabled. You need to enable it using the Options dialog box [113].

Paste After

Pastes the content of system clipboard after
- explicitly selected node or node range
- OR implicitly selected element.

The system clipboard may contain one or several XML nodes or just plain text.

Delete

Deletes
- text selection
- OR explicitly selected node or node range
- OR implicitly selected element.

Force Deletion

Like Delete [23] except that deletion will be performed even if the grammar constraining the document forbids to do so.

*Example 3.1. A use case for command **Force Deletion***

The content model of element <a> is child element <b> or a sequence of child element <c> followed by child element <d>.

A new <a> is by default created with the simplest possible content model, that is <b>. Then how to replace <b> by the sequence <c><d>? Deleting <b> is forbidden because it would give us an invalid <a>.

The answer is:
1. Force the deletion of <b> using the command described here. This makes <a> temporarily invalid but also relaxes the constraints on it.
2. Insert a <c>.
3. Insert a <d>. Element <a> is now valid.

This item is by default absent in the **Edit** menu. You need to explicitly enable it by checking "**Enable the 'Edit|Force Deletion' menu item**" in **Options** → **Preferences**, **General**|**Features** section.

Replace

Displays the Edit tool [59]. This dialog box can be used to specify an element or a text node replacing
- explicitly selected node or node range
- OR implicitly selected element.

Insert Before

Displays the Edit tool [59]. This dialog box can be used to specify an element or a text node to be inserted *just before*
- explicitly selected node or node range
- OR implicitly selected element.

Insert

Displays the Edit tool [59]. This dialog box can be used to specify an element or a text node to be inserted into the element containing the caret, *right here, at caret position*.

Insert After

Displays the Edit tool [59]. This dialog box can be used to specify an element or a text node to be inserted *just after*
- explicitly selected node or node range
- OR implicitly selected element.

Convert

Displays the Edit tool [59]. This dialog box can be used to specify an element or a text node replacing
- text selection
- OR explicitly selected node or node range
- OR implicitly selected element.

Unlike Replace [24] which creates an *empty* new element, Convert transfers the content of the selection to the new element which is the result of the conversion.

More precisely, in the case of the node selection:

- When a single element is selected, all its children, and also all compatible attributes, are transferred to the result of the conversion.

  Example:

  ```
  <simpara id="p1">The <emphasis>little</emphasis> lamb.</simpara>
  ```

  converted to `para` gives

  ```
  <para id="p1">The <emphasis>little</emphasis> lamb.</para>
  ```

- When several nodes or a single non-element node are selected, all these nodes are given a new parent element which is the result of the conversion.

  Example:

```
<simpara>Once upon a time,</simpara>
```

plus

```
<simpara id="p1">the <emphasis>little</emphasis> girl.</simpara>
```

can be converted to `blockquote` and that gives us

```
<blockquote>
  <simpara>Once upon a time,</simpara>
  <simpara id="p1">the <emphasis>little</emphasis> girl.</simpara>
</blockquote>
```

See also Wrap [25] a variant of Convert.

Wrap

This command is a variant of Convert [24]. The unique difference between Wrap and Convert [24] is that, with Wrap, when a single element is selected, the selected element is given a new parent element.

Example, with Wrap (and not with Convert [24]), it is possible to give a `<blockquote>` parent to the following `<simpara>`, when this `<simpara>` is implicitly or explicitly selected:

```
<simpara id="p1">The <emphasis>little</emphasis> lamb.</simpara>
```

That is, selecting `blockquote` using the Edit tool [59] gives:

```
<blockquote>
  <simpara id="p1">The <emphasis>little</emphasis> lamb.</simpara>
</blockquote>
```

Split

Splits explicitly selected element in two parts, the split point being specified by caret position.

Unlike almost all other commands, this command requires the element to be explicitly selected.

A less generic form of the **Split** command is *often* bound to key Enter (for example, this is the case for XHTML, DITA and DocBook).

Typing Enter inside a paragraph or a list item will split this element in two parts.

Join

Joins explicitly selected element to its preceding sibling, an element of same type. This gives a single element containing the child nodes of the two joined elements. This command is the inverse command of **Split** [25].

Unlike almost all other commands, this command requires the element to be explicitly selected.

A less generic form of the **Join** command is *often* bound (for example, this is the case for XHTML, DITA and DocBook) to
- key Backspace when the caret is at the beginning of a paragraph or a list item
- and to key Delete when the caret is at the end of a paragraph or a list item.

Typing Backspace at the beginning of a paragraph or a list item joins this element to the preceding paragraph or a list item.

Typing Delete at the end of a paragraph or a list item joins this element to the following paragraph or a list item.

3.1. Comment menu

Insert Comment Before

Inserts comment node before
- explicitly selected node
- OR implicitly selected element.

Insert Comment

Inserts comment node into
- explicitly selected *empty* element
- OR element containing caret, at caret position.

Insert Comment After

Inserts comment node after
- explicitly selected node
- OR implicitly selected element.

Comment Out

Replaces text selection or explicit node selection by a comment containing the selection.

Uncomment

Inverse command of **Comment Out** [26]. Parses the content of implicitly or explicitly selected comment and replaces this comment by parsed XML nodes.

3.2. Processing instruction menu

Insert Processing Instruction Before

Inserts processing instruction node before
- explicitly selected node
- OR implicitly selected element.

A newly inserted processing instruction uses the last target interactively specified using menu item "**Change Processing Instruction Target**". If menu item "**Change Processing Instruction Target**" has not yet been used, then the target of a newly inserted processing instruction is "`target`".

Insert Processing Instruction

Inserts processing instruction node into
- explicitly selected *empty* element
- OR element containing caret, at caret position.

Insert Processing Instruction After

Inserts processing instruction node after
- explicitly selected node
- OR implicitly selected element.

Change Processing Instruction Target

Displays a dialog box that can be used to change the target of
- explicitly selected processing instruction node
- OR implicitly selected processing instruction node (that is the processing instruction node containing the caret).

3.3. Reference menu

Copy as Reference

Copies to the clipboard a *reference* to the selected nodes (i.e. a pointer to selected nodes). This reference can be later pasted into another document[3], using any of the normal paste commands — **Edit** → **Paste Before** (**Ctrl**+**U**), **Edit** → **Paste** (**Ctrl**+**V**), **Edit** → **Paste After** (**Ctrl**+**W**) — in places where the grammar constraining the target document allows to do so.

The pasted reference cannot be edited in place. It is displayed with a light gray background to clearly indicate this. Use menu item **Edit** → **Document Reference** → **Edit Referenced Document** [29] (**Ctrl**+**Shift**+**E**) to open a new window allowing to edit the document containing the referenced nodes.

This command is enabled only for documents associated to a configuration declaring a *inclusion scheme* (see Section 17, “inclusionScheme” in *XMLmind XML Editor - Configuration and Deployment*). DITA documents use the `conref` inclusion scheme. DocBook, XHTML and TEI Lite documents use the XInclude inclusion scheme.

By default, it is possible to copy as a reference only an element having an ID attribute or the root element of a document. However, for documents using the XInclude inclusion scheme, this restriction may be relaxed by using **Options** → **Preferences**, **Edit** section, **Allow advanced use of XInclude** [114] checkbox. When this option is turned on, it becomes possible to copy as a reference any range of sibling nodes.

It is not possible to copy as reference the text selection, a reference, descendant nodes of a reference or any node selection directly containing one or more references. If you want to do so, simply use the normal **Edit** → **Copy** command, as this command preserves[4] existing references.

Shift-clicking menu item "**Copy as Reference**" invokes command "`copyAsInclusion in XMLmind XML Editor - Commands [implicitElement]`**`[multipleInstances]`**" which copies as a reference some contents intended to be pasted to several different locations in the same document. For now, only DocBook 5.1 documents support the "`[multipleInstances]`" option. For any other type of document, option "`[multipleInstances]`" is simply ignored.

[3] It can also be pasted in the same document at another location.
[4] But, unlike **Edit** → **Reference** → **Copy as Reference**, it cannot *create* references.

Include Text

Displays a file chooser dialog box allowing to choose a text file (of any kind: XML, HTML, `.bat`, C/C++, etc). The content of this text file is then included in the document being edited at caret position.

This kind of inclusion is implemented by the means of an `<xi:include parse="text">` element. Therefore, this command is disabled unless the document being edited supports the XInclude inclusion scheme.

Untransclude Reference

Replaces included nodes by the inclusion directive (e.g. `xi:include` element).

Any kind of selection inside the included nodes suffices to specify the subject of this command.

Untransclude All

Replaces all the included nodes found in the document being edited by the corresponding inclusion directives.

Retransclude Reference

Inverse action of Untransclude Reference [28]: replaces inclusion directive (e.g. `xi:include` element) by up-to-date included nodes.

The inclusion directive must be explicitly or implicitly selected.

Using untransclude allows to edit the inclusion directive by hand before retranscluding it. This is useful in the two following cases:

- This allows to add attributes (typically an ID) to the `conref` elements created using **Copy As Reference** then **Paste**.
- This allows to fine tune the `xpointer` attribute of the XIncludes created using **Copy As Reference** then **Paste**. Example: replace `xpointer="xpointer(id('disclaimer')/*[position() >= 1 and position() <= 8])"` by, simpler and more stable, `xpointer="xpointer(id('disclaimer')/*)"`.

Retransclude All

Replaces all the inclusion directives found in the document being edited by the corresponding included nodes.

Edit Referencing Document

If current document is referenced by another document already opened in XXE and displayed in another window, this command brings the window of this other document to front. If there is no such referencing document, this command is silently disabled.

Example 3.2. Example: book referencing chapters

Book `book.xml` references chapters `chap1.xml`, `chap2.xml`, `chap3.xml`, etc, created in separate documents.

Clicking anywhere inside first chapter displayed in the `book.xml` window then using menu item **Edit** → **Reference** → **Edit Referenced Document** [29] (**Ctrl**+**Shift**+**E**) brings the window containing `chap1.xml` to front. (If needed, `chap1.xml` is opened in XXE.)

Now being inside the `chap1.xml` window, using menu item **Edit → Reference → Edit Referencing Document** [28] (**Ctrl**+**Shift**+**B**) brings the window containing `book.xml` to front.

Edit Referenced Document

If the caret or the selection is inside a reference to an element contained in another document, this command brings the window of this other document to front. If the referenced document is not yet opened in XXE, this command will open it.

See example [28] above.

Shift-click on this menu item to open referenced document in read-only mode.

4. Search menu

Search

Displays the **Search** tool [69] configured for a search session from the current caret position[5] to the end of the document.

Keyboard shortcut: **Ctrl**+**F** (**F** like **F**ind).

Search Backwards

Displays the **Search** tool [69] configured for a search session from the current caret position[5] to the beginning of the document.

Keyboard shortcut: **Ctrl**+Shift+**F** (**F** like **F**ind).

Replace

Displays the **Search** tool [69] configured for a search/replace session from the current caret position[5] to the end of the document.

Keyboard shortcut: **Ctrl**+**M** (**M** like **M**odify).

Replace Backwards

Displays the **Search** tool [69] configured for a search/replace session from the current caret position[5] to the beginning of the document.

Keyboard shortcut: **Ctrl**+Shift+**M** (**M** like **M**odify).

Find Next

Search last searched string from the current caret position to the end of the document.

Keyboard shortcut: **Ctrl**+**G** (**G** like a**G**ain).

The text selection, if any, is automatically used to initialize the **Search** field of the **Search** tool [69]. As a consequence, in some cases, you don't need to use the **Search** tool at all. For example, selecting some text in the document and then directly pressing

[5] Turning off the "**Start from current caret position**" option allows to start searching from the beginning of the document (or from the end of the document depending on the **Direction** option).

Ctrl+**G** (**Search** → **Find Next**) is the quickest way to find the next occurrence of the selected text in the document.

Find Previous

Search last searched string from the current caret position to the beginning of the document.

Keyboard shortcut: **C**trl+Shift+**G** (**G** like a**G**ain).

4.1. The "Find and Replace Element" menu

Find Element

Displays the "**Find Element**" dialog box [97] ready to be used to search elements from explicitly or implicitly selected node to the end of the document.

Keyboard shortcut: **Esc** f (**F** like **F**ind).

Find Element Backwards

Displays the "**Find Element**" dialog box [97] ready to be used to search elements from explicitly or implicitly selected node to the beginning of the document (backwards).

Keyboard shortcut: **Esc** F.

Find and Replace Element

Displays the "**Find and Replace Element**" dialog box [96] ready to be used to search and replace elements from explicitly or implicitly selected node to the end of the document.

Keyboard shortcut: **Esc** r (**R** like **R**eplace).

Find and Replace Element Backwards

Displays the "**Find and Replace Element**" dialog box [96] ready to be used to search and replace elements from explicitly or implicitly selected node to the beginning of the document (backwards).

Keyboard shortcut: **Esc** R.

Find Next Element

Repeat last specified search operation starting at explicitly or implicitly selected node and ending at the end of the document.

Keyboard shortcut: **Esc** g.

Find Previous Element

Repeat last specified search operation starting at explicitly or implicitly selected node and ending at the beginning of the document (backwards).

Keyboard shortcut: **Esc** G.

Replace Next Element

Repeat last specified search and replace operation starting at explicitly or implicitly selected node and ending at the end of the document.

Keyboard shortcut: **Esc** h.

Replace Previous Element

Repeat last specified search and replace operation starting at explicitly or implicitly selected node and ending at the beginning of the document (backwards).

Keyboard shortcut: **Esc** H.

4.2. The "Web Search" menu

The entries of this menu invoke the web browser in order to perform a web search on the current text selection. For example, if you select text "`Pi`" in the document view and select menu item **Wikipedia**, this will open a window or tab in your web browser showing the definition of `Pi` in Wikipedia.

The entries of this menu are not fixed. They may be configured using **Options** → **Preferences**, **Web Search** section [115].

Using keyboard shortcuts for searching the web

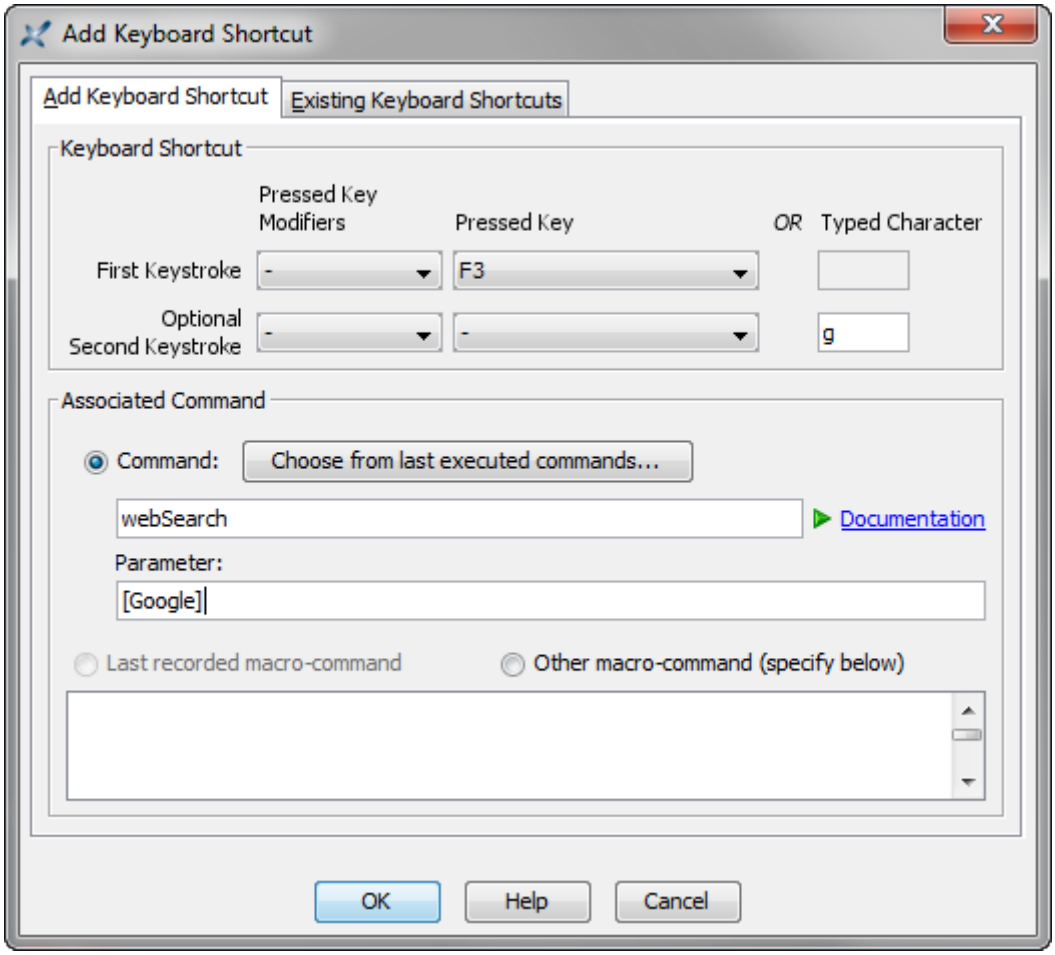
Note that the entries of this menu have no keyboard shortcuts. Instead, you need to press the underlined character found in the title of the menu item (the “hotkey”[6]). For example, If you want to perform a web search using **Google**, you must press **Alt**+**S** (for menu **Search**) then **W** (for submenu **Web Search**) then **G**.

If you prefer to use a real keyboard shortcut, then you'll have to define it using **Options** → **Customize Configuration** → **Add Keyboard Shortcut**.

When you'll do this, note that the name of the command to be invoked is `webSearch` and that its parameter is the name of the web search service put between square brackets. Example: `[Google]`.

[6] Not available on the Mac.

*Figure 3.1. Perform a web search using Google by pressing **F3** and then g*



5. View menu

The view settings of the document being edited specified using **Text Size** [33], **Display Images** [33], **Show Tags** [34], **Tree View Details** [34], **Add Views** [34] and the CSS stylesheet menu items [34] are automatically made persistent across editing sessions.

Redraw

Rebuilds the view of the whole document, no matter which text or nodes are selected.

Updating the references contained in a modular document

This command has also the side-effect of automatically updating *all* the references contained in a modular document (see Copy as Reference [27]). However for a reference to be updated, the referenced document must not have been modified in **XXE** and not yet saved to disk.

Example: document `book.xml` references `chap1.xml`, `chap2.xml` and `chap3.xml`. File `chap1.xml`, currently opened in **XXE**, has been modified then saved to disk. File `chap2.xml` is not currently opened in **XXE**. File `chap3.xml`, currently opened in **XXE**, has been modified but not yet saved to disk. Using this command will update what has

been included from `chap1.xml` and `chap2.xml`, but not from `chap3.xml`, before rebuilding the view of `book.xml`.

Text Size

This menu makes it easy changing the base font size of the active document view. Applies to both tree views and styled views.

Use your *mouse wheel* while pressing the **Ctrl** key (**Cmd** key on the Mac) rather than use the **Larger** and **Smaller** menu items.

Larger

Use a base font size larger by 2pt for the active document view. The largest possible base font size is 24pt.

Smaller

Use a base font size smaller by 2pt for the active document view. The smallest possible base font size is 8pt.

Normal

Use the “normal”, default, base font size (typically 12pt).

Display Images

You may need to enable this submenu by checking "**Enable the View|Display Images Submenu**" in **Options** → **Preferences**, **General|Features** section.

Specifies how images are to be displayed in the active document view. Applies only to styled views.

as Thumbnails

Show the bounding box of the (possibly scaled) image and, if there is enough room in this rectangle, also show a thumbnail (at most 128x128 pixels large).

as Bounding Boxes

Show the bounding box of the (possibly scaled) image.

Normally

Display the image normally, that is, possibly scaled if this has been specified using attributes such as `width` and `height`.

- Options "**as thumbnails**" and "**as bounding boxes**" have no effect on images having an intrinsic size smaller than 32x32 pixels. Such small images are always displayed normally.
- If the bounding box is sufficiently large, it is used to display the thumbnail (when option "**as thumbnails**" has been selected) and useful information about the image (file size, image size, etc).

  However, when the bounding box is small, some information may be elided. Therefore do not be surprised if you have selected option "**as thumbnails**" and that for some images, you don't see any thumbnail. Also do not be surprised if some bounding

boxes contain useful information about the image and other (smaller) bounding boxes are completely empty.

Show Tags

Specifies whether element tags are to be displayed in the active document view. Applies only to styled views.

Show Tags

Show most element tags.

Also Show Table Tags

Show all element tags, including those related to tables, that is, row group, row and cell tags.

Tree View Details

Specifies whether attributes, text node characters, comment characters, processing-instruction characters, are to be displayed in their entirety in the active document view. Applies only to tree views.

Show Attributes

Toggles the display of the attributes in the tree view.

Show Characters in Text Nodes

Toggles the display of text node characters in the tree view.

Show Characters in Comment Nodes

Toggles the display of comment node characters in the tree view.

Show Characters in Processing-instruction Nodes

Toggles the display of processing-instruction node characters in the tree view.

Add Views

XXE allows to edit a document using several views at the same time, for example, the styled view, a TOC view and a tree view. Except for the XML source view [34], these views are “synchronized in real-time” (that is, reflect the same document content in real-time).

This menu item displays a dialog box which may be used to add new views to current document tab and/or to remove existing views from the document tab. Up to four views may be added at the top, bottom, left or right sides of the central, main view.

CSS stylesheet menu items

Below the above menu item, a menu item is added for each CSS stylesheet available for current document. Selecting the name of a stylesheet causes the document view to use this stylesheet. If a document view already uses the selected stylesheet, the stylesheet is reloaded from its file (which is very handy when developing a new CSS stylesheet).

Among these CSS stylesheet menu items, you may find an item called "**XML source**". This menu item behaves slightly differently from the other items. See below [34].

5.1. The "XML source" menu item

This menu item is available only when the "**Edit source**" add-on has been installed.

The "**Edit source**" add-on adds an "**XML Source**" item at the end of the **View** menu. Selecting this item replaces the current styled or tree view by a source view of the document being edited.

The source view of a document shows the contents of the save file which would be created by **XXE** for this document (same automatic indentation, same named character entities, etc).

The source view of a document embeds an advanced text editor having the following capabilities.

- Syntax highlighting.
- Element folding.
- Auto-completion (after pressing **Ctrl**+SPACE) for element names, attribute names, attribute enumerated values and named character entities.
- "Not well-formed" errors are reported in really time.

Limitations:

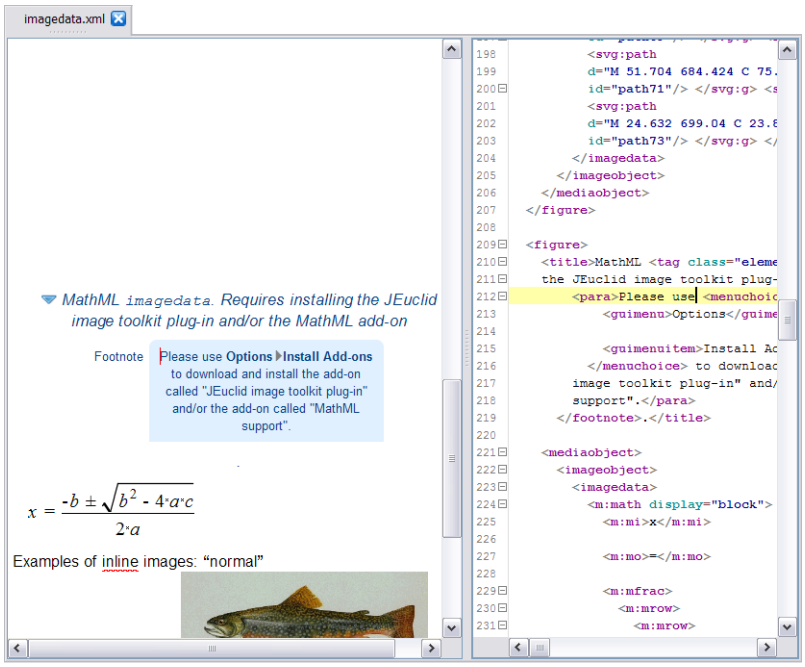
- Validity errors are *not* reported in really time.
- Auto-completion suggests all the child element names regardless of whether this would make the parent element invalid. For example, auto-completion will suggest child element `title` even when the `chapter` parent element already has a `title`.

In addition to the keyboard shortcuts found in all text editors, the source view also supports the following shortcuts:

Keyboard Shortcut	Description
Ctrl+SPACE (**Ctrl** even on the Mac)	Auto-completion for element names, attribute names, attribute enumerated values and named character entities. Element names Type '<' then press **Ctrl**+SPACE. Attribute names Press **Ctrl**+SPACE inside an open tag (e.g. after *foo* inside <*foo* . . .>). Named character entities Type '&' then press **Ctrl**+SPACE.
Ctrl+**F** (Cmd on the Mac)	Find. Press **Ctrl**+Shift+**F** to find backwards.
Ctrl+**G**	Find again. Press **Ctrl**+Shift+**G** to find again backwards.
Ctrl+**M**	Replace. Press **Ctrl**+Shift+**M** to replace backwards.
Ctrl+Shift+**L**	Goto line.

The source view is not “synchronized in real-time” with the other views of the same document

Being an (almost) normal view of an XML document, it's possible to use **View** → **Multiple Views** → **Add View** to have, for example, the styled view of a document and its source view side by side.



However, while all the tree and styled views of a document are “synchronized in real-time” (that is, reflect the same document content in real-time), this is not the case of the source view. For performance reasons, you'll have to save the document to disk (or at least check the document for validity) to synchronize a source view with the other views.

For example, if you click in the styled view of a document (left side in the above figure), modify it and save the document to disk, the source view of the document will be automatically updated. Note that if before doing that, the source view was modified, you'll loose these unsaved changes.

Conversely, if you click in the source view of a document (right side in the above figure), modify it and save the document to disk, the styled view of the document will be automatically updated. Note that if before doing that, the styled view was modified, you'll loose these unsaved changes.

In a nutshell, if you use the styled view of a document side by side with other views, always remember to save your document to disk (**Ctrl**+**S**; or at least check the document for validity: **Ctrl**+Shift+**V**) before switching from one view to an other.

6. Tools menu

Use as Master Document

This checkbox may be absent in the **Tools** menu. If this is the case, you need to explicitly enable it by checking "**Enable the 'Tools|Use as Master Document' menu item**" in **Options** → **Preferences**, **General**|**Features** section.

The following documents may be used as *master documents*: DITA map, DocBook 5.1 assembly, Ebook, DocBook modular documents (for example, a book including chapters by the means of XInclude).

Declaring a document as being a master document creates a *document set* containing the master document and its module document. This document set is automatically updated when the master document is saved to disk.

By grouping a master document and its module documents, you inform XMLmind XML Editor (**XXE**) that all the module documents referenced or included, directly or indirectly, by the master document are related. When **XXE** knows that some of the opened documents are related:

- it will more thoroughly check the cross-references which may exist between these documents;
- it will make it easier creating cross-references between these documents;
- it will make it easier following cross-references between these documents;
- if a DITA map contains key definitions, then this map acts not only as a cross-reference creation/validation context for its topics, but it also acts as a *key space*;
- if the "**Easy Profiling**" add-on has been installed, then the conditional processing profile selected for the master document is automatically shared by all module documents;
- if the document view area is split in two parts, the module documents opened from a master document will appear at the opposite of this master document. This allows to use the view of the master document as a rudimentary navigation pane.

The list of master documents is persistent across editing sessions.

Automatic master documents

Note that, by default, DITA maps, DocBook 5.1 assemblies and Ebooks (that is, all kinds of maps) are *automatically made master documents*.

If for some maps, you don't want this to happen, simply uncheck "**Tools** → **Use as Master Document**". When this is done, the map is added to the list of exceptions called "**Should NOT be automatically made master documents**". See Section 6.8.1, “Master document options” [119].

Example 3.3. Example explaining the enhanced cross-reference creation/validation provided by a master document

Modular book `mybook.xml` includes 3 chapters. Each chapter is found in its own file: `chapterA.xml`, `chapterB.xml` and `chapterC.xml`.

File `chapterA.xml` contains:

```
<chapter id="chapterA">
  <title>Chapter A</title>

  <para>Link to <link linkend="nowhere">nowhere</link>. Link to <link
  linkend="sectionB1">Section B1</link>.</para>

  <section id="sectionA1">
    <title>Section A1</title>

    <para>TODO.</para>
```

```
  </section>

  <section id="sectionA2">
    <title>Section A2</title>

    <para>TODO.</para>
  </section>
</chapter>
```

In the above file, the first `link` element points to a non-existent target and the second `link` points to the first section of `chapterB.xml`.

The user wants to check the links found in `chapterA.xml` and also to add an `xref` element pointing to the first section of `chapterC.xml`. In order to do that, she/he opens `chapterA.xml` in **XXE**.

Before using `mybook.xml` as a master document:

- The **Validity** tool reports 2 cross-reference warnings: `reference to non-existent ID "nowhere"` and `reference to non-existent ID "sectionB1"`.
- When the user inserts an `xref` element and specifies its `linkend` attribute, the **Attributes** tool suggests: `chapterA`, `sectionA1`, `sectionA2`.

After using `mybook.xml` as a master document:

- The **Validity** tool reports 1 cross-reference warning: `reference to non-existent ID "nowhere"`.
- When the user inserts an `xref` element and specifies its `linkend` attribute, the **Attributes** tool suggests: `chapterA`, `sectionA1`, `sectionA2`, `chapterB`, `sectionB1`, `sectionB2`, `chapterC`, `sectionC1`, `sectionC2`.

Declare Namespace

Displays Declare Namespace dialog box [101].

If the current document is conforming to a DTD, the dialog box allows to view the namespaces and their prefixes but not to modify them.

Edit Attribute

Displays Attributes tool [64].

Check Validity

Displays Validity tool [71], unless no validity errors are found in current document, in which case an OK message is displayed in the status bar.

This command is disabled if current document is not constrained by a grammar.

Current document validity is automatically checked each time the document is saved, therefore unless you are fixing an invalid document you don't really need to explicitly use this command.

Execute Command

For advanced users only. Displays a dialog box which allows to choose a command by name. This is needed when a command is not bound to a keystroke, menu item or tool bar button. Example:

command `convertCase` (see Section 15, “convertCase” in *XMLmind XML Editor - Commands*). This facility is especially useful when recording a macro-command [44].

This menu item is hidden by default. You need to enable it by checking "**Enable the 'Tools|Record Macro' Submenu**" in **Options** → **Preferences**, **General**|**Features** section.

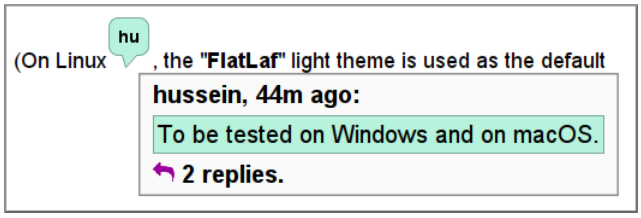
6.1. Remark menu

You may need to enable this submenu by checking "**Enable the 'Tools|Remark' Submenu**" in **Options** → **Preferences**, **General**|**Features** section.

What is a remark?

A remark is simply a `<?xxe-remark>` processing-instruction nicely rendered as a balloon in the styled view.

Figure 3.2. A remark made by user `hussein` and its tooltip.



This balloon contains the first two letters of the username (login) of the author of the remark.

This balloon is given a random background color, which is specific to the author of the remark. For example, the color of all the remarks created by user `hussein` will always be green, but there is no way for user `hussein` to choose the color of his own remarks.

Hovering the mouse over this balloon displays the contents of the remark in a tooltip.

Double-clicking on this balloon displays a remark editor. Which remark editor is displayed depends on whether this editor is used to create a new remark or to reply to/modify an existing remark.

Figure 3.3. Remark editor #1 displayed when a new remark is to be inserted in the document

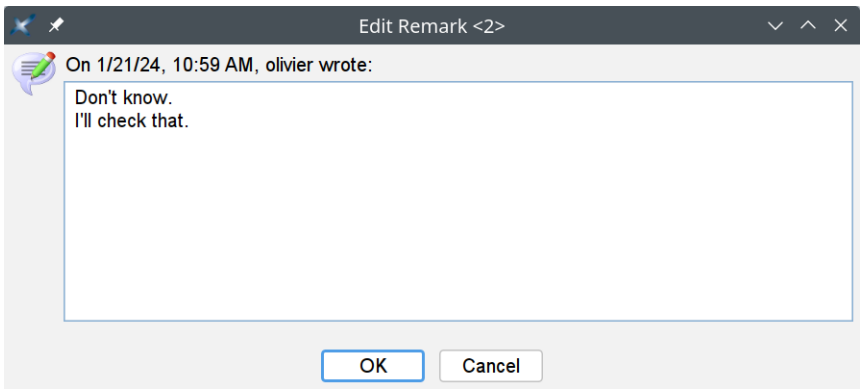
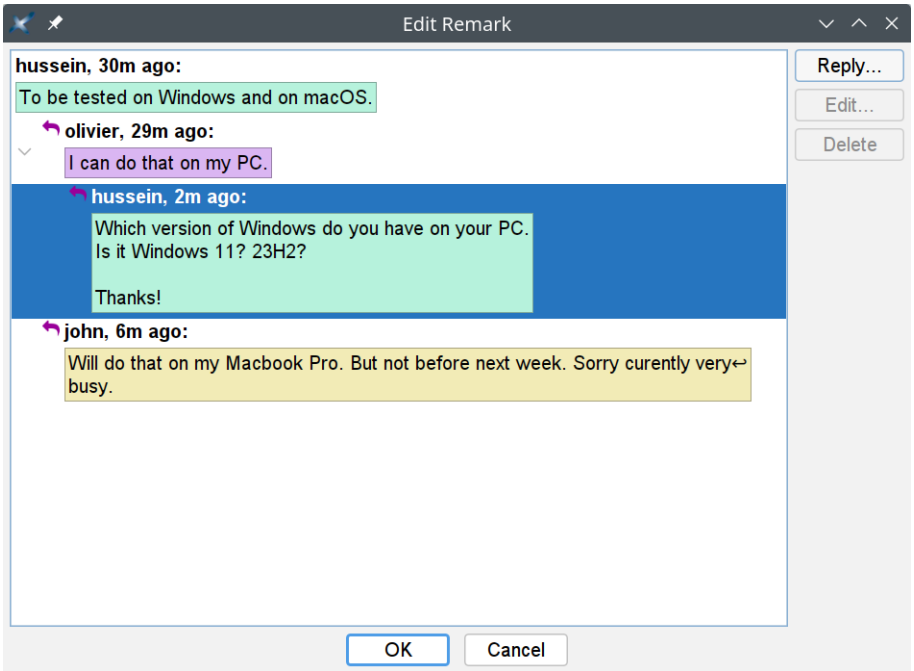


Figure 3.4. Remark editor #2 lets the user reply to a remark made by another user or modify her/his own remark or delete it



The Remark menu

Insert or Edit Remark

If a remark is explicitly selected or if the caret is contained in a remark, display the remark editor #2 which lets the user reply to a remark made by another user or modify her/his own remark or delete it.

Otherwise display the remark editor #1 in order to create a new remark. The new remark is inserted before the explicit selection if any, or at caret position otherwise.

Delete Remark

Delete explicitly selected remark, if any. Otherwise if the caret is contained in a remark, delete this remark.

Delete All Remarks

Delete all remarks.

Select Preceding Remark

Select preceding remark, if any.

Select Following Remark

Select following remark, if any.

6.2. Revisions menu

You may need to enable this submenu by checking "**Enable the 'Tools|Revisions' Submenu**" in **Options** → **Preferences**, **General|Features** section.

Do Not Keep Any Information About Revisions

This menu item turns off both "**Enable the Comparison of Revisions**" and "**Store All Revisions in the Document**" options.

Enable the Comparison of Revisions

This option instructs **XXE** to add information[7] to the document being edited in order to allow comparing two revisions of this document.

Turn this option on if you need to compare two revisions of the same initial document. However, for that, you'll have to keep each revision in its own XML file. In practice, this option is useful only when your documents are stored in a CMS, a document repository, on a versioning file system or more simply, if you often archive copies of the XML sources of your documents.

Store All Revisions in the Document

This option instructs **XXE** to start storing all the revisions[8] of the document being edited in the XML file containing this document.

Turning on this option implies turning on option "**Enable the comparison of revisions**". If all revisions are stored in the XML file, then you'll be able to use open them using **Tools** → **Revisions** → **Open Revision** [41] and to compare them using **Tools** → **Revisions** → **Compare Revisions** [42].

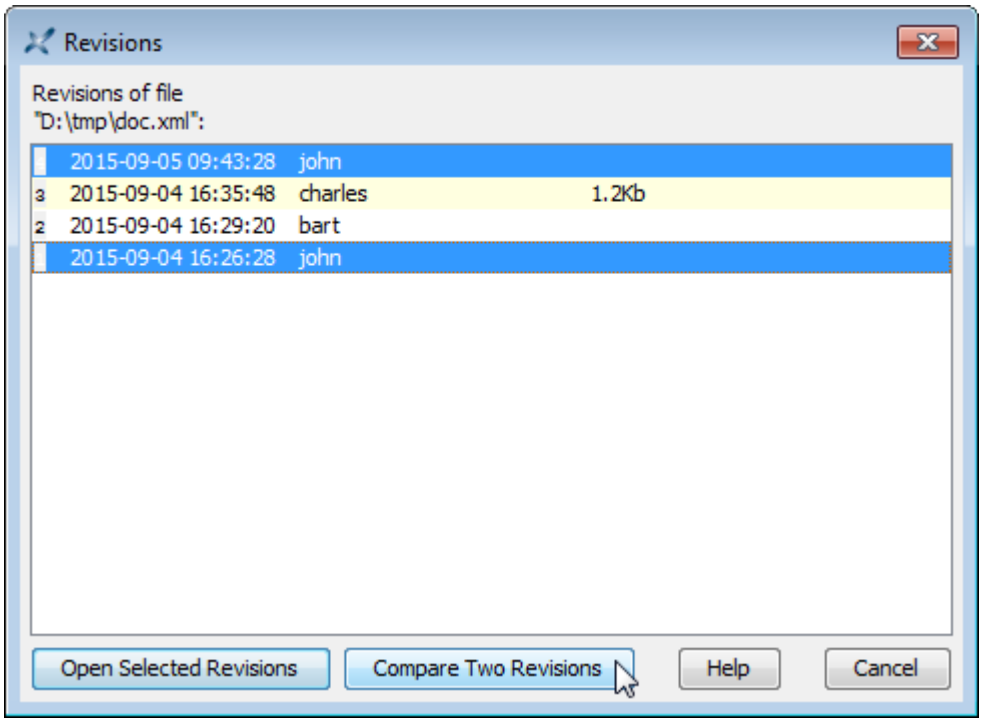
Turn this option on before sending your document to reviewers. Then, after using the the **Compare** tool [84] to review the changes made to your document, turn this option off.

Open Revision

Displays a dialog box [77] listing all the revisions of the active document. This dialog box allows to select one or more revisions in order to open or compare these revisions.

[7]More information about what's needed to be able to compare two revisions of the same initial document using the **Compare** tool [84] in Appendix A, *Description of the XML differencing algorithm implemented by the **Compare** tool* in *XMLmind XML Editor - Commands*.
[8]More information about how revisions are stored in the XML file in Appendix B, *Format of the revision history* in *XMLmind XML Editor - Commands*.

Figure 3.5. The "Open Revision" dialog box, here allowing to compare the document being edited (revision #4; latest one) to the initial revision (revision #1)



This menu item is disabled (“grayed”) when option "**Store All Revisions in the Document**" has not been chosen for the active document. This active document may be the document being edited (considered to be “latest revision”; opened in normal read/write mode) or an older revision of the document being edited (always opened in read-only mode).

Compare Revisions

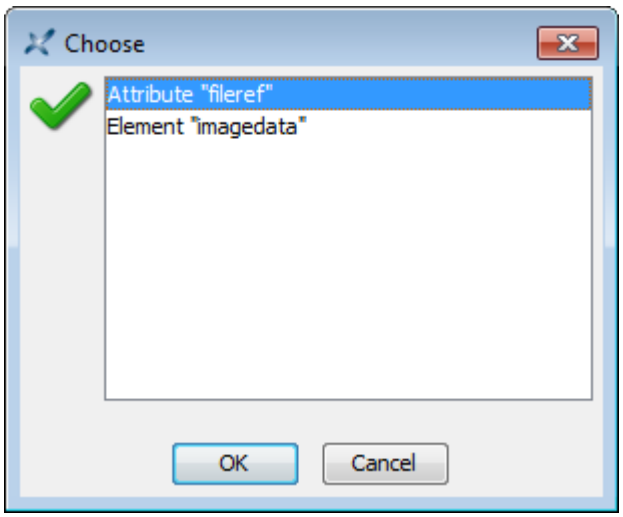
Displays the **Compare** tool [84] at the left of the main window. The **Compare** tool will show you the differences existing between two revisions of the same initial document. It also allows to accept or reject some or all the changes.

6.3. "Helper Application" menu

This menu contains entries allowing to use third-party applications to view or edit part or all of the document being edited.

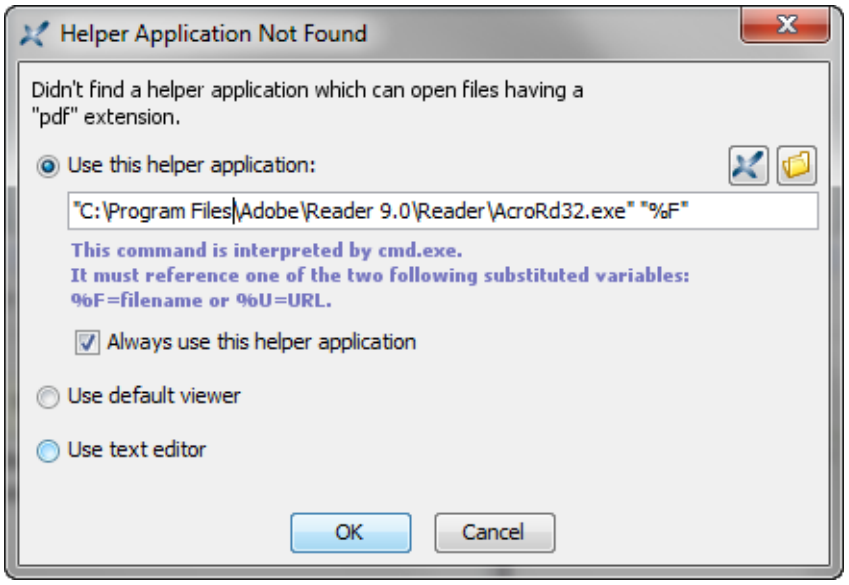
Open Element or Attribute Content in Helper Application

Requires an element to be implicitly or explicitly selected. Displays a dialog box allowing the select this element or one of its attributes from a list.



Opens specified element or attribute in a third-party ``helper'' application. If the helper application cannot be determined automatically (because it has not yet been registered using the **Preferences** dialog box, **Helper Applications** section), the user is prompted to specify it.

*Figure 3.6. The "**Helper Application Not Found**" dialog box*



Example of use: use your web browser to open the page referenced in the `url` attribute of the (DocBook 4) `ulink` element.

Edit Element or Attribute Content in Helper Application

Same as above except that the helper application is assumed to be an editor. If this editor is used to modify the element or attribute, then the changes are also automatically applied to the document being edited.

This command works as follows: let's suppose the element of interest contains an image encoded using base 64 (data type `xs:base64Binary`).

1. This command examines the first bytes of the image and, using this signature, determines which helper application to use.

2. If the helper application cannot be determined (because it has not yet been registered using the **Preferences** dialog box, **Helper Applications** section), the user is prompted to specify it.

3. It reads the image data from the element, decodes it and saves it to a temporary file.

4. It starts the helper application, an image editor, passing it the file containing the extracted image.

5. After the user quits the image editor, the command detects whether the extracted image has been modified and, if this is the case, reloads it in the element.

Examples of use: use GIMP to edit the image file referenced in the `src` attribute of the (XHTML) `img` element. Use Inkscape to edit the `svg:svg` element contained in an (DocBook 5) `imagedata` element.

Open Document in Helper Application

Opens the document being edited in a third-party ``helper'' application. If the helper application cannot be determined automatically (because it has not yet been registered using the **Preferences** dialog box, **Helper Applications** section), the user is prompted to specify it.

This entry is disabled if the document is newly created and has not yet been saved. Also note that this command is guaranteed to fail if the document is stored on the *Windows file system* and is currently write-locked by XMLmind XML Editor.

Example of use: use your web browser to preview the XHTML document being edited.

Edit Document in Helper Application

Same as above except that the helper application is assumed to be an editor. XMLmind XML Editor will automatically reload the document after the third-party editor is used to modify it.

This entry is disabled if the document is newly created and has not yet been saved. Also note that this command is guaranteed to fail if the document is currently write-locked by XMLmind XML Editor.

Example of use: use your favorite text editor to perform some low-level modifications on the XML document being edited.

6.4. "Record Macro" menu

You may need to enable this submenu by checking "**Enable the 'Tools|Record Macro' Submenu**" in **Options** → **Preferences**, **General|Features** section.

This menu allows to record a sequence of commands and to replay the recorded sequence at will.

This facility used in conjunction with **Select** → **Find Element** [20] facility or with the Search [69] tool may be seen as an advanced, versatile, yet simple to use, form of search/replace.

Start

Starts recording a sequence of commands.

Stop

Stops recording the sequence of commands.

Cancel

Cancels the recording of a sequence of commands.

View

Displays a dialog box containing recorded macro in XML form. Very handy to paste it in an XXE configuration file (see Section 4, “command” in *XMLmind XML Editor - Configuration and Deployment*).

Replay

Replays recorded sequence of commands.

Procedure 3.1. Procedure for recording a sequence of commands

1. Use the **Start** menu item.

2. Invoke commands as usual: use key bindings, menu items and tool bar buttons.

   Typing some text is of course supported. ``Tabbing" from a text node to another is supported too.

   The following tools can also be used during a recording: Edit [59], Attributes [64], Search [69], Characters [62].

   If you use the **Search** tool during a recording, just use the search part; do not use the replace part. And after the searched string is found and selected, click on the **Stop** button of the **Search** tool to give keyboard focus back to the document view. This will allow you to continue adding more commands to the recorded sequence.

3. Use the **Stop** menu item.

Only *editing commands* can be recorded. That is, commands that modify the contents and/or the selection marks of the document being edited. Actions such as **File** → **Open** or **View** → **Add** cannot be recorded.

At most 20 commands can be recorded. Typing contiguous characters, no matter how many, counts as a single command.

Attempting to record the following commands will automatically cause macro recording to be canceled:

- any command triggered by a mouse click,
- Undo [22], Redo [22], Repeat [22],
- any command which fails (example: searching a string and this string is not found),
- any command which cannot be executed given current editing context (example: pasting some text to a place where the schema forbids to do so).

Recording interactive command such as Insert After [24] works as expected: it is the command *along with the element interactively chosen by the user* which is recorded, and not the interactive invocation of **Edit**+**Insert After** (i.e. which activates the **Edit** tool or which displays the equivalent dialog box).

Recording command Execute Command [38] is fully supported and works as expected: it is the command executed by **Tools** → **Execute Command** which is recorded, and not the invocation of **Tools** → **Execute Command**.

7. XML menu

The **XML** menu is a *placeholder* for a menu in *XMLmind XML Editor - Configuration and Deployment*, defined in an **XXE** configuration file, which specific to an XML document type.

For example, such specific menu is defined for the DITA topic document type. When a DITA topic is opened in **XXE**, the **XML** menu is automatically populated with items ("**Paste As**", "**Insert or Edit indexterm**", etc) and its title changes from **XML** to **Topic**.

8. Options menu

Preferences

Displays the Preferences dialog box [106] which allows to specify a large number of user preferences.

Install Add-ons

Displays the Install Add-ons dialog box [170], which lets you download, install, upgrade and uninstall add-ons.

This menu item is absent when XMLmind XML Editor has been started using Java™ Web Start.

If you hold the Shift key and click on **Options** → **Install Add-ons**, you'll *automatically* upgrade user-installed add-ons (of course, if any and if needed to). More information in Section 14.1, "*Automatically* upgrading installed add-ons" [173].

8.1. "Customize Configuration" menu

This menu allows end-users (i.e. non-experts) to customize the configuration associated to the document being edited.

End-user customization of configurations by the means of this menu also works when XMLmind XML Editor is deployed using Java™ Web Start.

The following menu items have an effect on a *single* configuration: the configuration of the document being edited. For example, if you specify DocBook XSL stylesheet parameter `paper.type=USLetter` when you are editing a DocBook document, this will have no effect on the DocBook 5 and DocBook 5.1 configurations, even if these configurations share their **Convert Document** menu with the DocBook configuration.

Using any of the following menu items has an immediate effect on all the opened documents associated to the customized configuration. That is, no need to restart **XXE** to benefit from the customization.

Save Document As Template

Save the document being edited as a named template. This named template is then displayed in the **File** → **New** dialog box.

This menu item is disabled (grayed) if the document being edited has been modified and thus, needs to be saved. Therefore you need to save the document being edited to disk prior to using this facility.

Arbitrarily complex documents may be used as templates: a document may be modular, may reference graphics files, etc. This is needed for example to create a document template including the logo of a company.

User-defined document templates are added and removed using the "**Save Document As Template**" dialog box [140].

Save Selected Element As Template

Save the selected element as a named template. This named template may then be suggested by the **Edit** tool, depending on the operation to be performed and if this is allowed by the schema.

This menu item is disabled (grayed) unless an element is explicitly selected.

User-defined elements templates are added and removed using the "**Save Selected Element As Template**" dialog box [143].

Add Keyboard Shortcut

Add keyboard shortcuts. In other terms, bind up to two keystrokes (example: "`F2 u`" to a command (example: "`moveElement up[implicitElement]`").

User-defined keyboard shortcuts are added and removed using the "**Add Keyboard Shortcut**" dialog box [146].

Customize CSS Stylesheet

Allows customizing the CSS stylesheet being used by the active styled document view.

Custom CSS rules are added and removed using the "**Customize CSS Stylesheet**" dialog box [152].

Use Current View Settings as Default

Save the view settings of the document being edited as the defaults for *all* the documents of the same type (e.g. all DITA Topic documents). Note that the view settings of the document being edited may be modified using the items of the **View** [32] menu.

Customize Document Conversion Stylesheets

Using this menu item is a relatively simple way to influence the layout and style of the deliverable (PDF, RTF, HTML, etc) which results from the document conversion.

The document being edited is converted to other formats by the means of *XSLT stylesheets*. This menu item allows to:

- select an XSLT stylesheet other the default one,
- create a custom XSLT stylesheet on the fly,
- invoke a specialized editor —XMLmind XSL Customizer [163]— to modify a user-created XSLT stylesheet.

However, when the document being edited is converted to an *HTML-based format* (Web Help, EPUB, HTML Help, etc)[9], the HTML pages which are automatically generated by the aforementioned XSLT stylesheets are styled mainly by *CSS stylesheets*. When this is the case, this menu item allows additionally to:

- select a CSS stylesheet other the default one,
- create a custom CSS stylesheet on the fly,
- invoke a helper application [121] (generally, a text editor) to modify a user-created CSS stylesheet.

[9] As opposed to XSL-FO based formats (PDF, RTF, .docx, .odt, etc).

This menu item displays the "**Customize Document Conversion Stylesheets**" dialog box [155].

9. Tab menu

Split Document Area Horizontally

Clicking on the dashed line found at the bottom of a document tab causes the document area to be split in two parts. This lets you see two documents side by side.

If this option is turned on, the document area is split horizontally. If this option is turned off (the default), the document area is split vertically.

Note that turning this option on and off has an immediate effect on the document area if this area has already been split in two parts.

Default: not checked.

Close

Closes active document.

Close All

Closes all opened documents.

Show Preceding

Displays preceding (in the order the documents have been opened or created) document tab.

Show Following

Displays following (in the order the documents have been opened or created) document tab.

Below the above menu items, a menu item is added for each document opened in **XXE**. Selecting the name of a document in this list causes this document to become the active one and thus, causes the corresponding tab to be displayed.

10. Help menu

Getting Started

Displays a short tutorial in the help browser.

Full Tutorial

Displays the Tutorial page, which is part of the XMLmind XML Editor web site, in the Web browser.

Help

Displays the help browser.

Help About

In order to display the help section related to a specific GUI component of **XXE**, first execute this command (the cursor changes signaling that you are now in contextual help mode) and then click on that component.

Full Documentation Set

Displays the Documentation page, which is part of the XMLmind XML Editor web site, in the Web browser.

Show Element Reference

Opens in the Web browser the reference documentation of the explicitly or implicitly selected element. Note that this menu item is enabled only if a reference manual which is usable for this purpose (e.g. DocBook, DITA) is found on the Web.

Show Content Model

Opens a window containing an hypertext reference manual listing all elements and attributes specified in the DTD, W3C XML Schema or RELAX NG schema of the document being edited.

This manual, which is organized like "DocBook: The Definitive Guide" by Norman Walsh and al., is intended to help content authors understand the DTD or schema of the document being edited.

Check for Updates

Displays a dialog box informing you about the availability of a version of **XXE** which is newer than the one you are currently using.

This dialog box contains check box "**Automatically check for updates**". This option is turned on by default. It allows to automatically check for updates[10]. This automated check is performed behind the scene, every 24 hours, once per editing session, 2 minutes after **XXE** has been started.

About XMLmind XML Editor

Displays the customary **About** dialog box.

The **Copy** button found in the **About** dialog box copies to the clipboard extensive information about the runtime environment of **XXE**. Please do not forget to attach this information to the bug reports you send us.

[10]This feature is really unobtrusive and does not transmit *any* data to our servers.

Chapter 4. The document view area

Figure 4.1. Tab corresponding to the active document

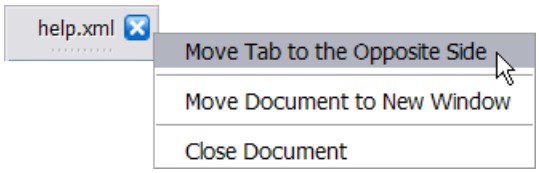


Figure 4.2. Tab corresponding to the active document when it is in read-only mode

Notice that the tab of a read-only document has a permanent *dark gray* indicator at its left. This contrasts with the tab of an editable, read-write, document which is given a *green* indicator when this document has been modified and the changes need to be saved to disk.

There are several ways to directly open a document in read-only mode. Example: Shift+click on **File** → **Open** [15] menu item or on the **Open** button of the main tool bar [52].

The tabs of documents included or referenced in the same *master document* [36] are all given the same background color.

Items of the right-click popup menu:

Move Tab to the Opposite Side

Moves this tab to the opposite side of the document view area. When several documents are opened in XMLmind XML Editor, this has the effect to split the document view area in two parts. See also **Window** → **Split Windows Horizontally** [48].

This is equivalent to clicking on the dashed line found at the bottom of a tab.

Move Document to New Window

Opens a new main window and moves the active document to this new window. See also **Window** → **New Window** [15].

Dragging a tab and dropping it outside any **XXE** window (e.g. onto the desktop) also opens a new main window and moves the active document to this new window.

This menu item is available only after the "**New Window**" feature has been enabled using **Options** → **Preferences**.

Close Document

Closes the active document and switches to the next document opened in XMLmind XML Editor, if any.

This is equivalent to clicking on the blue cross found at the right of the tab.

Drag & drop of document tabs and tool [59] tabs

- It's possible to change the order of tabs using drag & drop. To do that, simply select a tab, drag it and drop it between before first tab, between two tabs or after last tab. Possible drop locations are rendered using green rectangles.
- Dragging a tab and dropping it to a side not having tabs allows to split the tabbed pane in two and to move the dragged tab to the split part of the container.

  The other ways to split a tabbed pane are to click the “tear-off” dashed line found at the bottom of a tab OR to select item "**Move Tab to the Opposite Side**" from the right click menu of a document tab.
- Dragging a tab and dropping it outside any **XXE** window (e.g. onto the desktop) has the same effect as clicking **Move Document to New Window** [50]. It opens a new main window and moves the active document to this new window.
- It's also possible to drag a document tab and drop it in a different **XXE** main window.

  A new **XXE** main window is created by selecting menu item **Window** → **New Window**. This menu item is available only after the "**New Window**" feature has been enabled using **Options** → **Preferences**.

Chapter 5. The main tool bar

1. Buttons which are common to all document types

New

See **File** → **New** [15].

Open

See **File** → **Open** [15].

Shift-click on this button to open a document in read-only mode.

Save

See **File** → **Save** [16].

Save All

See **File** → **Save All** [16].

Undo

See **Edit** → **Undo** [22].

Redo

See **Edit** → **Redo** [22].

Repeat

See **Edit** → **Repeat** [22].

Command History

See **Edit** → **Command History** [22].

Cut

See **Edit** → **Cut** [22].

Copy

See **Edit** → **Copy** [22].

Paste Before

See **Edit** → **Paste Before** [22].

Paste

See **Edit** → **Paste** [23].

Paste After

See **Edit** → **Paste After** [23].

Delete

See **Edit** → **Delete** [23].

Find and Replace Element

See **Search** → **Find and Replace Element** → **Find and Replace Element** [30].

Insert or Edit Remark

See **Tools** → **Remark** → **Insert or Edit Remark** [40].

2. Buttons which are specific to type of the document being edited

The main tool bar also contains buttons which are specific to the type of the document being edited. For example, when a DITA topic is opened in **XXE**, you'll also find "**Toggle i**", "**Toggle b**", "**Convert to plain text**", etc, buttons.

Such buttons are specified in configuration element tool bar in *XMLmind XML Editor - Configuration and Deployment* found in an **XXE** configuration file (".xxe" file).

Chapter 6. The node selection bar

1. The node path bar

The node path bar displays the *path* of

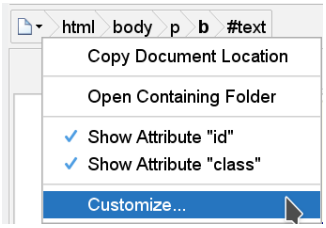
- selected node if a single node has been selected
- OR first selected node if a node range has been selected
- OR node containing the caret otherwise (even if there is a text selection).

The path of a node is

- the name of the element if the node is an element,
- `#text` for a text node,
- `#comment` for a comment node,
- `#processing-instruction` for a processing instruction node,

preceded by the path of its parent element.

The node path bar is a very convenient tool for selecting and acting on nodes:

User interaction in the node path bar	Effect
Click on a node name.	Select this node.
Shift-click on a element name.	Select all the child nodes of this element.
Right-click on a node name.	Select this node then display a simplified, contextual, version of the **Edit** menu [21].
Drag the file icon .	Drag and drop in another application, the *location* of the document being edited in XXE. Do not forget to save the document being edited in **XXE** before dropping its location in a viewer application. See also Drag and drop in XMLmind XML Editor [17].
Right-click the file icon .	Displays a contextual menu containing the following entries:  **Copy Document Location** Copy to the clipboard the location (file path or URL) of the document being edited.

User interaction in the node path bar	Effect
	Open Containing Folder Open in *the native file manager* the folder containing the document being edited. **Show Attribute "*attribute_name*"** Checkbox which may be used to instruct the node path bar to hide or show the value of attribute "*attribute_name*". This attribute must have been declared using the **Customize** dialog box introduced below. More information in Section 1.3, “Configuring the node path bar to display the value of some attributes” [56]. **Customize** Displays a dialog box letting the user specify which attribute values are to be displayed by the node path bar. More information in Section 1.3, “Configuring the node path bar to display the value of some attributes” [56].

1.1. Non-editable nodes

The path of a non-editable node is displayed using a blue-gray color. A non-editable node has been pasted in the document being edited after using command Copy as Reference [27].

In order to modify such node, one must edit the document containing it in a separate window. Use menu item **Edit** → **Reference** → **Edit Referenced Document** [29] (**Ctrl**+**Shift**+**E**) for that.

1.2. Elements which are in non-validating mode

Normally, XMLmind XML Editor works in validating mode. In such mode, the author can only do the editing operations which are allowed by the DTD or schema. For example, the author cannot remove required attributes (unless he/she uses **Force Removal**).

When an element is found to be invalid, XMLmind XML Editor automatically switches to a lenient editing mode for this element and all its descendants.

After the invalid element is fixed by the author, XMLmind XML Editor automatically switches back to its normal, validating, mode.

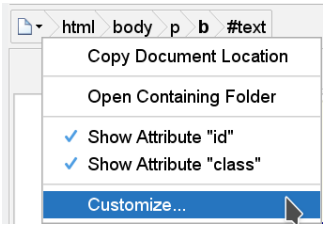
- An element underlined in *orange* means that this element is in *non-validating mode 1*. In this mode, XMLmind XML Editor still suggests the right attributes and child elements to the author. But these are only suggestions: the author may add and remove any attributes and child elements he/she wants, and this, at any place and in any number.

- An element underlined in *red* means that this element is in *non-validating mode 2*. In this mode, XMLmind XML Editor is not able to suggest the right attributes and the right child elements to the author. The author may add and remove any attributes and child elements he/she wants, at any place and in any number.

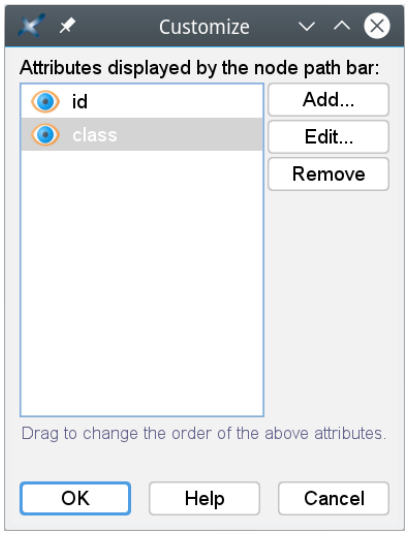
Note that, in the case of certain RELAX NG schemas, XMLmind XML Editor may automatically switch to a lenient editing mode, *even if the underlined element is perfectly valid*. This behavior is explained in great details in *XMLmind XML Editor - Support of RELAX NG Schemas*.

1.3. Configuring the node path bar to display the value of some attributes

The "**Customize**" item of the contextual menu of the “file icon” found at the beginning of the node path bar



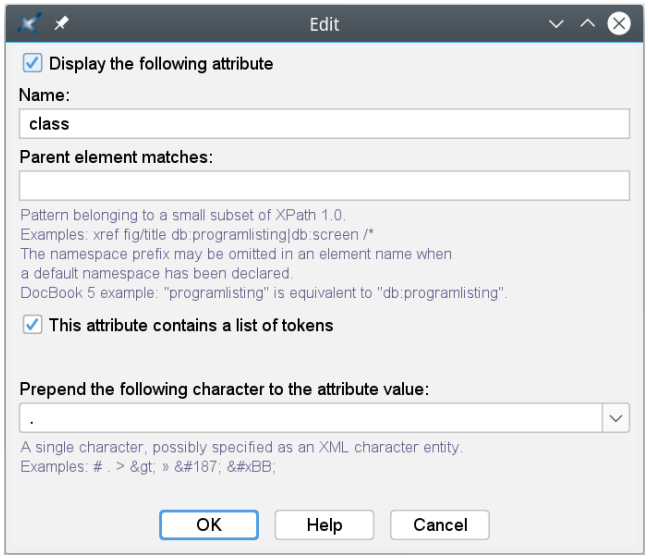
displays the following dialog box:



This dialog box may be used to configure the node path bar to make it display one or more attribute values after some element names. XHTML example: **html** > **body** > **pre#example.fancy.line-numbers** > **code**. Character "#" is used to prefix the value of attribute `id`. Character "`.`" is used to prefix the tokens comprising the value of attribute `class`.

Click **Add** to specify an attribute to be displayed by the node path bar. Click **Edit** to change selected attribute specification. Click **Remove** to delete selected attribute specification.

Clicking **Add** or **Edit** displays the following attribute specification editor:



Display the following attribute

Check this checkbox to make the node path bar display specified attribute.

Note that the contextual menu of the “file icon” found at the beginning of the node path bar contains a number of checkboxes letting the user quickly switch an attribute from visible to hidden and the other way round.

Name

The name of the attribute to be displayed.

Parent element matches

An XPath pattern matching the parent element of the attribute to be displayed. Leave field empty to specify: any parent element.

This attributes contains a list of tokens.

Check this checkbox to specify that the attribute contains a list of tokens. When this is the case, the node path bar splits this list and displays each token preceded by the character specified using the combobox below.

Prepend the following character to the attribute value

Specifies the character to be prepended to the attribute value.

2. Navigation tool bar

Most commands which causes **XXE** to “jump” from one location to another automatically leave an invisible “back mark” behind them. Such marks, specifying a location in a document, are recorded in the navigation history. Menu items **Go Back** and **Go Forward** allow the user to navigate back and forth between the current location of the caret and these back marks.

The commands which automatically leave a back mark behind them are:

- **Edit** → **Reference** → **Edit Referenced Document** [29] and **Edit** → **Reference** → **Edit Referencing Document** [28].
- Text search and replace commands such as **Search** → **Search** [29], **Search** → **Find Next** [29], etc.

- Find and replace element commands such as **Search** → **Find and Replace Element** → **Find Element** [30], **Search** → **Find and Replace Element** → **Find and Replace Element** [30], etc.
- The **Validity** tool [71].

The navigation history is common to all the documents opened in **XXE**. For example, clicking **Go Back** may cause **XXE** to jump to another document. And if this other document has been closed by the user, **XXE** will attempt to reopen it.

The navigation history is automatically cleared after closing all the documents opened in **XXE**.

Go Back

Go back in the navigation history.

Go Forward

Go forward in the navigation history.

Navigation

Displays a menu containing the following entries:

Show History

Display a dialog box containing all the entries of the navigation history. Clicking on an entry closes the dialog box and then causes **XXE** to jump to the corresponding location.

Remember this Location

Add the current location of the caret to the navigation history.

Clicking this menu item is rarely needed as most commands which causes **XXE** to jump from one location to another automatically leave an invisible back mark behind them.

Chapter 7. Tools

It is possible to display two tools at the same time. By default, XMLmind XML Editor displays both the **Edit** tool and the **Attributes** tool. If you don't like this layout or if your screen resolution is too low to allow this, simply click on the “tear-off” dashed line found in the **Edit** tab. This will move the **Edit** tool back to the pane containing all the other tools.

More generally, clicking on the “tear-off” dashed line found in a tab may be used to split the tabbed pane in two parts and/or move a tab from one part to the other. Of course, the divider used to separate the two parts may be dragged to resize these parts.

See also Drag & drop of document tabs and tool tabs [51].

1. Edit tool

This tool works in two steps:

1. Click on one of the following buttons:

   Replace

   (See **Edit** → **Replace** [24] for a complete description of the command.)

   Insert Before

   (See **Edit** → **Insert Before** [24] for a complete description of the command.)

   Insert

   (See **Edit** → **Insert** [24] for a complete description of the command.)

   Insert After

   (See **Edit** → **Insert After** [24] for a complete description of the command.)

   Convert

   (See **Edit** → **Convert** [24] for a complete description of the command.)

   Wrap

   (See **Edit** → **Wrap** [25] for a complete description of the command.)

   This first action just fills the list below these buttons with all the allowed arguments (element name or text node) for the selected command.

2. Choose an element name from the list or type its name in the text field. It is this second action which actually triggers the command.

How to specify an argument for one of the editing commands described above

- An element is specified by clicking on its name in the list. A single click is sufficient.

  If the selected operation allows to specify a text node as its argument, the list contains not only element names but also a "`(text)`" item.

- Alternatively, the element name can be specified using the text field above the list:

  1. Type the element name in the field (or type "`(text)`" if this item is found in the list).

  2. Press Enter or click on the **OK** button.

The text field supports auto-completion.

This auto-completion feature can be configured using the Options dialog box [113].

- Type **Esc** or click on the **Cancel** button to cancel current command and to clear the form.

Procedure 7.1. Procedure for specifying an element argument when the command is applied to an element of the unconstrained type

1. Type any element name in the text field.

   If the namespace of the element name has not been declared, you will have to declare it using **Tools** → **Declare Namespace** [38] before being allowed to specify this name.

2. Press Enter or click on the **OK** button.

What is an element of the *unconstrained type*?

The type of an element is unconstrained if

- the element is contained in a document which is not constrained by a grammar,
- OR the content of the element is invalid according to the grammar used to constrain the document.
- OR the content of an ancestor of the element is invalid according to the grammar used to constrain the document.

In all cases, **XXE** allows the user to add or remove any child element or text node.

In the second case, fixing the invalid element will automatically make it recover its actual, constrained, type.

Procedure 7.2. Procedure for specifying an element argument that matches an element wildcard

1. Type an element name that matches the wildcard in the text field.

   If the namespace of the element name has not been declared, you will have to declare it using **Tools** → **Declare Namespace** [38] before being allowed to specify this name.

2. Press Enter or click on the **OK** button.

   XXE will of course check that the name you have typed conforms to the element wildcard.

What is a *element wildcard*?

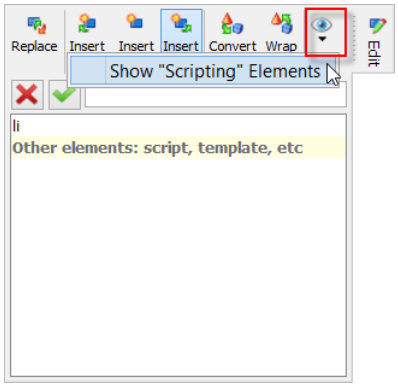
Using a W3C XML Schema or a RELAX NG schema (but not a DTD), it is possible to specify an element type which constrains some of its child elements just to belong to a certain set of namepaces and nothing more.

The set of allowed namespaces is called a wildcard.

1.1. Reducing the number of elements displayed by the Edit tool

Clicking the “eye button” found at the right of the tool bar of the **Edit** tool displays a menu.

*Figure 7.1. Menu when an XHTML5 document is opened in **XXE** (inserting an element after a `li`)*



This menu allows to toggle the visibility of elements belonging to certain categories. Simply uncheck a menu entry to “hide” in the **Edit** tool all the elements belonging to the corresponding category.

Even when an available element has been made hidden, it's still possible to specify it by typing its name (no auto-completion though) in the text field of the **Edit** tool.

Configuration Families	Element Categories (*Hidden* by default unless stated otherwise)
XHTML5	**Scripting** `script`[a], `noscript`, `template`, `slot`.
DITA Topic	**Programming** `syntaxdiagram`, `msgblock`, etc.
DocBook	**Programming** `classsynopsis`, `cmdsynopsis`, `constructorsynopsis`, etc[b]; `productionset`, `msgset`, etc.

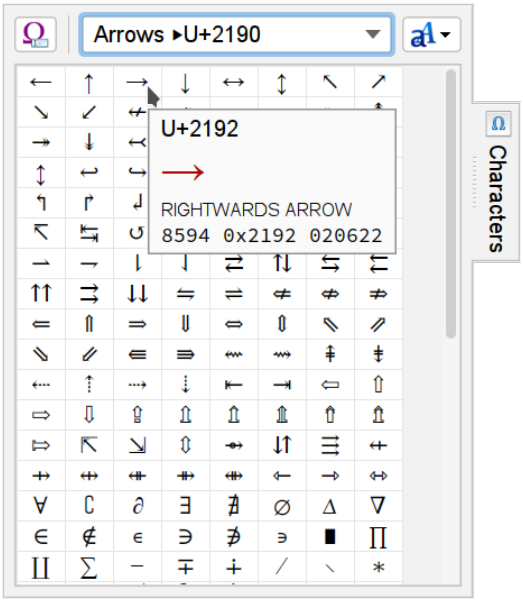
[a] Always visible when *ancestor* element is `head`.

[b] Always visible when *ancestor* element is `refentry`.

2. Characters tool

Click on a character to insert[1] it at caret position in the document being edited.

*Figure 7.2. The **Characters** tool*



In addition to the combobox which may be used to select and display a range of Unicode characters, this tool has two buttons:

Insert special character by name

Displays a dialog box letting the user select and insert at caret position a character specified using its entity name (e.g. `mdash`).

[1]Note that this tool honors the *Overwrite Mode* (**OVR**) [89] when this mode has been turned on.

*Figure 7.3. The dialog box displayed by the "**Insert special character by name**" button*

Note that the command invoked by this button, `insertCharByName` in *XMLmind XML Editor - Commands* (keyboard shortcut **Esc** n) does not insert a *reference* to a character entity (e.g. `—`); it inserts a *character* (e.g. U+2024). It must be considered as a handy alternative to using the **Characters** tool.

Change font

Displays the following menu:

Change Font

Change the font used by the **Characters** tool. This menu item displays a dialog box letting the user choose any font present on your system. The chosen font is automatically saved as a user preference.

Revert to Default Font

Revert to the font used by default by the **Character** tool (which is generally `Serif-14`).

2.1. The “Favorites” palette

The last item of the Unicode range combobox is called **Favorites**. It can be used to select and display a custom palette of up to 256 characters. This palette is populated by right-clicking on a character and by choosing the "**Add to Favorites**" item in the following menu:

Copy

Copy to the clipboard the character upon which the user has right-clicked.

Add to Favorites

Add to the **Favorites** palette the character upon which the user has right-clicked.

Remove from Favorites

Remove from the **Favorites** palette the character upon which the user has right-clicked.

Show Favorites

A quick way to switch to the **Favorites** palette.

3. Attributes tool

- There are two methods for adding or changing the attributes of the (explicitly or implicitly) selected element:
  1. Using the attribute form (the upper side of the attribute tool).This should be the method of choice for persons who prefer to use the keyboard.
  2. Using the attribute table (the lower side of the attribute tool). This should be the method of choice for persons who prefer to use the mouse.

- Adding an attribute for an element of the *unconstrained type* must be done using the attribute form. Simply type the name and value of the attribute in the attribute form and click on the **OK** button (or press **Enter** in the value field).

What is an element of the *unconstrained type*?

The type of an element is unconstrained if
- the element is contained in a document which is not constrained by a grammar,
- OR the content of the element is invalid according to the grammar used to constrain the document.

In both cases, XXE allows the user to add or remove any attribute, the value allowed for these attributes being any string.

In the latter case, fixing the invalid element will automatically make it recover its actual, constrained, type.

- Adding an attribute that matches an *attribute wildcard* must be done using the attribute form. Simply type the name and value of the attribute using the attribute form and click on the **OK** button (or press **Enter** in the value field). XXE will of course check that the name you have typed conforms to the attribute wildcard.

What is a *attribute wildcard*?

Using a W3C XML Schema or a RELAX NG schema (but not a DTD), it is possible to specify an element type which, for example, has 3 ``ordinary'' attributes `a1`, `a2`, `a3` but also any number of other attributes if the name of these extra attributes belong to certain namespaces, for example, the "`http://www.w3.org/1999/xlink`" namespace.

The set of allowed namespaces is called an attribute wildcard.

3.1. Using the attribute table

The content of the attribute table can be described as follows:

- All attributes set for the selected element are displayed by the table.
- By default[2], all possible attributes for the selected element, *even those not set*, are also displayed by the table.

  Attributes which have not been set are displayed in gray. Attributes which have been set are displayed in black.
- Attributes are listed sorted in alphabetical order.
- The names of required attributes are displayed using a bold font.
- The names of fixed attributes are displayed using an italic font.

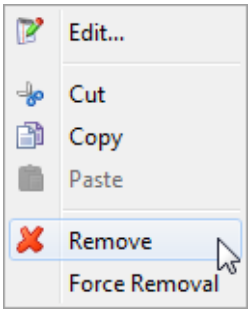
Procedure for adding an attribute or changing its value:

1. Click on the field at the right of the attribute name. This field is editable.
2. Type the value of the attribute.

   If the attribute type is enumerated, this field is a menu rather than a text field, so you can directly choose the value of the attribute without having to type anything.
3. Press **Enter** to commit the change.

Procedure for removing an attribute:

1. Right-click on the row of the attribute to be removed.
2. Select the **Remove** item in the displayed popup menu.



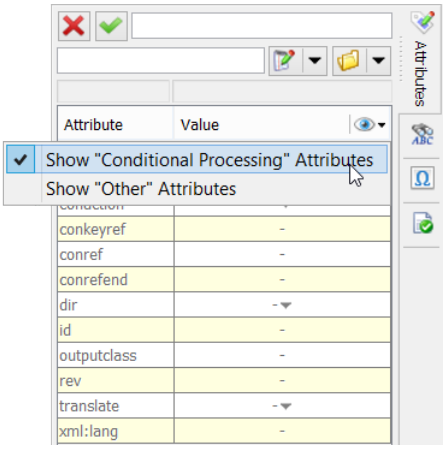
The "**Force Removal**" item is by default absent in the above popup menu. You need to explicitly enable it by checking "**Enable the 'Edit|Force Deletion' menu item**" in **Options** → **Preferences**, **General**|**Features** section.

3.1.1. Reducing the number of attributes displayed by the attribute table

Clicking the “eye button” found at the right of the header of the attribute table displays a menu.

[2]This can be configured, see Section 3.1.1, “Reducing the number of attributes displayed by the attribute table” [65] below . See also the `attributeVisility` configuration element in *XMLmind XML Editor - Configuration and Deployment*.

*Figure 7.4. Menu when a DITA topic is opened in **XXE***



This menu allows to toggle in the attribute table the visibility of attributes belonging to certain categories. Simply uncheck a menu entry to “hide” in the attribute table all the attributes belonging to the corresponding category.

- Once an attribute has been specified for an element, it is displayed in the attribute table in all cases, no matter whether it belongs to a category marked as hidden.
- Even when an available attribute has been made hidden, it's still possible to specify it using the attribute form [67] (auto-completion fully supported there) found above the attribute table.

Configuration Families	Attribute Categories (*Hidden* by default unless stated otherwise)
XHTML	**Scripting** `onclick`, `onload`, `onmouseenter`, etc.
XHTML5	**ARIA** `role`, `aria-*`. **Browser Behavior** `autocapitalize`, `contenteditable`, `draggable`, etc. **Microdata** `itemprop`, `itemscope`, etc. **Scripting** `onclick`, `onload`, `onmouseenter`, etc. **XML** `xml:base`, `xml:lang`, etc. **XSI** `xsi:type`, etc.
DITA	**Conditional Processing** `audience`, `platform`, etc. *Visible by default.*

Configuration Families	Attribute Categories (*Hidden* by default unless stated otherwise)
	Other Metadata `importance`, `base`. **Localization** `translate`, `dir`. **Debug** `xtrc`, `xtrf`.
DocBook v5+	**Effectivity** `arch`, `condition`, etc. *Visible by default*. **Other Common** `remap`, `xml:base`, etc. **Linking** `linkend` and `xlink:href`[a]; `xlink:actuate`, `xlink:arcrole`, `xlink:from`, etc. **Scripting** `onclick`, `ondblclick`, etc. (Found on an HTML `table`.)

[a]Always visible when parent element is `link`, `xref`, etc

3.2. Using the attribute form

Procedure for adding an attribute or changing its value:

1. Type the name of the attribute in the name field (first field of the form).
2. Press **Enter** or **Tab** to move to the value field (second field of the form).
3. Type the value of the attribute in the value field.
4. Press **Enter** in the value field to commit the change. This also gives the keyboard focus back to the document view.

Both the name and value fields support auto-completion. However auto-completion in the value field only works for attributes having certain types, for example, any enumerated type, `ID`, `IDREF`, `IDREFS`.

This auto-completion feature can be configured using the **Preferences** dialog box [113].

Procedure for removing an attribute:

1. Select the attribute by either clicking on it in the attribute table or by typing its name in the name field (first field of the form).
2. Press **Enter** to move to the value field (second field of the form).
3. Click on the right part — the down arrow — of the **Edit** button.

4. Click the **Remove** item in the displayed menu.

Procedure for adding an attribute to an element of the unconstrained type:

1. Type any attribute name in the name field (first field of the form).

If the namespace of the attribute name has not been declared, you will have to declare it using **Tools** → **Declare Namespace** [38] before being allowed to add the attribute.

2. Type the value of the attribute in the value field (second field of the form).
3. Press **Enter** in the value field to commit the change. This also gives the keyboard focus back to the document view.

Procedure for adding an attribute that matches an attribute wildcard:

1. Type an attribute name that matches the wildcard in the name field (first field of the form).

If the namespace of the attribute name has not been declared, you will have to declare it using **Tools** → **Declare Namespace** [38] before being allowed to add the attribute.

2. Type the value of the attribute in the value field (second field of the form).
3. Press Enter in the value field to commit the change. This also gives the keyboard focus back to the document view.

   XXE will of course check that the name you have typed conforms to the attribute wildcard.

The attribute form contains the following buttons:

Add or change attribute

Commits the changes. Analogous to pressing **Enter** in the value field.

Cancel

Cancels the non-committed changes and clears the attribute form. Analogous to typing **Esc** in the name or value field.

Edit

The right part — the down arrow — of this button displays a menu, similar to the popup menu which is displayed when you right-click on a row of the attributes table.

The left part of this button displays a dialog box which can be used to easily specify a value for the attribute being edited. The value specified using this dialog box is directly assigned to the attribute.

Specialized helper dialog boxes are displayed for attributes having certain types, among these types:

- any enumerated type,
- `ID`, `IDREF`, `IDREFS` types,
- `hexBinary` and `base64Binary` types.

The specialized dialog box which is displayed when the type of the attribute being edited is an enumerated type or a reference type of any kind (`ID`, `IDREF`, `href`, etc) is documented in Section 4, “The "**Attribute Value**" dialog box” [103].

For all other types of attributes, a very simple multi-line editor is displayed. This may be useful to view or edit attribute values which are too long to be comfortably viewed or edited using the value field.

Note that in the case of the dialog box displayed for enumerated types and ID or IDREF types, a *single click* is sufficient to select a value from the displayed list. Also note that the text field above the list supports *auto-completion*.

Browse Files

Displays a file chooser dialog box. The URL or filename specified using this dialog box is is directly assigned to the attribute.

Clicking on the right part — the down arrow — of this button displays a menu with 8 radio buttons allowing to choose between:

- Open or save files.
- File or directories.
- Absolute or relative paths (relative paths are relative to the location of the element being edited).
- URL or plain file names.

The choices made using these radio buttons are persistent during the editing session (but not across editing sessions like, for example, user preferences specified using the **Preferences** dialog box [106]).

4. Search tool

The text selection, if any, is automatically used to initialize the **Search** field of the **Search** tool [69]. As a consequence, in some cases, you don't need to use the **Search** tool at all. For example, selecting some text in the document and then directly pressing **Ctrl**+**G** (**Search** → **Find Next**) is the quickest way to find the next occurrence of the selected text in the document.

Procedure for performing a text search:

1. Type text to be searched in **Search** field or choose it from the menu of the combobox.
2. Change the search options as needed (see below).
3. Type Enter in the **Search** field or click on the **Start** button.
4. During the search session
   - click **Skip** on to skip found text,
   - OR click on **Skip Element** to skip the element containing the found text.
5. The search session can be stopped by
   - clicking on **Stop**,
   - OR typing **Esc** (this also gives back the keyboard focus to the document view),
   - OR by simply clicking anywhere in the document view.

Procedure for performing a text search/replace:

1. Type text to be searched in **Search** field or choose it from combo box menu.
2. Check the **Replace** toggle
3. Type replacement text in **Replace** field or choose it from combo box menu.
4. Change the search options as needed (see below).
5. Type Enter in the **Search** or **Replace** fields or click on the **Start** button.
6. During the search/replace session
   - click **Skip** on to skip found text,
   - OR click on **Skip Element** to skip the element containing the found text.
   - OR click **Replace** on to replace found text.
   - OR click **Replace All** on to replace all occurrences of searched text.
7. The search/replace session can be stopped by
   - clicking on **Stop**,
   - OR typing **Esc** (this also gives the keyboard focus back to the document view),
   - OR by simply clicking anywhere in the document view.

Search options:

Ignore case

The search is case-insensitive. Example: "`foo`" matches both "`foo`" and "`Foo`".

When "**Ignore case**" has been checked, "**Regular expression**" has not been checked and both the searched string and the replacement string contain only lower-case letters or the hyphen character ('`-`'), the replacement string actually used is capitalized exactly like the found string it will replace.

Example: if you specify "`editor`" as the search string and "`modeller`" as the replacement string, the **Search** tool will replace "`editor`" by "`modeller`", "`Editor`" by "`Modeller`" and "`EDITOR`" by "`MODELLER`".

Counterexample: if you specify "`IBM`" as the search string and "`Lenovo`" as the replacement string, the **Search** tool will replace "`IBM`", "`Ibm`" and "`ibm`" by "`Lenovo`".

Whole word

The found string must be a word, that is, the found string must be surrounded by white spaces. Example: "`foo`" matches "`foo`" but not "`foobar`".

Regular expression

The searched string must be a valid regular expression. A regular expression is specified in a syntax similar to that used by Perl.

In such case, `$1`, `$2`, ..., `$9` may be used in the replacement string to refer to the substrings matching the parenthesized groups of the regular expression.

`$0` is replaced by the string matching the regular expression in its entirety. "`$$`" may be used to quote character '`$`'.

Examples:

- "`f(o+)`" matching "foo", replaced by "`g$1`", gives "`goo`".
- "`f(o+)`" matching "`foo`", replaced by "`$0bar`", gives "`foobar`".
- "`f(o+)`" matching "`foo`", replaced by "`g$$1`", gives "`g$1`".

Smart text boundaries

Searching string "`Hello world!`" in an XML document is not as obvious as it seems: for example, is "`Hello world!`" with word "`Hello`" contained in an emphasis element followed by text node "` world!`" supposed to be found by **XXE**?

- If this toggle is checked, the answer is yes. "`Hello world!`" is found within "`<em>Hello</em> world!`" but not within "`<p>Hello </p><p>world!</p>`".

  This mode uses the schema used to validate current document to recognize logically contiguous text across different types of elements.

- If this toggle is not checked, the answer is no. Each text node is separated from other text nodes whatever the type of the element containing it.

This option is turned off by default, as it is rarely needed and also because it may cause the **Search** tool to fail in finding some strings for certain document types.

Direction

Up

Search from the current caret position to the beginning of the document.

Down

Search from the current caret position to the end of the document.

Highlight all

During the text search and/or replace operation (that is, temporarily), highlight all text spans matching the search criteria.

Wrap search

Restart the search from the beginning of the document when the end of the document has been reached. (If **Direction** is **Down**, restart the search from the end of the document when the beginning of the document has been reached.)

5. Validity tool

This tools displays the list of validity error messages (if any) found by **XXE** when opening a document or after the last use of

- **Tools** → **Check Validity** [38] or Validity State [88],
- OR **File** → **Save** [16] or **File** → **Save As** [16] (validity is automatically checked each time a document is saved).

The color of the message reflects the severity of the error. See icons used by Validity State [88].

Clicking on the number of an error message selects the element where the validity error was found. If the document is edited after its validity is checked (typically to fix some of the validity errors), clicking on some of the error messages may have no effect because the corresponding erroneous element no longer exists.

When the document has severe validity errors (Invalid Data, Invalid Structure), less severe errors (Semantic Error, Semantic Warning, Invalid Reference) are generally *not displayed* by the **Validity** tool.

6. The "Browse Files" tool

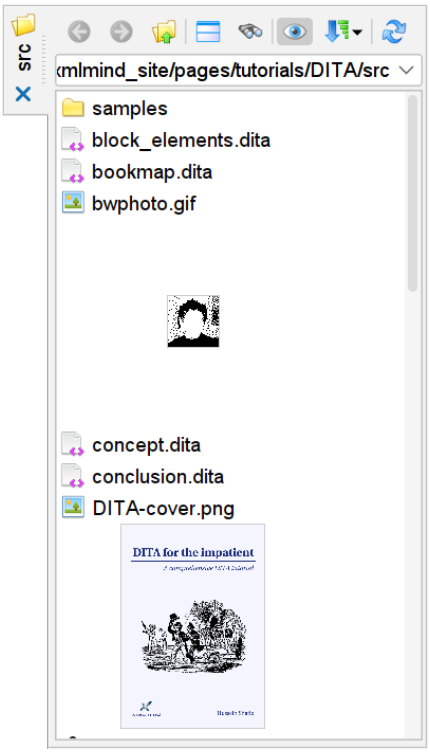
This tool is displayed after you select any of the menu items of the **File** → **Folder** [17] submenu.

You may need to enable the **File** → **Folder** submenu by checking "**Enable the 'File|Folder' Submenu**" in **Options** → **Preferences**, **General**|**Features** section.

This tool is similar to the file manager applications which are native to all operating systems: Windows Explorer, Mac OS X Finder, Linux/KDE Nautilus, etc. The main advantages of using one or more "**Browse Files**" tools over the native counterparts are:

- Tightly integrated in XMLmind XML Editor.
- Does not use much screen real estate.
- Makes use of all XXE plug-ins. For example, if you install the "*JEuclid image toolkit plug-in*" add-on, you'll be able to preview MathML equations.
- Last but not least, if you have the "*WebDAV virtual drive plug-in*" add-on and/or the "*FTP virtual drive plug-in*" add-on installed, you'll be able to manage files *stored on all sorts of remote servers*. Installing such add-ons is done simply by using **Options** → **Install Add-ons** [46].

*Figure 7.5. The "**Browse Files**" tool showing the source files of our DITA tutorial*



Notice in the above screenshot that the "**Show image thumbnails**" option (see below [73]) is turned on and that the icon next to "`concept.dita`" has a small lock, meaning that is file is currently locked[3].

Toolbar buttons:

Back

Browse previously visited folder.

Forward

Browse previously visited folder by undoing the **Back** action [73].

Go up one level

Browse parent folder.

Split

Split the "**Browse Files**" tool in two parts, making it easy to drag/drop or to copy/paste files between these tools.

Search

Open a "**Search Files**" tool [78] allowing to search by their contents the XML files found in this folder.

Show image thumbnails

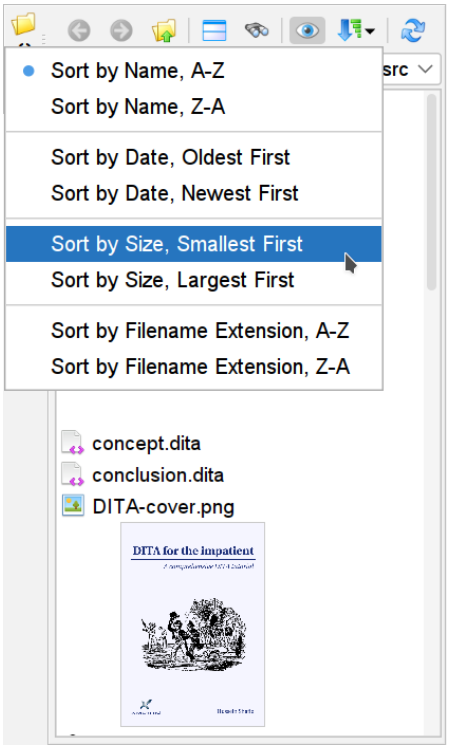
This checkbox adds a thumbnail after each list item corresponding an image file.

[3]Details about the lock are given by the tooltip of the file icon.

This facility makes use of all the *image toolkit plug-ins* currently installed in **XXE**. If you are missing an image format (e.g. SVG), please install the corresponding image toolkit plug-in (e.g. "*Apache Batik image toolkit plug-in*") using **Options** → **Install Add-ons** [46].

Sort files by...

Displays a menu letting you sort the listed files according to different file properties: name, date, size, etc, in either ascending or descending order. Note that only files are sorted using selected choice; folders are always listed before any file and are always sorted by their name.



Just like "**Show image thumbnails**" [73], the choice made using this menu is *not* a user preferences. (The choice made by the user will not “survive” editing sessions.) A newly created "**Browse Files**" tool uses default "**Sort by Name, A-Z**". However a "**Browse Files**" tool created by clicking **Split** [73] uses the same sort option as its originating "**Browse Files**" tool.

Refresh

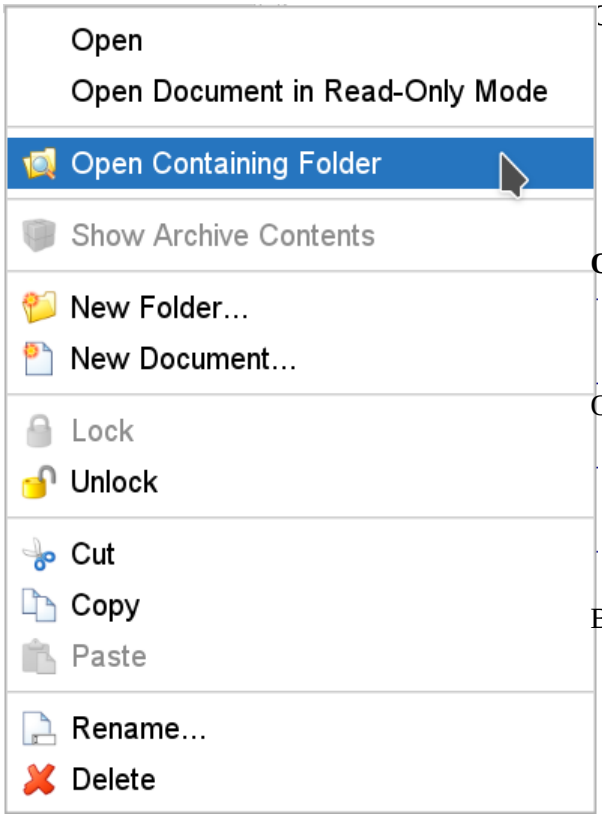
Update what is displayed by the "**Browse Files**" tool. Clicking this button is needed when the content of the folder is modified outside XMLmind XML Editor.

Below the toolbar is found an interactive pane containing a visual representation of the folder:

- Up, Down, Shift+Up, Shift+Down may used to select one or more files. These shortcuts work as expected in any multi-selection list.
- Press Enter to open the selected file. If the selected file is an XML file, it is opened in **XXE**. If the selected file is not an XML file, a dialog box [43] is displayed allowing the user to choose a helper application.

  Press Shift+Enter to open selected XML file in read-only mode.

- Click, Shift+Click, Ctrl+Click may used to select one or more files. These actions work as expected in any multi-selection list.
- Double-click on a file to open it. If the selected file is an XML file, it is opened in XXE. If the selected file is not an XML file, a dialog box [43] is displayed allowing the user to choose a helper application.
- Right-click on a file to first select it and then to display the following popup menu:



Open

Open the selected file. If the selected file is an XML file, it is opened in **XXE**. If the selected file is not an XML file, a dialog box [43] is displayed allowing the user to choose a helper application.

Open Document in Read-only Mode

Same as above, but the document is opened in read-only mode.

Open Containing Folder

Open selected file or directory in the standard file manager of your operating system. For example, on the Mac, this will open selected file in the Finder.

Browse Archive Contents

Open a new "**Browse Files**" tool showing the contents of selected Zip archive. Enabled only if the selected file is a Zip archive of any known kind (`.zip`, `.jar`, `.epub`, `.docx`, `.xlsx`, `.odt`, etc).

Note that the "**Browse Files**" does not allow to modify the contents of a Zip archive. For example, double-clicking on a XML file found in a Zip archive automatically opens this file in **XXE** in read-only mode.

New Folder

Create a subfolder in current folder.

New Document

Create an XML file using the same dialog box as the one displayed by **File** → **New** [15].

Lock

Lock selected file.

If the current folder is stored on a WebDAV server, it is also possible to lock a subfolder of the current folder.

Unlock

Unlock selected file.

In principle, the file must have been locked by you. However, this menu item will attempt to unlock it in all cases and depending on your privileges, this action may succeed or fail.

Cut

Cut selected files.

Copy

Copy selected files.

Paste

Paste files cut or copied using the above menu items.

Rename

Rename selected file.

Delete

Delete selected files.

Revisions

This menu item is present only when the underlying storage facility supports file versioning. More information in Section 6.1, “Opening and comparing revisions of a document” [76].

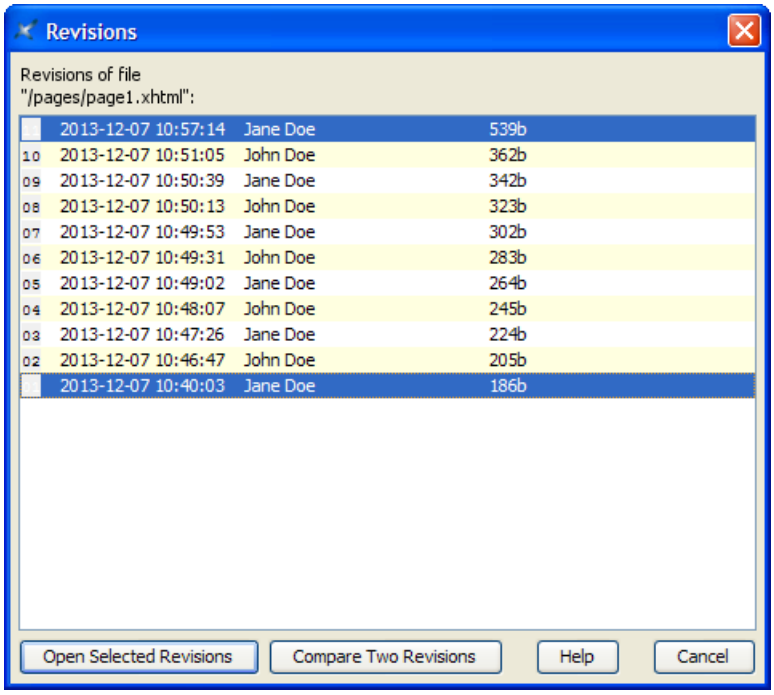
- The customary keyboard shortcuts **Ctrl**+**A**, **Ctrl**+**X**, **Ctrl**+**C**, **Ctrl**+**V**, **Del**, Backspace work as expected in any file manager.
- The "**Browse Files**" tool is fully drag and drop enabled. It supports the copy and move operations. It does not support the link operation.

  Note that when an XML file is dragged and dropped onto **XXE**, the document is always opened *or reopened* in **XXE**.

6.1. Opening and comparing revisions of a document

When the underlying storage facility supports file versioning (e.g. Google Drive™), the popup menu of the "**Browse Files**" tool has a **Revisions** entry. This entry displays the following dialog box:

Figure 7.6. The "List Revisions" dialog box



This dialog box lists all the revisions of the document being selected in the "**Browse Files**" tool. Revisions are ordered from most recent (top of the list; bigger revision numbers) to least recent (bottom of the list; smaller revision numbers). An item of the list contains the following information: the number of the revision, its creation date, the name of the user who created it, its file size.

The "**Open Selected Revisions**" and "**Compare Two Revisions**" buttons will always be disabled ("grayed") when selected file cannot be opened in **XXE**. This happens for example for files having a `.html` extension which are considered to contain plain HTML (`text/html`) rather than XHTML (`application/xhtml+xml`).

When you know that files having a given extension actually contain XML, you may instruct **XXE** to open such files by using **Options** → **Preferences**, **Tools**|**Helper Applications** [121]. As expected, the settings will also have an effect on the "**Open Selected Revisions**" and "**Compare Two Revisions**" buttons.

Open Selected Revisions

May be used to open in **XXE** one or more revisions selected from the list. The most recent revision is opened normally, in read-write mode. Older revisions are opened in read-only mode.

Using **File** → **Save As** on a document opened in read-only mode will cause it to switch to the normal read-write mode.

This precisely how you can revert to an old revision of a document. Example: saving "`/pages/page1.xhtml 1`" (which identifies revision #1 of file "`/pages/page1.xhtml`") as "`/pages/page1.xhtml`" will cause this document to revert to its initial revision.

Compare Two Revisions

May be used to directly compare using the **Compare** tool [84] two revisions selected from the list. However note that

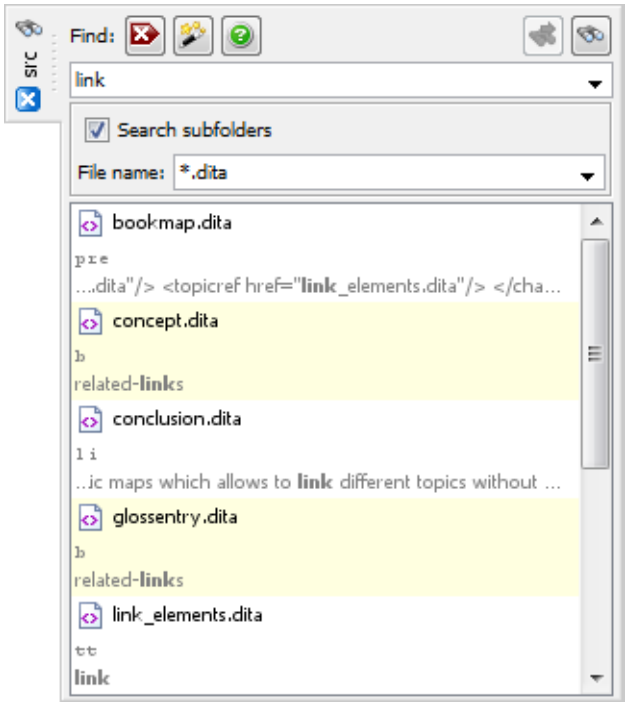
- this button is disabled when the "**Tools** → **Revisions**" submenu feature has been disabled using **Options** → **Preferences**, **General**|**Features** section;
- this button will display an alert dialog box when the comparison of revisions has not been enabled for the document being selected in the "**Browse Files**" tool.

7. The "Search Files" tool

This tool is displayed after you click the **Search** [73] button in the "**Browse Files**" tool. The "**Browse Files**" tool is itself displayed after you select any of the menu items of the **File** → **Folder** [17] submenu.

You may need to enable the **File** → **Folder** submenu by checking "**Enable the 'File|Folder' Submenu**" in **Options** → **Preferences**, **General**|**Features** section.

Figure 7.7. Search our DITA tutorial for topic files (`*.dita`) containing substring `"link"`



Toolbar buttons

Clear

Clear the query text field (see below), the "**File name:**" text field and the "search results" pane.

Create or edit an advanced query

Display a query editor dialog box allowing to create an advanced query or to edit the current query (that is, the one specified in the query text field).

Help

Displays an online help for "**Search Files**" tool.

Search next

Displays up to 20 more results for the current search.

Search

Start searching or restart current search . Display at most the first 20 results.

When a search is being performed, the **Search** and **Search next** buttons are given a special **Stop** icon . Clicking this icon allows to stop the search.

The query text field

Below the toolbar is found a text field where you can type a query. It's possible to type simple or advanced queries in this text field.

Advanced queries are documented in Section 7.1, “The "**Create or edit an advanced query**" dialog box” [80].

A simple query such as "`Hello world!`" (without the quotes) means: find all XML files containing an *element* whose *direct textual content* contains *substring* "`hello world!`" (case-insensitive).

This is very different from web search engines such as Google™ where query "`Hello world!`" (without the quotes) means: find all pages containing *word* "`hello`" *and word* "`world`" (case-insensitive).

Note that this simple text search facility is always case-insensitive. Also note that the whitespace in the textual content of elements, attributes, comments and processing-instruction is always normalized while traversing the content of an XML file. That is, the possible indentation of the XML file will not affect the results of the search.

The query text field may be left empty, in which case the search will be performed on the basenames of the files. See below.

The "search files by name" form

The "**Search subfolders**" checkbox allows to perform the search, not only in the current folder, but also recursively in all its subfolders.

Below the "**Search subfolders**" checkbox is found a "**File name:**" text field where you can type a *glob pattern*. This glob pattern is used to match the basename of the files being searched.

When this field is left empty, XXE will search all the files known to be XML (the union of `*.xml`, `*.dita`, `*.ditamap`, `*.dbk`, `*.xsl`, `*.xsd`, `*.fo`, etc).

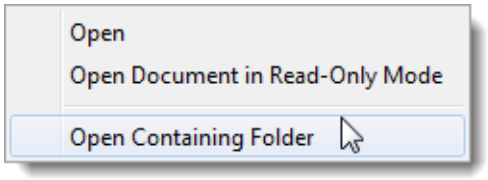
Note that, by default, `.htm`, `.html` and `.shtml` files are *not* considered to be XML files. Only `.xhtml` files are considered to be XML files. Therefore, if you always create XHTML files and nevertheless give your files extensions such as `.htm`, `.html` or `.shtml`, do not forget to specify "`*.*htm*`" in the "**File name:**" text field.

Note that the query text field may be left empty, in which case the search will be performed on the basenames of the files. For example, specify "`*.{svg,png}`" in the "**File name:**" field and leave blank the query field in order to list all `SVG` and all `PNG` files.

The "search results" pane

Below the above toolbar and text fields, is found an interactive pane containing the results of a search:

- Press Up to select the preceding result. Press Down to select the following result.
- Press Enter to open in XXE the document corresponding to selected result.
- Click on a result to select it.
- Double-click on result to open in XXE the document corresponding to this result.
- Right-click on a result to first select it and then to display the following popup menu:



Open

Open in XXE the document corresponding to selected result.

Open Document in Read-only Mode

Same as above, but the document is opened in read-only mode.

Open Containing Folder

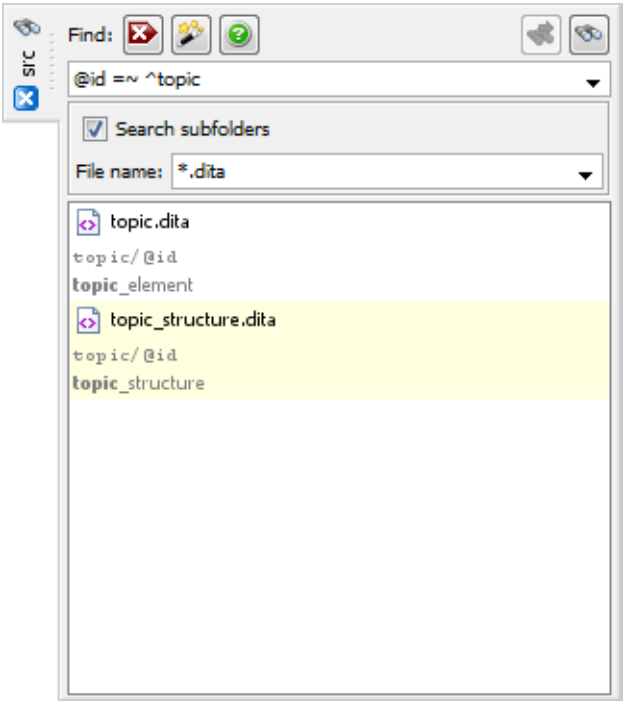
Open the folder containing the document corresponding to selected result. This menu item displays the "**Browse Files**" tool [72].

- Dragging the selected result will drag the corresponding document inside or outside XXE.

  Note that when a document is dragged and dropped onto XXE, this document is always opened *or reopened* in XXE. This is not the case of all the "open document" facilities described above. Such facilities have been designed for *navigation* and thus, will not reopen a document if it's already opened in XXE. Instead, the opened document will automatically scroll to display the element of interest.

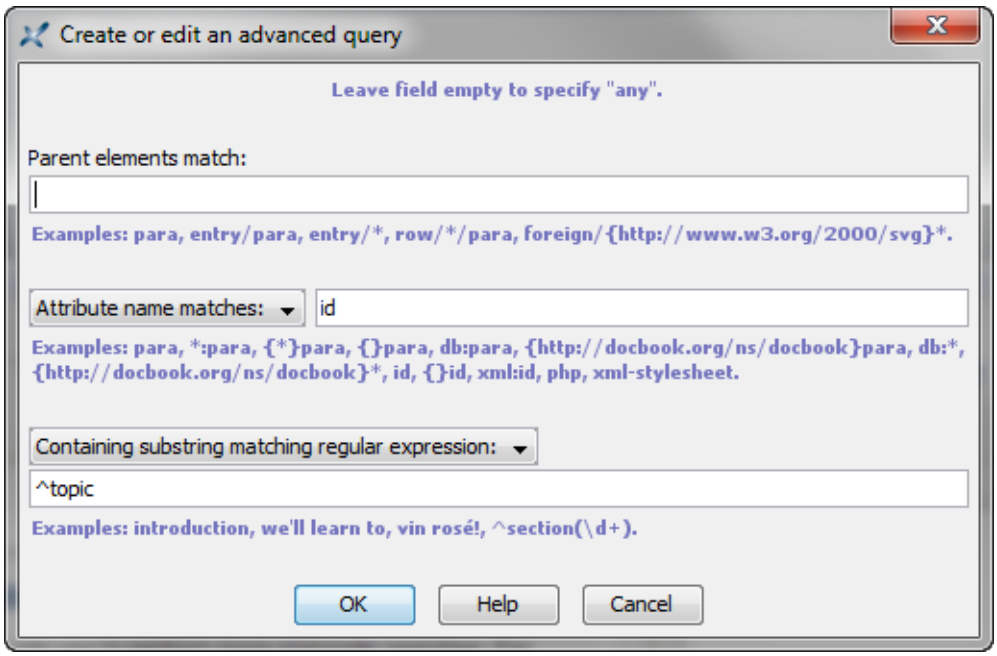
7.1. The "Create or edit an advanced query" dialog box

The "**Search Files**" tool is not limited to simple text searches. You can use it to perform more elaborate searches. For example: find all documents containing an element whose `id` attribute starts with "`topic`":



In the above screenshot, notice the query being used: "`@id =~ ^topic`". The syntax of this query [83] is easy to remember. However, fortunately, you don't even need to do this effort. Suffice to click on "**Create or edit an advanced query**" to display a query editor dialog box.

*Figure 7.8. The "**Create or edit an advanced query**" dialog box*



Description of the text fields and comboboxes from top to bottom and left to right:

Text field "**Parent elements match:**"

Specifies the parent element and also possibly the ancestors of the subject of the search.

An empty "**Parent elements match:**" text field means: whatever the parent or ancestors.

Subject type combobox

Specifies the node type of the subject of the search: element, attribute, processing-instruction or comment.

Subject name text field

Specifies the name of the subject of the search when this subject is an element or attribute. Specifies the target of a processing-instruction when the subject of the search is a processing-instruction.

- A non-qualified name such as "`para`" matches any name having "`para`" as its local part . That is, the namespace and the prefix of a name are not considered.
- A qualified name such as "`db:para`" matches any name having "`db`" as its prefix and "`para`" as its local part. The namespace of a name is not considered.
- A name specified using Clark's notation such as "`{http://docbook.org/ns/docbook}para`" matches any name have "`http://docbook.org/ns/docbook`" as its namespace and "para" as its local part. The prefix of a name is not considered.

  Use "`{}`" to specify the absence of namespace. Example: "`{}para`".
- In a name specification, an "`*`" wildcard may be used in the local part, the prefix or the namespace. Examples: "`db:*`", "`*:para`", "`{*}para`".

An empty subject name text field means: whatever the name of the element or attribute or whatever the target of the processing-instruction.

Match operation combobox

Specifies how the textual content of the subject is matched with the searched value:
- the textual content contains the searched value,
- OR the textual content is equal to the searched value,
- OR the textual content contains a substring matching the searched value, which is a regular expression.

All these operations are case-insensitive.

Searched value

Specifies the searched value, some plain text or a regular expression, depending on the chosen the match operation. The text search is case-insensitive.

The supported syntax for regular expressions is documented here.

An empty searched value text field means: whatever the textual content of the subject.

Examples:

- Find documents containing `literal` elements containing a text equal to "`XXE`": **Element name matches:** `literal`; Searched value: `XXE`.
- Find documents containing `table` elements having an `xml:id` attribute: **Parent elements match:** `table`; **Attribute name matches:** `xml:id`.
- Find documents containing comments: simply select "**Comment**" in the Subject type combobox.

- Find documents containing the <?xxe-relaxng-schema> processing-instruction: **PI target equals:** xxe-relaxng-schema.
- Find documents containing absolute HTML links: **Parent elements match:** a; **Attribute name matches:** href; **Containing substring matching regular expression:** ^http://.

7.2. Syntax of search queries

```
query -> [ subject ]? [ op ]? [ value ]?

subject -> [ ancestors ]? simple_subject

ancestors -> [ name '/']+

simple_subject -> element | attribute |
                    processing-instruction | comment

element -> name

attribute -> '@'name

processing-instruction -> 'processing-instruction(' target ')'

comment -> 'comment()'

name -> local_name |
         prefix ':' local_name |
         '{' [ namespace ]? '}' local_name

local_name -> NCName | '*'

prefix -> NCName | '*'

namespace -> anyURI | '*'

target -> NCName | '*'

op -> '=*' | '=' | '=~'

value -> plain_text_or_regular_expression
```

The default *subject* is: any element.

The default *op* is: =*.

The default value is: any value.

Op	Description
=*	The textual content of the subject contains the searched value.
=	The textual content of the subject is equal to the searched value.

Op	Description
=~	The textual content of the subject contains a substring matching the searched value, which is a regular expression. The supported syntax for regular expressions is documented here.

Examples:

- Find documents containing elements containing substring "XXE": `=* XXE` or more simply `XXE`
- Find documents containing attributes containing substring "XXE": `@* =* XXE`
- Find documents containing `literal` elements containing a text equal to "XXE": `literal = XXE`
- Find documents containing `table` elements having an `xml:id` attribute: `table/@xml:id =`
- Find documents containing comments: `comment() =`
- Find documents containing the `<?xxe-relaxng-schema>` processing-instruction: `processing-instruction(xxe-relaxng-schema) =`
- Find documents containing absolute HTML links: `a/@href =~ ^http://`

8. The Compare tool

You may need to enable this tool by checking "**Enable the 'Tools|Revisions' Submenu**" in **Options** → **Preferences**, **General**|**Features** section.

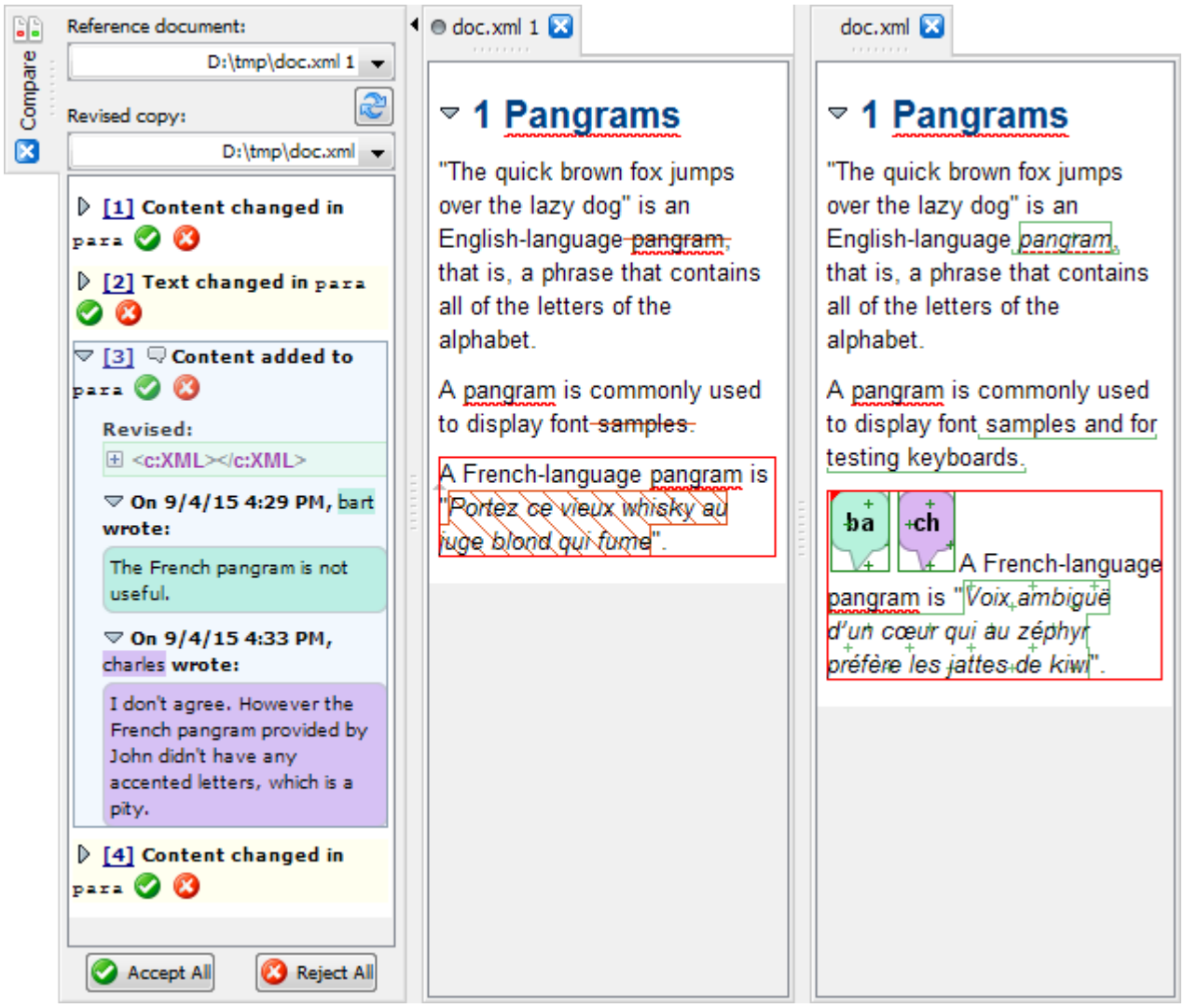
8.1. Why use the Compare tool?

The **Compare** tool allows to compare two revisions of the same initial document.

Unlike generic XML comparison tools such as DeltaXML or Altova DiffDog, you must explicitly enable the comparison of revisions in the initial document if you want to be able to compare two revisions using the **Compare** tool. This is done by selecting menu item **Tools** → **Revisions** → **Enable the Comparison of Revisions** [41] (or **Tools** → **Revisions** → **Store All Revisions in the Document** [41]).

After selecting menu item **Tools** → **Revisions** → **Compare Revisions** [42] in order to display it, the **Compare** tool will show you the differences existing between two revisions. It also allows to accept or reject some or all the changes.

*Figure 7.9. The **Compare** tool showing the differences existing between `doc.xml 1` (initial revision) and `doc.xml` (latest revision), displayed side by side*

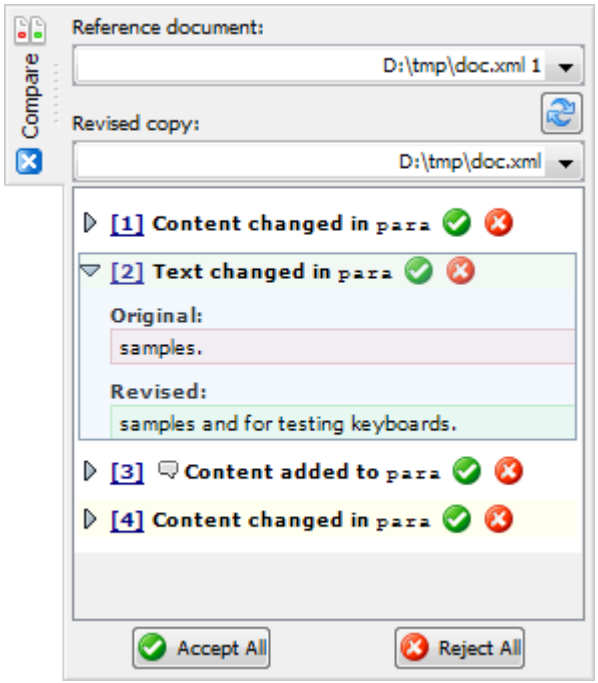


8.2. Using the Compare tool

The **Compare** tool is displayed at the left of the main window after selecting menu item **Tools** → **Revisions** → **Compare Revisions** [42].

In order to compare two documents using the **Compare** tool, you must select these documents using the comboboxes found at the top of the tool. These comboboxes list all the documents opened in XMLmind XML Editor. However you can only choose two revisions of the same initial document (see Section 8.1, “Why use the **Compare** tool?” [84]).

*Figure 7.10. The **Compare** tool showing change #2*



In the above screenshot, notice the icon which indicates that a change contains one or more remarks [39].

Below the comboboxes is found an interactive pane containing the list of changes made in the revised copy to the reference document:

- Click on a change to select it.

  Doing this gives the keyboard focus to the interactive pane and thus allows to use keyboard shortcuts (see below) rather than the mouse.

- Click on the number of a change or double-click anywhere in a change first to explicitly select the changed elements in the document views and second to center the changed contents in these views. Doing this allows to clearly see what happened.

  It is strongly recommended to display the view of the reference document and the view of the revised copy side by side. This is done by:
  1. Turning on option **Window** → **Split Windows Horizontally** [48].
  2. Right-clicking in the tab of the revised copy and selecting "**Move Tab to the Opposite Side**" [50] from the popup menu.

- Click on the triangle icon to expand/collapse the corresponding change.

- Click on to accept the selected change. However, if there are changes which are similar to the selected change, the button displays a popup menu letting the user choose between "**Accept this change**" and "**Accept this change and *N* similar changes**". The second menu item is especially handy if for example, you have used "**Replace All**" in the revised document.

Click the **Accept All** button to accept all the remaining changes.

Accepting a change means approving the contents found in the revised copy.

Once a change has been accepted, it is removed from the list and the corresponding marks are removed from the document views.

- Click on to reject the selected change. However, if there are changes which are similar to the selected change, the button displays a popup menu letting the user choose between "**Reject this change**" and "**Reject this change and *N* similar changes**". The second menu item is especially handy if for example, you have used "**Replace All**" in the revised document.

  Click the **Reject All** button to reject all the remaining changes.

  Rejecting a change means restoring in the revised copy the original contents found in the reference document.

  Once a change has been rejected, it is removed from the list and the corresponding marks are removed from the document views.

- Press Up to select the preceding change. Press Down to select the following change.
- Press Enter first to explicitly select the changed elements in the document views and second to center the changed contents in these views. Doing this allows to clearly see what happened.
- Press Space to expand/collapse the selected change.
- Press **F3** to accept the selected change. Press **F5** to accept all the remaining changes.
- Press **F4** to reject the selected change. Press **F6** to reject all the remaining changes.
- If you modify the revised copy, it's possible to quickly update the list of changes by clicking **Update changes**.

Chapter 8. Status bar

1. Validity state

Clicking on this button displays **Validity** tool [71], unless no validity errors are found in current document, in which case an OK message is displayed in the status bar.

This button is disabled if current document is not constrained by a grammar.

Current document validity is automatically checked each time the document is saved, therefore, unless you are fixing an invalid document, you don't really need to explicitly use this button.

The icon contained in this button shows the validity state of current document (from least severe error to most severe error):

Icon	Name	Meaning
	N/A	The document is valid and has no semantic errors.
	Invalid Reference	The document has minor validity errors such as bad cross-references. Working on a document which has bad cross-references is very common so you should not be alarmed by this situation.
	Semantic Warning	The document is valid but has semantic warnings. This happens when the document being edited is itself a W3C XML Schema or a RELAX NG schema or when a user-defined `validateHook` reports such warnings (See Section 34, “validateHook” in *XMLmind XML Editor - Configuration and Deployment*).
	Semantic Error	The document is valid but has semantic errors. This happens when the document being edited is itself a W3C XML Schema or a RELAX NG schema or when a user-defined `validateHook` reports such errors (See Section 34, “validateHook” in *XMLmind XML Editor - Configuration and Deployment*).
	Invalid Data	The document has validity errors signaling bad values for attributes or for elements.
	Invalid Structure	This document has severe validity errors such an invalid child sequence inside an element. *It is not a good idea to keep working on a document with such severe validity errors because **XXE** has not been designed to be convenient to use in such situation.* After fixing some structure errors using **XXE**, save and then reload the document. This will automatically discard superfluous white space which was not detected at first because of the validity errors. This

Icon	Name	Meaning
		trick is much faster that trying to remove invalid white space text nodes by hand.

2. Document cache indicator

The document cache indicator is hidden by default. You need to enable it by checking "**Enable the Document Cache Indicator**" in **Options** → **Preferences**, **General**|**Features** section.

Document cache

As of v4.9, XXE makes a heavy use of its document cache. This indicator allows to have a feedback while the document cache is caching documents. Its tool tip shows the number of documents currently cached as well as the capacity of the cache. The number of cached documents is also rendered visually.

This indicator is not a button. You'll have to use **Options** → **Preferences**, **Advanced**, "**Cached data**" section to configure or clear the document cache.

3. Status messages

Show Message Log

Displays a dialog box containing last messages reported by XXE.

These messages are sorted by category. One of the most useful category is "**Command Execution**" which contains the messages reported during the execution of the last command. Commands such as **DocBook** → **Convert Document** → **Convert to PDF** are really verbose, that's why this category is so useful.

It is also possible to display the same dialog box by right-clicking in the message area of the status bar.

4. Spell checker

Automatic Spell Checker

Toggle used to activate and deactivate the automatic (AKA on-the-fly) spell checker. See also option "Activate if this is specified in the configuration file [120]".

Check Spelling

Displays the "**Check Spelling**" dialog box [103].

5. Overwrite mode

INS/OVR

INS indicates *Insert Mode*: typing a character inserts it at caret position.

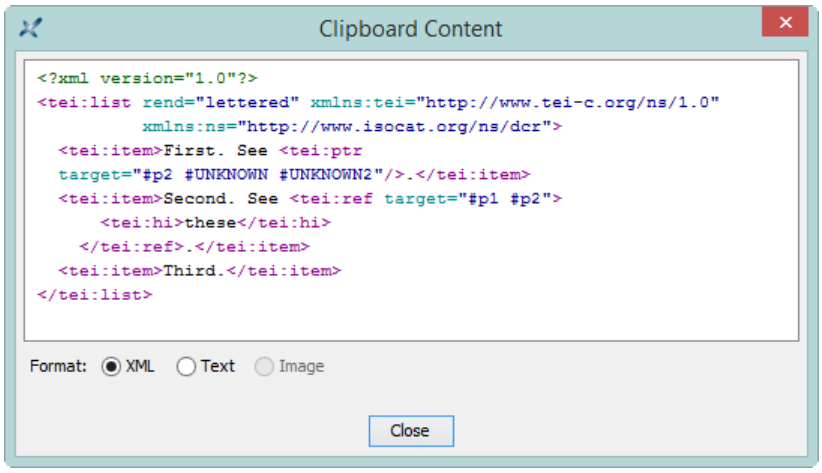
OVR indicates *Overwrite Mode*: typing a character *replaces* the character found at caret position by the typed character. If the caret is positioned at the very end of a text (or comment or processing-instruction) node, then typed characters are simply inserted there.

Pressing the **INS/OVR** button allows to switch from Insert Mode to Overwrite Mode and vice versa. The shortcut associated to this button is by default **Esc I** on the Mac and **Esc Insert** (press **Esc**, release **Esc**, then press **Insert**) on the other platforms.

6. Clipboard utilities

Clicking on this button (or right-clicking in the text field next to it) displays a dialog box showing the contents of system clipboard.

Figure 8.1. The clipboard viewer dialog box



The clipboard viewer indicates the format of the data found in the system clipboard: **XML**, **Text** or **Image**[1].

It's possible to switch from **XML** to **Text** and, *if the text is a well-formed XML fragment*, from **Text** to **XML**, the latter being useful when you copy XML fragments from a third party application.

6.1. Clipboard content

Next to the above button, a read-only text field permanently shows a short description of current clipboard content:

- an element name if a single element has been copied or cut to the clipboard,
- OR `#text` for a text node or for a sequence of characters,
- OR `#comment` for a comment node.
- OR `#processing-instruction` for a processing instruction node,
- OR the number of nodes (between square brackets) if more than one node have been copied or cut to the clipboard . Example "`[5]`" if the five list items of this itemized list have been copied to the clipboard.

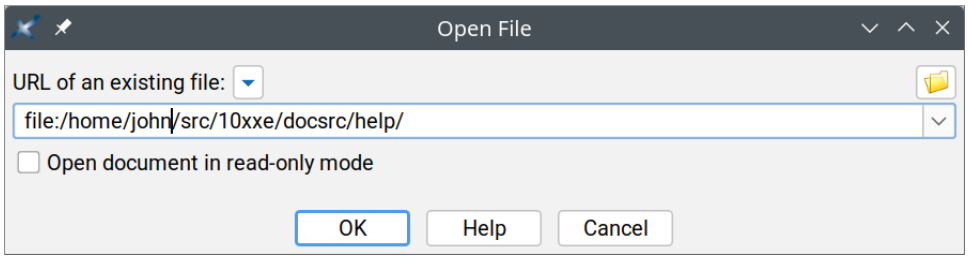
[1] If both the plain text and image formats are available (e.g. MS-Word does this), the clipboard viewer indicates just **Text**.

Chapter 9. Dialog boxes

1. The URL chooser dialog box

The URL chooser dialog box allows to specify the location of a file or directory on the local file system and, if *virtual drive plug-ins have been installed*[1], on remote (e.g. HTTP, FTP) file systems. This file or directory may be an existing one or a file or directory to be created; it depends on the command displaying this dialog box.

This dialog box is similar to the "**Open Web Location**" dialog box found in many Web browsers. To use it, simply type in its text field the **URL** (Uniform Resource Locator — see syntax [93] below) of a file or directory.



Now rather than typing an URL, you can *select it* using a file chooser dialog box:

Browse local files

Allows to choose a local file or directory using the standard file chooser dialog box.

Always works, whatever is specified in the text field.

Browse remote files

Allows to choose a file or directory found on the local file system and, if *virtual drive plug-ins have been installed*, on remote (e.g. HTTP, FTP) file systems.

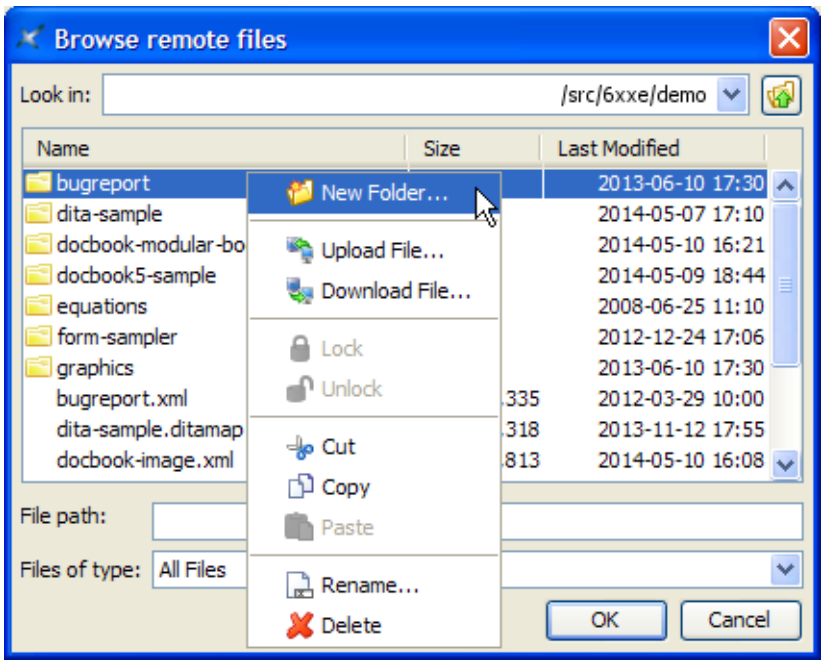
Cannot work if the text field is totally blank. You have to type[2] at least the URL of the initial directory you would like to browse.

Note that the special chooser dialog box displayed after clicking on this button has a simple yet handy, integrated file manager (similar, albeit simpler, to the "**Browse Files**" tool [72]). To use it, simply right-click anywhere on the file list:

[1] A virtual drive plug-in allows XMLmind XML Editor to edit documents stored in places other than the local file system. This is done by emulating a hierarchical file system.

All sorts of add-ons, including virtual drive plug-ins, can be downloaded and installed using **Options** → **Install Add-ons** [46].

[2] Or choose an item from the dropdown menu next to the text field. This menu lists last recently visited URLs.



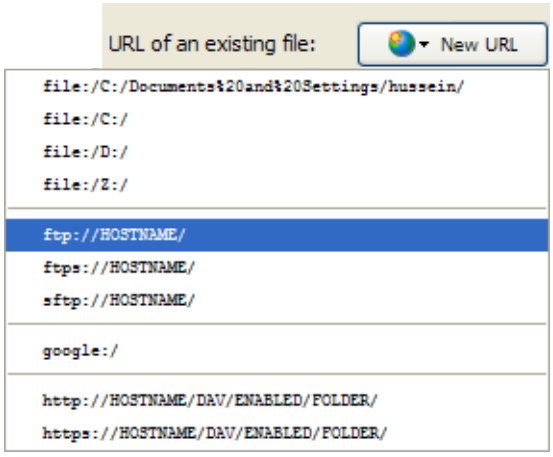
Clear

Erases the contents of the text field where the URL is typed.

The dropdown menu next to the Clear button

(Disabled unless virtual drive plug-ins have been installed.)

Instead of erasing the text field, this menu allows to replace its contents by the protocol part of an URL, e.g. "`http://`", "`ftp://`", etc. The list of these prefixes depend on which virtual drive plug-ins have been installed.



Specifying the "`http://`" URL of a document found on a WebDAV server

Assumes that WebDAV virtual drive plug-in has been installed and that the option "**Use URL chooser rather than file chooser**" has been selected.

Before using the **Browse remote files** button, make sure to type in the text field the URL of an initial directory *which is WebDAV enabled*.

Example: you have typed "http://www.acme.com/" as an initial URL. After clicking on the **Browse remote files** button, you get the following error message: "`Cannot connect to http://www.acme.com/: http://www.acme.com/ does not support WebDAV`". Try to type the URL of a subdirectory such as "`http://www.acme.com/publish/`" because, may be in the "`www.acme.com`" server, directory "`/`" is not WebDAV enabled, while directory "`/publish/`" is.

Syntax of Uniform Resource Locators

An *URL* (Uniform Resource Locator), often called *Internet Address* in Web browsers, has (to make it simple) three parts:

1. The protocol part. Example: "`http://`" in URL "`http://www.xmlmind.com/xmleditor/download.shtml`".

   Unlike Web browsers, XMLmind XML Editor requires you to specify the protocol.

2. The authority part, often a server, sometimes empty. Example: "`www.xmlmind.com`" in URL "`http://www.xmlmind.com/xmleditor/download.shtml`".

   Example of URL with an empty authority: `file:///C:/temp/documenation.zip`. Because the authority is empty, this URL could be specified as `file:/C:/temp/documentation.zip`.

3. The file path part. Example: "`/xmleditor/download.shtml`" in URL "`http://www.xmlmind.com/xmleditor/download.shtml`".

   File path components, that is, directories, are separated by forward slashes ('`/`'), whatever is the platform running XMLmind XML Editor.

   The file path of a directory must end with a '`/`'. Example: "`http://www.xmlmind.com/xmleditor/`" is the URL of directory `/xmleditor/`.

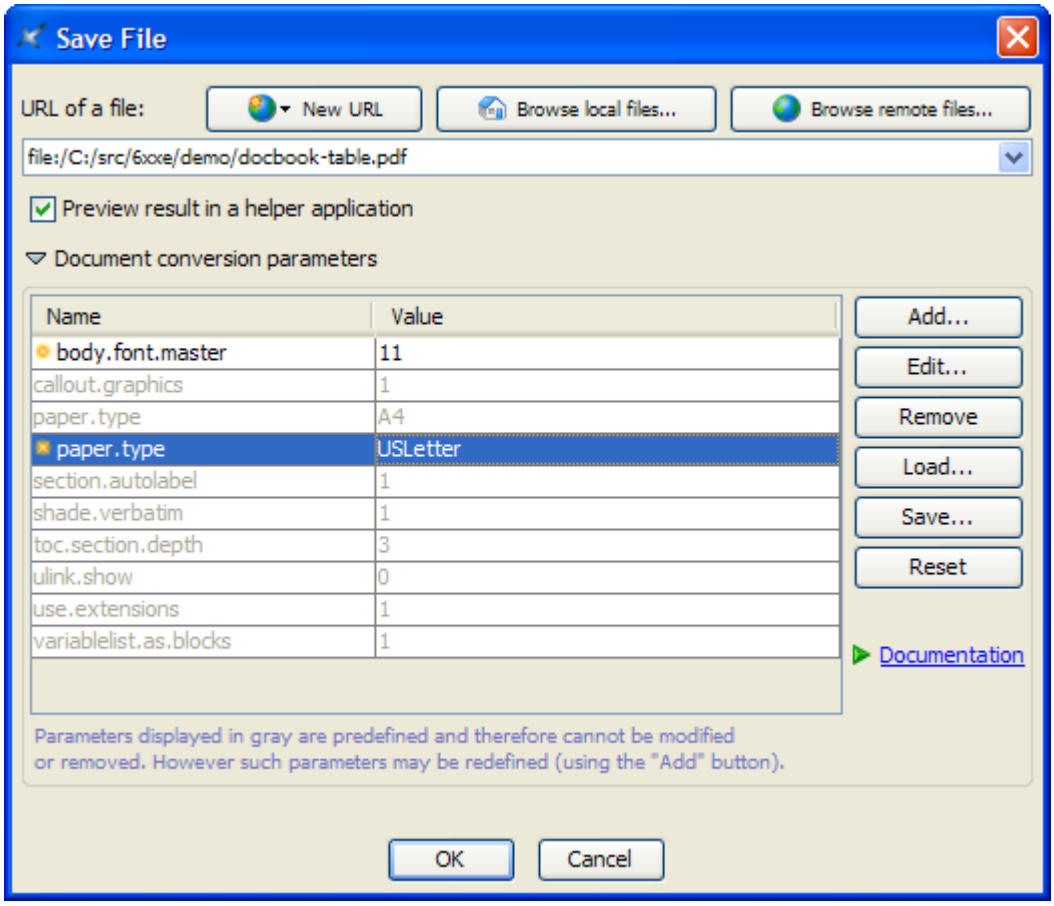
More examples:

- `http://www.xmlmind.com/store/buy.php`
- `http://blazer.inria.fr/~john/`
- `https://www.acme.com:8080/registry/customers.xml` (8080 is the port number of the `www.acme.com` Web server)
- `ftp://zenda.prisoner.tv/pub/docs/refman.zip`
- `ftp://john:sesame@zenda.prisoner.tv/pub/docs/refman.zip (same as above with a username "john" and password "sesame").`
- `file:///home/john/documentation.zip`
- `file:/home/john/`
- `file:///C:/Documents%20and%20Settings/john/documentation.zip` (spaces must be escaped as "`%20`")

1.1. The "Document conversion parameters" panel

The URL chooser dialog box is displayed when you select any of the "**Convert Document**" menu items. When this is the case, notice that the dialog box has two extra features:

*Figure 9.1. An URL chooser dialog box having an expanded "**Document conversion parameters**" panel*



1. A "**Preview result in a helper application**" checkbox. When checked, this checkbox allows to start a helper application in order to preview the generated file. For example, it allows to start Acrobat Reader in order to preview the generated PDF file. How to specify a helper application is explained in Section 6.8.5, “Helper applications options” [121].

2. A "**Document conversion parameters**" panel. This panel is initially folded to keep the dialog box as simple as possible.

The "**Document conversion parameters**" panel allows to specify the XSLT style sheet parameters (e.g. `paper.type=USLetter`) used when converting the document being edited to other formats (e.g. PDF, HTML, etc). Note that the XSLT style sheet parameters specified that way are persistent across editing sessions.

In the above screen shot:

- The user has added parameter `body.font.master=11`. (The default value of this parameter, as specified in the XSLT style sheet, is 10.)

  Notice the yellow star which precedes newly added and newly modified parameters.

- The user has redefined parameter `paper.type` to `USLetter`.

Notice how the user-defined parameter follows the predefined, grayed, parameter.

The **Documentation** link is supposed to open in your favorite Web browser the reference manual of the XSLT style sheet parameters.

Therefore, If you select a parameter by clicking on it and then click on the **Documentation** link, you should be able to read the documentation of this parameter.

However, this is not guaranteed to work because:

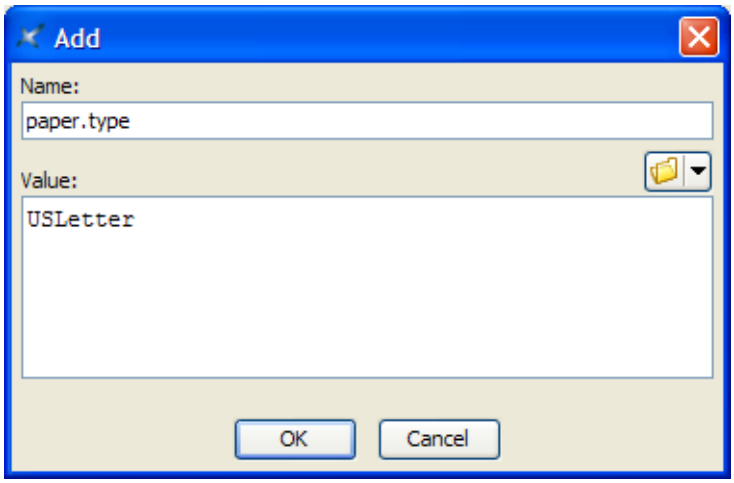
1. This generally requires an access to the Internet.
2. This documentation is not always available.

Button reference

Add

Adds a user-defined parameter.

Clicking this button displays the following dialog box.



This dialog box allows to the name of the parameter and its value. Note that the value of a parameter may contain whitespace, including newline characters.

If, for example, you want to redefine `paper.type` from `A4` to `USLetter` as shown in the screen shot of the URL chooser dialog box. Select the existing `paper.type` parameter by clicking on it then click the **Add** button. This will spare you the effort of re-typing `paper.type` as the name of the parameter you want to add.

Edit

Allows to modify selected user-defined parameter. Displays the same dialog box as the one displayed after clicking the **Add** button.

Predefined, grayed, parameters cannot be modified.

It also possible to change the *name* of the parameter here. Doing this will simply add a new parameter or replace an existing one. In fact, the **Add** and **Edit** buttons may almost be used interchangeably.

Remove

Removes selected user-defined parameter.

Predefined, grayed, parameters cannot be removed.

Load

Adds parameters loaded from an XML file previously created using the **Save** button below.

Save

Saves all user-defined parameters to an XML file.

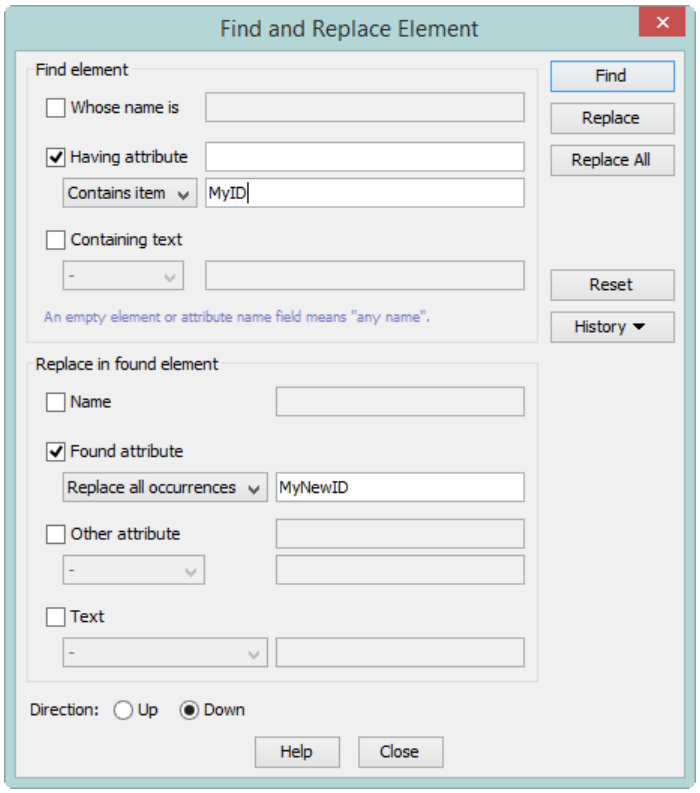
Reset

Removes all parameters.

2. The "Find and Replace Element" dialog box

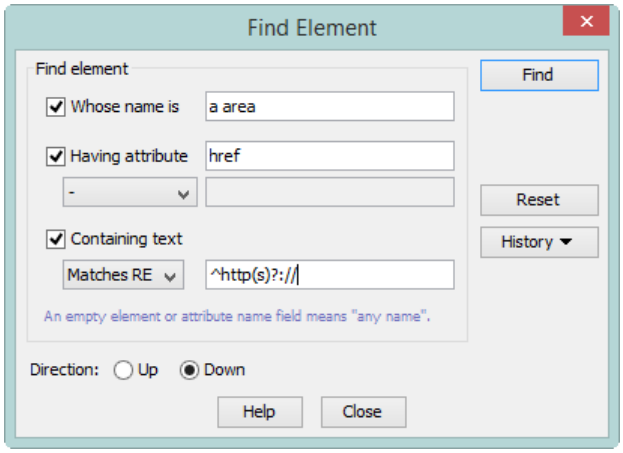
Selecting menu item **Search** → **Find and Replace Element** → **Find and Replace Element** [30] displays the following modal dialog box:

*Figure 9.2. The "**Find and Replace Element**" dialog box*



Note that selecting **Search** → **Find and Replace Element** → **Find Element** [30] displays the "**Find Element**" modal dialog box, a simplified variant of the "**Find and Replace Element**" dialog box.

*Figure 9.3. The "**Find Element**" dialog box*



"**Find element**" fields are:

Whose name is

Search for an element having any of specified names. This field may contain one or more qualified XML names (e.g. `p`, `html:li`) or wildcards (e.g. `*`, `svg:*`). The namespace prefixes, if any, used in specified names must have been declared in the document being edited. An empty field may be used to specify “any name”, which is equivalent to wildcard "`*`".

Having attribute

Search for an element having any of specified attributes. This field may contain one or more qualified XML names (e.g. `href`, `xml:lang`) or wildcards (e.g. `*`, `xlink:*`). The namespace prefixes, if any, used in specified names must have been declared in the document being edited. An empty field may be used to specify “any name”, which is equivalent to wildcard "`*`".

The combobox and the text field below the checkbox specify the attribute value searched for:

Equals

A matching attribute value is equal to the text specified in the field.

Contains

A matching attribute value contains the text specified in the field.

Contains item

A matching attribute value contains a list of items separated by whitespace, one of these items being the text specified in the field.

Matches RE

A matching attribute value contains a substring matching the regular expression pattern specified in the text field.

Containing text

Search for an element containing *directly or indirectly*[3] specified text.

The combobox and the text field below the checkbox specify the text searched for:

[3]Part or all of the searched text may be contained in *descendants* of this element.

Equals

A matching text is equal to the text specified in the field.

Contains

A matching text contains the text specified in the field.

Matches RE

A matching text contains a substring matching the regular expression pattern specified in the text field.

Direction

Up

Next search/replace operation starts at explicitly or implicitly selected node and ends at the beginning of the document (backwards search/replace).

Down

Next search/replace operation starts at explicitly or implicitly selected node and ends at the end of the document.

"**Replace in found element**" fields are:

Name

Change the name of found element to specified name (equivalent to using **Edit** → **Convert** [24]). This field contains a qualified XML name (e.g. `p`, `html:li`). The namespace prefix, if any, used in specified name must have been declared in the document being edited.

Found attribute

Replace the value of found attribute. Found attribute is the attribute that is searched for using the "**Having attribute**" field [97].

For a given element, several attributes may match the search criteria. In this case, the replace operation described below applies to all matching attributes.

The combobox and the text field below the checkbox specify the attribute change:

Replace all occurrences

Replace all occurrences of the substring searched for by the text specified in the field.

If there is no substring searched for ("-" is selected in the "**Having attribute**" combobox), set the value of found attribute to the text specified in the field.

Set to

Set the value of found attribute to the text specified in the field.

Delete

Remove found attribute.

Other attribute

Add, replace or delete an attribute other than the *found attribute* (see "**Found attribute**" above). This field contains a qualified XML name (e.g. `href`, `xml:lang`). The namespace prefix, if any, used in specified name must have been declared in the document being edited.

The combobox and the text field below the checkbox specify the attribute change:

Set to
Set the value of this other attribute to the text specified in the field.

Delete
Remove this other attribute (if it exists).

Text
Add, replace or delete some text contained in found element.

The combobox and the text field below the checkbox specify the text change:

Replace all occurrences
Replace all occurrences of the substring searched for by the text specified in the field.

If there is no substring searched for ("-" is selected in the "**Containing text**" combobox), set the text contained in the element to the text specified in the field.

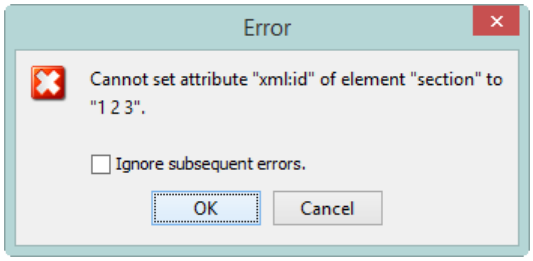
Set to
Set the text contained in the element to the text specified in the field.

Delete
Delete all the text contained in the element.

Using the "Find and Replace Element" dialog box is safe

You cannot make your document invalid by using any of the above replace operations. For example, you cannot convert an XHTML `p` to an `ul`. Other example, you cannot add attribute `xml:id="1 2 3"` to a DocBook `section`.

When one of the replace operations specified using the "**Find and Replace Element**" dialog box would make the document invalid, an error dialog box is displayed and the element in which replacements were to be made is skipped.



Buttons are:

Find
Find next element meeting specified search criteria and select it.

Replace
If currently selected element does *not* meet specified search criteria, search for next element meeting search criteria and select it (that is, behave as if button **Find** was clicked).

If currently selected element meets specified search criteria, make specified replacements (if any) in it, then search and select next element meeting search criteria.

Replace All

Starting at explicitly or implicitly selected node, make specified replacements in all elements meeting search criteria.

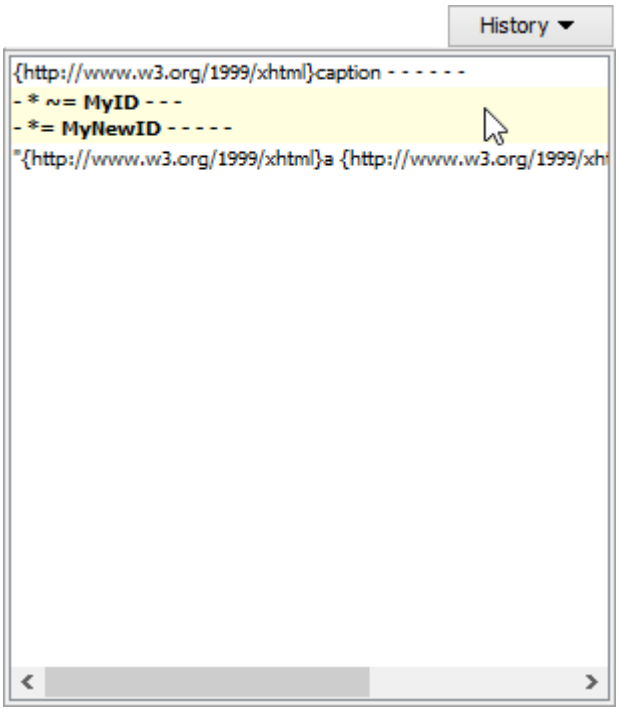
Reset

Clears all "**Find element**" fields and all "**Replace in found element**" fields.

History

Displays a list containing entries for last specified "**Find Element**" and "**Find and Replace Element**" operations.

*Figure 9.4. The popup list displayed by the "**History**" button.*



An entry corresponding to a "**Find and Replace Element**" operation is two lines long. Selecting such entry initializes both the "**Find element**" fields and the "**Replace in found element**" fields. It's possible to use this kind of history entry within the simplified "**Find Element**" dialog box but in the case, the replace operations (which were specified in the "**Replace in found element**" fields) are ignored.

An entry corresponding to a "**Find Element**" operation is one line long. Selecting such entry initializes just the "**Find element**" fields. It's possible to use this kind of history entry within the "**Find and Replace Element**" dialog box but in the case, all the "**Replace in found element**" fields are cleared.

Example 9.1. DocBook example: change the ID of an element and update all links to this element

1. Check **Having attribute**.

   a. Select **Contains item** in the combobox.

   b. Type the old ID (e.g `MyID`) in the text field.

2. Check **Found attribute**.

    a. Select **Replace all occurrences** in the combobox.

    b. Type the new ID (e.g. `MyNewID`) in the text field.

3. Click **Replace All**.

This use case is depicted in Figure 9.2, “The "**Find and Replace Element**" dialog box” [96].

Example 9.2. XHTML example: add attribute `target="_blank"` to all external links

1. Check **Having attribute**.

    a. Select **Matches RE** in the combobox.

    b. Type "`^http(s)?://`" in the text field.

2. Check **Other attribute**.

    a. Type "`target`" in first text field.

    b. Select **Set to** in the combobox.

    c. Type "`_blank`" in second text field.

3. Click **Replace All**.

3. The "Declare Namespace" dialog box

This dialog box can be used

- To declare a namespace used by the name of an attribute or an element (it is required to do so before adding the element or the attribute) or used in the value of the attribute or in the textual content of the element (for example, when the document being edited is a schema).

  In practice, this is very rarely needed because document templates generally contain declarations for all the namespaces they might use.

- To change the prefix of a namespace.

  What is called a prefix here should be called the *nickname* or the *mnemonic* of the namespace. Without nickname "`xsi`" for namespace "`http://www.w3.org/2001/XMLSchema-instance`", an attribute name such as "`xsi:schemaLocation`" would be displayed as "`{http://www.w3.org/2001/XMLSchema-instance}schemaLocation`" by the GUI of **XXE**, which is quite unreadable.

  Real namespace prefixes only exist while the document is being loaded and while the document is being saved. Do not try to add an `xmln` attribute to elements to specify an namespace and its prefix. **XXE** does this automatically each time the document is saved, the nicknames declared using this dialog box being used preferably over automatically generated prefixes such as `ns`, `ns2`, `ns3`, etc.

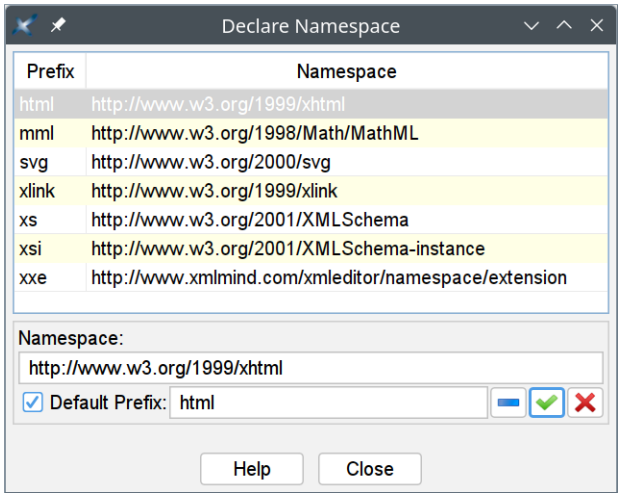
- To make a namespace the *default namespace*, that is, the namespace for which no prefix is displayed (for element names, not for attribute names).

It is not recommended to mark a namespace as being “default” if some elements in the document have names without a namespace. In such case, it would be impossible to tell if name "`title`" is "`title`" with no namespace or is in fact "`{http://www.foo.com/namespace}title`", where "`http://www.foo.com/namespace`" has been marked as being the default namespace.

However, when the current document is conforming to a DTD, this dialog box can only be used to view the namespaces and their prefixes and not to edit them.

This dialog box may display more namespaces than expected. These extra namespaces generally come from the template in *XMLmind XML Editor - Configuration and Deployment* used to create the document and/or the schema to which the document is conforming. However, when saving the document being edited, these namespaces are not necessarily *all* written to the save file.

*Figure 9.5. The "**Declare Namespace**" dialog box showing extra namespaces for an XHTML document*



Procedure for declaring a namespace

1. You may have to clear the form by clicking on the **Cancel** button
2. Specify the URI of the namespace in the **Namespace** field.
3. Specify the prefix of the namespace in the **Prefix** field.

   Specifying a prefix is mandatory even if the namespace is to be marked as “default”.
4. Optionally mark it as being the default namespace by checking the **Default** toggle.
5. Type Enter in any field or click on the **OK** button.

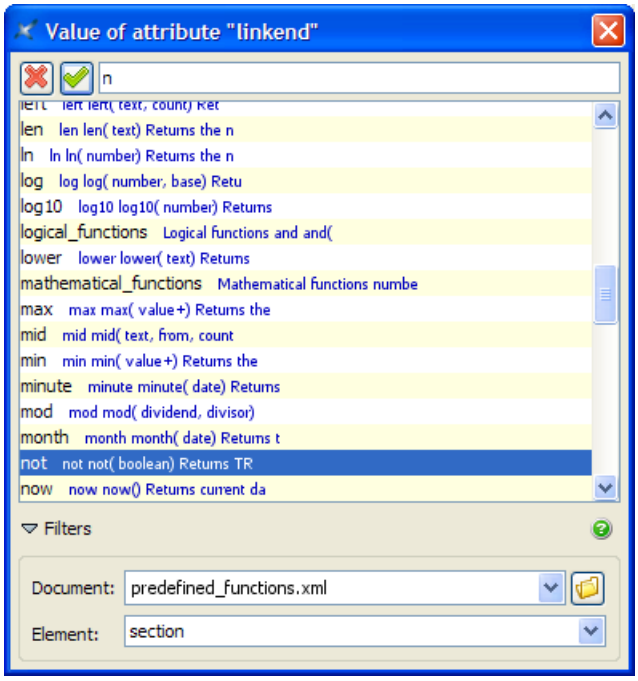
Procedure for changing the declaration of a namespace

1. Select the namespace declaration to be edited by clicking on its row in the table.
2. Specify the prefix' of the namespace in the **Prefix** field.

   Specifying a prefix is mandatory even if the namespace is to be marked as “default”.
3. Optionally mark it as being the default namespace by checking the **Default** toggle.
4. Type Enter in any field or click on the **OK** button.

4. The "Attribute Value" dialog box

This dialog box is displayed when you click the **Edit** button in the **Attributes** tool [64] and the type of the attribute being edited is an enumerated type or a reference type of any kind (`ID`, `IDREF`, `href`, etc).



- A single click is sufficient to select a value from the displayed list.
- The text field above the list supports *auto-completion*.
- For most reference types, a **Filters** pane is found below the list. Initially, this pane is collapsed.

  The **Filters** pane allows to filter the listed values by document and/or by element type.

  The **Document** combobox is automatically initialized with documents related to the document being edited:
  - If the document being edited is a module referenced or included by a *master document* [36], then you'll find in the combobox all the other documents also referenced in this Navigation Pane.
  - If the document being edited includes parts of other documents, then you'll find in the combobox all the included documents.

  The **Add** button which is next to **Document** combobox allows to add more related documents.

  For example, you have *not* opened the document set comprising `chapter1.xml` and `chapter2.xml`, you are currently editing `chapter1.xml` and want to add a link found in `chapter2.xml`. Simply use the **Add** button to add `chapter2.xml` and you'll be able to see all the link targets found in this file.

5. The "Check Spelling" dialog box

This dialog box is displayed by clicking the "**Check Spelling**" button [89] which is next to the "**Automatic Spell Checker**" button found in the status bar.

Procedure for checking current document for spelling errors:

1. Click on the **Start** button.
2. During the check spelling session, use any of the buttons described below.
3. The check spelling session can be stopped by
   - clicking on **Stop**,
   - OR typing **Esc**.

Default language

Selects the language of the dictionary used by the spell checker *when such language is not specified in the element being checked for spelling* (typically through the use of an attribute such as `lang` or `xml:lang`).

It is possible to switch from a default language to another at any time. In such case, when the default language is actually used (that is, when `lang` or `xml:lang` attributes have not been specified), the spell checker is automatically restarted with the other dictionary, beginning at the last word for which the spell checker has found an error.

The last selected default language is recorded in the user preference file in order to be automatically chosen in subsequent **XXE** sessions.

Example 9.3. Default language versus actual language example

In this example,

- `Test.xhtml` does not have `xml:lang` attributes set on its elements, which means that the default language, English, is used to spell check this document.
- Except, in `span` containing "le petit cha est mort" where attribute `xml:lang` has been set to `fr`. This means that for that element only, the default language, English, is ignored and that a French dictionary will be used.

In the above screenshot, notice that the actual language of `span` "le petit cha est mort", `fr`, is displayed at the top/right of the "**Check Spelling**" dialog box.

Replace

Replaces the erroneous word by the content of the **Replace with** text field.

An empty text field may be used to delete the erroneous word.

Replace All

Replaces all occurrences of the erroneous word by the content of the **Replace with** text field.

This button is disabled for errors other than "`Unknown word`" or "`Improperly capitalized word`".

Ignore

Skips the word for which the spell checker has found an error.

Ignore All

Skips all occurrences of the word for which the spell checker has found an error.

This button is disabled for errors other than "`Unknown word`" or "`Improperly capitalized word`".

Skip Element

Skips the element containing the word for which the spell checker has found an error.

Learn

Records the word in the personal dictionary for currently selected language.

This button is disabled for errors other than "`Unknown word`" or "`Improperly capitalized word`".

About learned words

When a spell checking session is started, it automatically knows all the words learned during past and current spell checking sessions.

A “spell checking session” is started each time:

- You click the **Start** button of the "**Check Spelling**" dialog box.
- You open a document and auto spell checking is turned on for this document.
- Auto spell checking was turned off and you manually turn it on for the document being edited. This is done by clicking the "**Automatic Spell Checker**" button found at the left of the status bar.

Limitations:

- Words learned during past and current spell checking sessions are acquired when a new session is started, and never in the middle of a session.

  Example: let's suppose auto spell checking is turned on for all documents. Document A and document B are both opened in **XXE**. You use the popup menu of the automatic spell checker to force **XXE** to learn word "`foo`" in document A. You switch to document B which also contains word "`foo`". Word "`foo`" is still underlined in document B.

- "**Ignore All**" words and "**Replace All**" words are not shared by spell checking sessions.

Start from caret

If this option is turned on, spell checking starts from the current caret position. If this option is turned off, spell checking starts from the beginning of the document.

6. The "Preferences" dialog box

The preferences specified in this dialog box are stored in *XXE_user_preferences_dir*/`preferences.properties`, a Java™ property file (an ISO-8859-1 text file using a very simply *key* :*value* format).

XXE user preferences directory is:

- *$HOME*`/.xxe11/` on Linux.
- *$HOME*`/Library/Application Support/XMLmind/XMLEditor11/` on the Mac.
- *%APPDATA%*`\XMLmind\XMLEditor11\` on Windows. Example: `C:\Users\john\AppData\Roaming\XMLmind\XMLEditor11\`.

  If you cannot see the "`AppData`" directory using Microsoft Windows File Manager, turn on **Tools**>**Folder Options**>**View**>**File and Folders**>**Show hidden files and folders**.

6.1. New options

Immediately save newly created document

When a new document is created using **File** → **New**, **File** → **Open as Template** or **File** → **Open Copy** and this toggle is checked, **XXE** will immediately prompt the user for a save file name. This save file name is used to actually create on disk the file corresponding to the new document.

Note that even when this toggle is *not* checked, **XXE** may immediately prompt the user for a save file name. This will happen for *composite document templates*, that is, document templates referencing graphics files and/or including document modules.

Default: not checked.

6.2. Open options

Use the native file chooser in preference to the multi-platform file chooser

If this toggle is checked, the native file chooser will be used in preference to Java's multi-platform file chooser.

Note that file extension filters are not supported when the native file chooser is invoked by Java™.

Also note that automatically appending the proper extension to the selected save filename (see option "**Ensure that a save filename has an extension**" [111]) is not possible when this option has been turned on.

Default: checked on the Mac, not checked on the other platforms.

When no DTD or schema, guess ignorable white space

If this toggle is checked, when opening a document not constrained by a DTD or a schema, white space characters are stripped from elements containing child elements separated by white space.

This heuristic is generally a good one for XML data.

Default: checked.

When no DTD or schema, simulate a DTD

If this toggle is checked, a dynamic, non constraining, pseudo DTD is used for documents not conforming to a DTD or schema.

The pseudo DTD simply remembers all attributes and child elements added to each element during the editing session. Thanks to this pseudo DTD, the pick lists of the **Edit** and **Attributes** tools are always filled with sensible values.

Default: checked.

Ignore <?xml-stylesheet?>

If this toggle is checked, xml-stylesheet processing instructions specifying which style sheets to use for the newly loaded document are ignored. If the **XXE** configuration associated to the newly loaded document specifies CSS style sheets, these style sheets are used instead.

Using this option is useful if `<?xml-stylesheet?>` specifies a style sheet for use by a Web browser such as Mozilla and not a style sheet for use by **XXE**.

Default: not checked.

Inform about non-editable document parts

If this toggle is checked and if the document being opened is modular (that is, contains references to nodes coming from other documents), **XXE** display a dialog box informing the user that included nodes have been marked as non-editable.

Default: checked.

Lock documents stored on the local filesystem

If this toggle is checked, the file containing the document being opened in the editor is automatically locked using an *advisory, application-level, lock*.

In practice, this means that:

- Applications other **XXE** can open and modify the document being edited in an instance of **XXE**.
- If you try to open in **XXE** a document being edited in another instance of **XXE** (run by you or by one of your coworkers), **XXE** will inform you that the file is locked and will tell you who is locking it. After that you'll be able to open the document but, if you modify it, you'll have to save it to another location.

Caveat: In order to use this facility, you and *all* your coworkers possibly editing the same files, need to turn this option on. If one of your coworkers forgets to do it, **XXE** will not be able to detect that the file being opened is locked, and in such case, your coworker may overwrite a document being edited by another person.

How to forcibly remove a lock

A lock is implemented by the means of a hidden text file containing information about who has locked the corresponding document.

On Windows, the lock file corresponding to document `C:\doc\doc.xml` is `C:\doc\doc.xml.LOCK` (with the hidden attribute set on this file).

On the other platforms, the lock file corresponding to document `/home/john/doc/doc.xml` is `/home/john/doc/.doc.xml.LOCK`.

Manually deleting the lock file forcibly removes the lock set on the corresponding document. Note that it is harmless to delete such lock files and this, at any time.

Default: not checked.

When possible, lock documents stored on remote filesystems

Unlike the above toggle, this form applies only to files stored on filesystems other than the local one (WebDAV, FTP, Zip, etc).

Preferred lock mode

Specifies the type of lock [108] used for the document being edited, *when the underlying document store supports locking*. Currently, only WebDAV supports document locking.

Note that the mode specified here is just a wish. **XXE** will adapt the lock mode to what is actually supported by the underlying document store. For example, if the **Shared** mode has been chosen by the user and the underlying document store only supports **Exclusive** locking, the document being edited will be locked using an **Exclusive** lock.

Default: Exclusive

Identifier as a lock owner

Specifies a free form string identifying you as the person locking a document. This string is presented to other users wishing to know who is locking a document. It is recommended to specify your email address (example: `jdoe@acme.com`).

Default: empty string which implies *`user_name`*`@`*`host_name`* (example: `jdoe`@fast.acme.com)

Table 9.1. Lock Types

Lock type	Description
None	The document being edited in not locked.
Shared	The document being edited is locked using a *shared* lock. A shared lock does not prevent you from overwriting a document shared-locked by another author but a least, you are informed that the document is being ``used'' by this other author. (In such case, you'll generally want to contact him to decide what to do.)
Exclusive	The document being edited is locked using an *exclusive* lock. An exclusive lock prevents you from overwriting a document exclusively-locked by another author.

6.3. Save options

XML formatting

Some **XXE** configurations specify a number of XML formatting options by using the `saveOptions` in *XMLmind XML Editor - Configuration and Deployment* configuration element. MathML example:

```
<saveOptions addOpenLines="false" predeclarePrefixPreferences="false" />
```

The options found below may not be used to override what's specified in the `saveOptions` configuration element. For example, even if you check "**Add open lines**", MathML files will not be saved with extra open lines.

Encoding

Specifies the encoding used for XML files saved by **XXE** (if save options have not been specified in an **XXE** configuration file — see below).

Default: the original encoding of the file.

Save characters outside encoding as entity references

If this toggle is checked, all characters not supported by the encoding are saved as entity references.

Example: the Euro sign is not supported by the ISO-8859-1 encoding. If this toggle is checked, the Euro sign is saved as "`€`". If this toggle is not checked, the Euro sign is saved as "`€`".

Of course, for a character to be saved as an entity reference, the corresponding entity must have been defined in the DTD.

Example: the Euro sign is not supported by the ISO-8859-1 encoding. If there is no entity defined for this currency symbol, the Euro sign is saved as "`€`" whether the toggle is checked or not.

Default: checked.

In addition to characters outside the encoding, save the following characters as entity references

Specifies which characters, even if they are supported by the encoding, are always saved as entity references.

Example: the Copyright sign is supported by the ISO-8859-1 encoding but you may prefer to see it saved as "`©`". In such case, specify `"169"` in this text field.

This attribute contains a list of character ranges. A character range is either a single character or an actual range *`char1`*`:`*`char2`*.

A character may be specified using its Unicode character number, in decimal (example: `233` for e acute), in hexadecimal (example: `0xE9`) or in octal (example: `0351`).

Because names are easier to remember than numbers, a character may also be specified using its entity name as defined in the DocBook 4.2 DTD (example: `eacute`). Note these symbolic specifications are supported whatever is the DTD or schema of the document being saved.

Examples: `nbsp` `160` `0xA0` `0240` `reg` `174` `0x00ae` `0256` `pound:yen` `163:165` `0xA3:0xA5` `0243:0245`

Default: empty.

Ignored if toggle "Save characters outside encoding as entity references [109]" is not checked.

Indent
: If this toggle is checked, XML files saved by **XXE** are indented (if save options have not been specified in an **XXE** configuration file -- see below).

The fields and toggles that follows in the **Indent** frame may be used to parametrize indentation.

Default: checked.

Indentation
: Specifies the number of space characters used to indent a child element relatively to its parent element.

Default: 2.

Max. line length
: Specifies the maximum line length for elements containing text interspersed with child elements.

Default: 78.

This value is only used as a hint: XML files created by **XXE** may contain lines much longer than the specified length.

Add open lines
: If this toggle is checked, an open line is added between the child elements of a parent element (if the content model of the parent only allows child elements).

Default: checked.

Do not indent documents not conforming to a DTD or schema
: If this toggle is checked, when saving documents not conforming to a DTD, W3C XML Schema or RELAX NG schema, XML files generated by **XXE** are not indented, and this, even if the **Indent** toggle is checked.

Note that when this toggle is not checked, **XXE** uses very simple heuristics to indent unconstrained documents. Indenting such documents this way may add white space to places where it is significant.

Default: not checked.

Autosave feature

Automatically save modified documents
: If this toggle is checked, **XXE** will automatically save modified documents.

Default: not checked.

Max. modifications before saving
: A modified document will be automatically saved after specified number of modifications. Typing a character counts as a modification.

Default: 300.

Idle time (seconds) before saving

If a modified document is no longer edited during specified number of seconds, this document is automatically saved, even if the "**Max. modifications before saving**" count has not yet been reached.

This option allows to make the auto-save feature as non-intrusive as possible.

Default: 30.

Other options

Ensure that a save filename has an extension

If this toggle is checked, **XXE** automatically appends an extension to the save filenames (**File** → **Save As**, **File** → **Save Copy**, etc) having no extension at all.

The extension is guessed based on the current filename of the document to be saved. For example, if you save `/tmp/page.html` as `/home/john/index`, **XXE** automatically appends "`.html`". When the extension cannot be guessed this way, **XXE** fallbacks to "`.xml`".

Note that automatically appending the proper extension to the selected save filename is not possible when option "**Use the native file chooser in preference to the multi-platform file chooser**" [106] has been turned on.

Default: checked.

Before saving, make a backup copy of the file

If this toggle is checked, a copy of the original file is made (same name but ending with '~') before saving the modified copy.

Note that backup files are created at the beginning of the editing session and not each time a file is saved. This makes the backup files much more useful. For example a **diff** between `foo.xml~` and `foo.xml` will show you what you did during last editing session.

Also note that this feature is supported only for save files stored on the local file system. For example, backup files will never be created on a WebDAV server.

Default: checked.

Automatically update references in modular documents

If this toggle is checked, references found in modular documents are automatically updated each time a referenced document is saved to disk.

Example: document `book.xml` references `chapter1.xml` and `chapter2.xml`. File `chapter2.xml` is modified and saved to disk using **XXE**. Nodes included in `book.xml` coming from `chapter2.xml` are automatically updated.

Counter-example: document `book.xml` references `chapter1.xml` and `chapter2.xml`. File `chapter2.xml` references `section1.xml` and `section2.xml`. File `section2.xml` is modified then saved to disk using **XXE**. Nodes included in `book.xml` *indirectly* coming from `section2.xml` are *not* automatically updated. In such case, if you really want to update `book.xml`, you'll have to use **View**+**Redraw** (**Ctrl**+**L**) See note about Updating the references contained in a modular document [32].

Default: checked.

6.4. Print options

The options for the page footer are not described here because they are identical to those used for the page header (described below).

Begin

The page header has 3 areas: begin (the left for left-to-right languages), middle, end (right). This field specifies the text printed at the left of the page header.

Default: empty.

Each area can contain a mix of text and variables [112]

Middle

Specifies the text printed at the center of the page header.

Default: empty.

End

Specifies the text printed at the right of the page header.

Default: empty.

Color

Specifies the color of the text of the page header.

Default: gray.

Note that the font used for the page header is the default font of the style sheet (see the View options [116] below).

Underline [Overline]

Specifies if a thin line is to be printed below the page header [above the page footer].

Default: checked.

Table 9.2. Substituted Variables

Variable	Description
`%F`	File name of the document being edited
`%f`	Same as `%F` but shortened to approximately 30 characters
`%B`	Base name of the document being edited
`%D`	Current date
`%T`	Current time
`%P`	Equivalent to localized "`page %I of %C`"
`%I`	Current page number
`%C`	Total page count

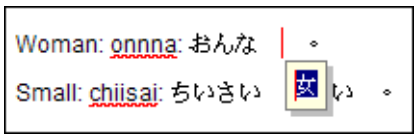
6.5. Edit options

Use integrated input method support

When an input method (e.g. Japanese Romaji Input) has been selected by the user,

- If this toggle is checked, a composition area, managed by the XML editor, is displayed inside the document view, below the caret.
- If this toggle is not checked, a composition window, managed by the operating system, is displayed outside the main window of the XML editor.

Figure 9.6. Composition area displayed below the caret



Default: not checked. (Slightly more efficient if you never use input methods.)

This feature cannot be made to work on Linux. It should work fine only on Windows and on the Mac.

Clicking with middle button pastes system selection

If this toggle is checked, clicking with the middle button (or with the mouse wheel) pastes the characters copied to the ``system selection".

On platforms not supporting system selection (all but generic Unix/Linux), this action pastes the content of an internal clipboard.

Default: not checked.

Append mode

Specifies how to quickly complete what is being typed in an autocompletion-enabled text field.

None

There is no quick way to complete what is being typed (other than using Up and Down arrows).

Automatic

What is being typed is automatically completed to compose first possible suggestion.

Manual

The user needs to explicitly press on the space bar (or on **Ctrl**+Space, if some of the choices contain whitespaces) to complete as much as possible what has been typed.

Example: the list of choices is "`aaz aaa bbz bbb ccz ccc`". The user types "`b`".

None

Nothing happens and pressing on the space bar has no effect.

Automatic

The text field is automatically updated to contain "`bbz`".

Manual

Nothing happens, but pressing on the space bar will add a "b" to what has been typed (which gives "bb").

Default: **Manual**.

In the choice list, only show suggestions

If this toggle is checked, the list which displays possible choices (e.g. elements to be inserted in the document) is kept as short as possible.

Default: not checked.

By default, such list displays all possible choices, whether matching what has been typed in the autocompletion-enabled text field or not.

Example: the list of choices is "aaz aaa bbz bbb ccz ccc". The user types "b".

If this toggle is not checked, the list contains "aaz aaa bbz bbb ccz ccc" and item "bbz" is highlighted.

If this toggle is checked, the list contains "bbz bbb" and item "bbz" is highlighted.

Automatically select an attribute

When toggle **Never** is not checked, **Tools** → **Edit Attribute** (**Ctrl**+**E**) automatically selects an attribute of the element. The selected attribute is determined as follows, in this order:

1. First required attribute containing the "???" placeholder value.

2. First attribute, required or not, containing the "???" placeholder value.

3. If toggle **Unique required attribute** is checked, *unique* required attribute, whatever its value.

Default: **First attribute containing a placeholder value**.

Max. undo actions

Specifies the maximum number of undo (redo) actions a user will be able to perform. Limited to 100 because a single undo action may consume a great deal of memory.

Default: 20.

Allow advanced use of XInclude

By default, command **Edit** → **Reference** → **Copy as Reference** [27] allows to copy as a reference only the root element of a document or an element having an ID. If this toggle is checked, it becomes possible to copy as a reference any range of sibling nodes.

Default: not checked.

Warn about advanced use of XInclude

This checkbox is disabled unless "Allow advanced use of XInclude [114]" has been checked.

If this toggle is checked, you'll be informed when what you are copying as a reference qualifies as "an advanced use of XInclude". Typically you'll be informed that the element you are copying as a reference has no ID attribute.

If you check Allow advanced use of XInclude [114], it is also strongly recommended to also turn this option on.

Default: not checked.

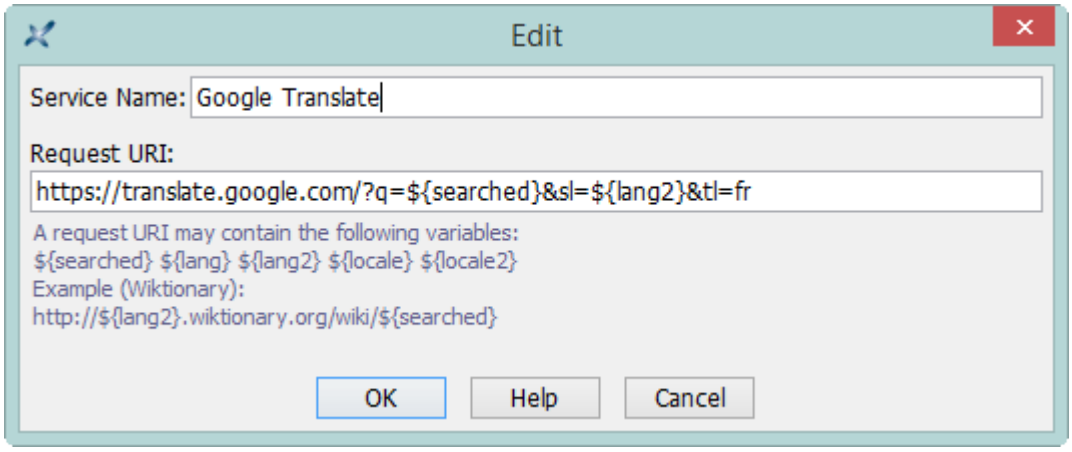
6.6. Web search options

This pane allows to configure the entries of the **Web Search** menu [31].

- Clicking **Add** or **Edit** displays a dialog box (see below) allowing to add or modify a *web search service specification*.
- Clicking **Remove** removes the web search service specification selected in the list.
- Clicking **Reset** restores the stock list[4] of web search service specifications.

By default, this stock list contains an entry for "**Google Translate**". This entry references an URI which translates some text to French (the URI ends with "`&tl=fr`"). If you want to specify another target language, simply replace the language code for French, `fr`, by another language code. For example, Italian: `https://translate.google.com/?q=${searched}&sl=${lang2}&tl=it`.

Figure 9.7. Dialog box allowing to add or edit a web search service specification



A web search service is specified by:

- A (case-insensitive) symbolic name. Example: `Google`.
- A request URI containing variables. Example: `https://www.google.com/search?q=*${searched}*&hl=*${locale2}*&lr=lang_*${lang2}*`.

The following variables are substituted with their values when the web browser is invoked by **XXE** to open the URI:

Variable	Description
`${searched}`	The searched text is the current text selection.
`${locale}`	The language of the user interface (GUI) of **XXE**.
`${locale2}`	Same as `${locale}`, but just the first two ISO 639-1 letter codes. Example: if `${locale}` is `fr-CA`, then `${locale2}` is `fr`.

[4]This list is specified in system property WebSearch.services in *XMLmind XML Editor - Commands*. When this system property has not been defined, a default list comprising Google, Wikipedia, etc, is used.

Variable	Description
`${lang}`	The language of searched text. This language is automatically determined by **XXE** using a number of rules in *XMLmind XML Editor - Commands* (for example, lookup of attribute `xml:lang`).
`${lang2}`	Same as `${lang}`, but just the first two ISO 639-1 letter codes.

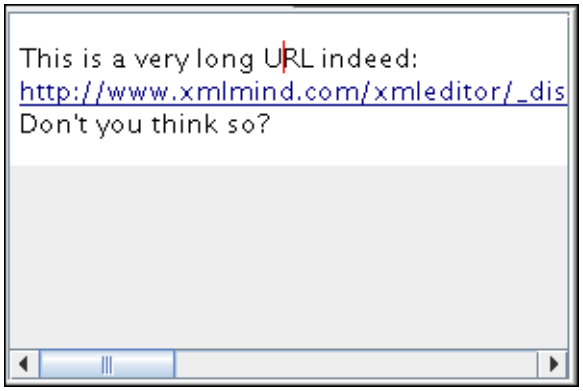
Note that the value of a variable is automatically escaped if needed too. For example, "`vin rosé`" becomes "`vin%20ros%C3%A9`".

Notice that this pane does not allow to specify keyboard shortcuts. If you want to use keyboard shortcuts for searching the web, please refer to Using keyboard shortcuts for searching the web [31].

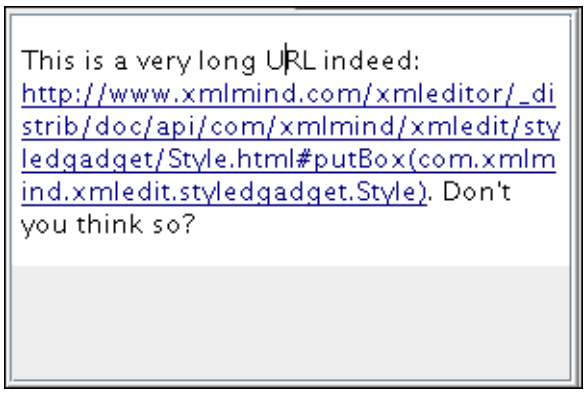
6.7. View options

Wrap words wider than available space

If this toggle is checked, words which are wider than the space available in the document view, are wrapped. For example, when this toggle is **not** checked (note the horizontal scrollbar):



When this toggle is checked:



This option, which slightly modifies the behavior of word wrap in both the tree view and in the styled view, is useful to deal with long URLs and with languages such as Japanese or Chinese which do not use spaces between words.

Default: not checked.

6.7.1. "CSS stylesheet defaults" subsection

The following options parameterize the CSS style sheet used to visualize the document or to print it.

Setting some of these preferences will have no visible effect if the style sheet author has specified the corresponding properties in the style sheet. For example, if the user's preferred background is specified in the **CSS stylesheet defaults** subsection as being light yellow and if the style sheet author has specified the root element `background-color` as being white, the document will be rendered with a white background.

"serif" font family
: Specifies the font family used for property value `font-family:serif`.

  Default: Serif (the Java™ default serif font family).

"sans-serif" font family
: Specifies the font family used for property value `font-family:sans-serif`.

  Default: SansSerif (the Java™ default sans-serif font family).

"monospace" font family
: Specifies the font family used for property value `font-family:monospace`.

  Default: Monospaced (the Java™ default monospaced font family).

"cursive" font family
: Specifies the font family used for property value `font-family:cursive`.

  Default: Serif (the Java™ default serif font family).

"fantasy" font family
: Specifies the font family used for property value `font-family:fantasy`.

  Default: Serif (the Java™ default serif font family).

Default font family
: Specifies the default value for property `font-family`.

  Default: the "sans-serif" font family.

Default font size
: Specifies the default value for property `font-size`. See also **View** → **Text Size** [33].

  Default: 11pt. Unit is “CSS points”, a unit which is independent of the platform, screen, display scaling, etc.

Default background
: Specifies the default value for property `background-color`. (Displays standard color chooser dialog box.)

  Default: white.

Default text color
: Specifies the default value for property `color`. (Displays standard color chooser dialog box.)

Default: black.

6.7.2. "Tree view settings" subsection

The following options may be used to configure the fonts and colors of a tree view.

Base font size

The sizes of all the fonts used to render a tree view are computed relatively to this font size. See also **View** → **Text Size** [33].

Default: 10pt. Unit is “CSS points”, just like **Default font size** [117], a unit which is independent of the platform, screen, display scaling, etc.

Fonts and colors

Displays a dialog box [135] allowing to specify the fonts[5] and colors used to render a tree view.

These fonts and color settings apply not only to a “stand-alone” tree view, but also the “mini” tree views embedded in a styled view (e.g. DocBook `colspec` and `spanspec` elements are rendered as “mini” tree views within the styled view).

6.7.3. Preferences related to images

Load images in a background task

Specifies whether the images referenced by a document should be loaded after the body of the document, in a background task.

No

The images are loaded at the same time as the body of the document. This means that when opening a document, you'll not see the body of the document until all the referenced images are loaded too. This option is well-adapted to images stored on the local filesystem, which is generally fast.

Yes

The images are loaded after the body of the document, in a background task. This means that when opening a document, you'll quickly see the body of the document and after that, each image referenced in the document will be revealed one after the other.

This option is well-adapted to images stored on remote filesystems (WebDAV, FTP, etc), which are often slow.

This option is also very useful if you author documents stored on the local filesystem, but containing a large number of images.

Remote images only

Images stored on remote filesystems (e.g. WebDAV, FTP, etc) are loaded after the body of the document, in a background task. Images stored on the local filesystem are loaded at the same time as the body of the document.

Default: **Remote images only**.

[5] Font family and font style, but not the font size which is computed using "**Base font size**" [118].

6.8. Tools options

6.8.1. Master document options

Master documents

Lists all XML documents which should be automatically made master documents [36]. This list applies to documents which otherwise would not be automatically made master documents, typically DocBook modular documents.

Maps (e.g. DITA map, DocBook assembly) are automatically made master documents

If checked, maps of all kinds (that is, DITA maps, DocBook 5.1 assemblies and Ebooks) are automatically made master documents.

Default: checked.

Except for the following maps

Lists all XML documents which should *not* be automatically made master documents [36]. If above "**Maps are automatically made master documents**" is checked, this list applies to maps which otherwise would be automatically made master documents.

Click **Remove** to remove selected entry from a list. Click **Remove All** to clear a list.

Note that removing an entry from a list has no immediate effect on an already opened master document. You'll have to close the master document and then reopen it to see the effect.

6.8.2. Validate options

Filter duplicate ID errors found in modular documents

Pasting in a document two references to the same element will cause the **Validity** tool to display duplicate ID errors. This is a problem because the author didn't do any mistake by pasting these two references and because too many of these ``false'' duplicate ID errors may hide real duplicate ID errors. That's why turning on this option will discard such ``false'' duplicate ID errors from the **Validity** tool.

Default: checked.

Automatically show Validity tool

Document validity is automatically checked each time a document is opened or saved to disk. If this toggle is checked, when validity errors are found in the document to be opened or saved, the **Validity** tool “tab” is automatically selected in order to display the error messages.

Default: checked.

Only for the most severe validity errors

Ignored unless "**Automatically show Validity tool**" is checked. If this toggle is checked, do not automatically select the **Validity** tool “tab” when the document being checked for validity has just cross-reference errors or semantic warnings.

Default: checked.

6.8.3. Spell options

Activation of the automatic (AKA on-the-fly) spell checker:

Never activate the automatic spell checker
: When a document is opened, never activate the automatic spell checker, even if this is specified in the configuration file associated to the opened document.

  Default: not checked.

Always activate the automatic spell checker
: When a document is opened, always activate the automatic spell checker, even if this is *not* specified in the configuration file associated to the opened document.

  Default: not checked.

Activate if this is specified in the configuration file
: When a document is opened, activate the automatic spell checker if this is specified in the configuration file associated to the opened document (using the `spellCheckOptions` configuration element, see Section 29, “spellCheckOptions” in *XMLmind XML Editor - Configuration and Deployment*).

  Default: checked.

6.8.4. Revisions options

You may need to enable this group of options by checking "**Enable the 'Tools|Revisions' Submenu**" in **Options** → **Preferences**, **General**|**Features** section.

When a document is created or opened:

Do not automatically keep information about revisions
: Do not do anything special to deal with document revisions when a document is created or opened in **XXE**.

  Default: checked.

Automatically enable the comparison of revisions
: When a document is created or opened in **XXE**, automatically add information[7] allowing to compare two revisions of this document.

  Default: not checked.

Automatically store all revisions in the document
: When a document is created or opened in **XXE**, start storing all the revisions[8] in the XML file containing this document.

  Turning on this option implies turning on option "**Automatically enable the comparison of revisions**". If all revisions are stored in the XML file, then you'll be able to use open them using **Tools** → **Revisions** → **Open Revision** [41] and to compare them using **Tools** → **Revisions** → **Compare Revisions** [42].

  Default: not checked.

6.8.5. Helper applications options

File types

List of file types. Each file type has an associated helper application. This helper application is assumed to be able to open files detected as having this type. A helper application may be a viewer or an editor.

The following buttons may be used to modify this list:

Add

Displays the "Helper Application Editor" dialog box [136] in order to add a new file type to the list.

Edit

Displays the "Helper Application Editor" dialog box [136] in order to view or modify selected file type.

Remove

Removes selected file type from the list.

Default viewer

Specifies which default viewer to use in case the type of the file to be opened has not been detected. In practice, commands making use of the default viewer typically assumes that it is in fact *a Web browser*. This implies that these commands assume that a default viewer can open URLs as well as filenames and that it can open text, HTML, GIF, PNG and JPEG files.

This field must contain a command line interpreted by the native shell of the platform: `cmd.exe` on Windows and `/bin/sh` on the Mac and on Unix.

This command line must reference one of these two substituted variables: `%U` and `%F`. In principle, `%U` is replaced by the URL of the file to be opened by the helper application and `%F` is replaced by a filename. In practice, `%U` is just a *hint* meaning: the helper application can open URLs as well as filenames.

Default: depends on the platform:

- On Windows: `start "" "%U"`
- On the Mac: `open "%U"`
- On Unix: dynamically detected. For example: `firefox "%U"`

The following buttons may be used to modify this field:

Reset

Resets the field to its default value (see above).

Choose

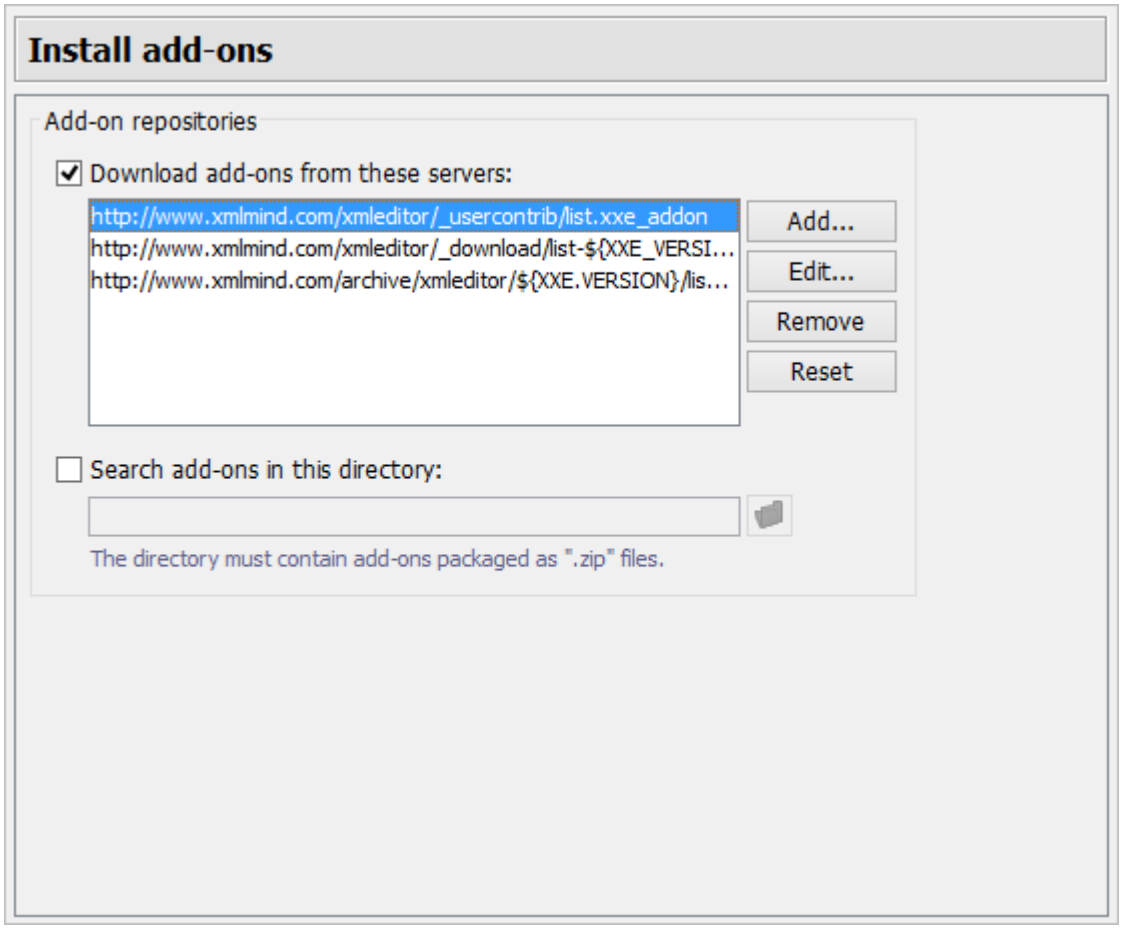
Displays the standard file chooser in order to specify an application (e.g. a `.exe` or a `.bat` file on Windows). String `" "%F""` is automatically appended to the chosen application.

See also Helper applications on the Mac [139].

6.9. Install add-ons options

This preference sheet is absent when XMLmind XML Editor has been started using Java™ Web Start or as an applet.

*Figure 9.8. The "**Install add-ons**" preferences sheet*



Download add-ons from these servers

Add-ons available for download are listed in `.xxe_addon` files. The list of “servers” below this checkbox contains the URLs of these `.xxe_addon` files.

- Use button **Add** to add an URL to the list.
- Use button **Edit** to modify the selected URL.
- Use button **Remove** to remove the selected URL from the list.
- Use button **Reset** to reset the list to the default list below.

Default: checked. The default list of “servers” is:

- `http://www.xmlmind.com/xmleditor/_usercontrib/list.xxe_addon`

  Lists all user-contributed add-ons.

- `http://www.xmlmind.com/xmleditor/_download/list-`*`${XXE_VERSION}`*`.xxe_addon`

  For example, variable *`${XXE_VERSION}`* is replaced by "`10_7_0`" if the version of **XXE** is 10.7.0.

  This location exists only for the most recent version of **XXE**.

- `http://www.xmlmind.com/archive/xmleditor/`*`${XXE.VERSION}`*`/list-`*`${XXE_VERSION}`*`.xxe_addon`

  For example, variable *`${XXE.VERSION}`* is replaced by "`10.7.0`" if the version of **XXE** is 10.7.0.

  This location exists only for any version older than the most recent version of **XXE**.

Search add-ons in this directory

Zip files containing add-ons may have been downloaded using a Web browser and then copied to a local directory (for example, because you have problems downloading add-ons directly from **XXE**). In such case, check this box and specify in the text field below it this local directory.

Note that this directory may also contain unzipped add-ons (not recommended though). This is useful because sometimes Web browsers have the bad habit of automatically unzipping the downloaded Zip files.

Default: not checked. No local directory containing Zip files.

6.10. General options

Use a unique instance of XMLmind XML Editor

If this toggle is checked, a unique instance of XMLmind XML Editor is used to open all your XML documents. For example, if you double-click on the icon of an XML document in the file ``explorer'', the running instance is used to open this XML document. When this toggle is not checked, a new instance of **XXE** is started in order to load the document clicked upon.

If there is no running instance of **XXE**, a new instance is started. If the running instance is hidden by other windows or is iconified, it is made visible before opening the requested document.

This feature also works fine from the command line. For example, on Linux, if this toggle is checked, executing "`xxe mydoc.xml &`" will cause the running instance to be used to open `mydoc.xml`.

If you need to change this option, make sure to close all running instances expect one. Then change the option in the last instance and close it. Then restart **XXE**.

Default: not checked.

Port

Specifies the TCP/IP port used by different instances of **XXE** to communicate with each other. It is highly recommended to use a port in the dynamic/private range: 49152 to 65535.

Default: 49987

Automatically reopen last opened documents

This option is examined just after **XXE** is started, when no documents to be opened have been explicitly specified (e.g. by double-clicking on an XML file or by using command line options).

When turned on, this option forces **XXE** to reopen all the last documents opened during the preceding editing session. The tab order and view layout of these documents, as well as which document was the active one, are restored too. However note that if several main windows [15] were opened, it's only the session of the last closed main window which is restored.

See also the -last [192] command line option for a simpler alternative.

Display scaling

May be used to globally change the size of all the items comprising the user interface of **XXE**.

Default: system settings.

- When Java™ natively supports the HiPDI screen of the computer running **XXE**, the "**Use custom settings**" slider is disabled (grayed). When this is the case, the only way to change the display scaling of **XXE** is to change the display scaling of the operating system using the control panel.
- On some Linux computers having HiDPI screens, HiDPI is not automatically detected nor natively supported by Java™. When this is the case, you'll have to use the "**Use custom settings**" slider. Fortunately, this setting is done once for all.
- Only the “system” Look&Feel is guaranteed to support a display scaling larger than 100%. This system Look&Feel corresponds to the "**(default)**" choice of the **Style** combobox [124]. For example, the Nimbus Look&Feel is not guaranteed to support a display scaling larger than 100%.

Locale

May be used to force the language used in **XXE** menus and dialog boxes. For example, may be used to force the use of English on a machine where the default locale is German.

Default: default locale of the machine running **XXE**.

Style

May be used to change the Look&Feel of **XXE** user interface. "**(default)**" is the FlatLaf Look&Feel, light theme, on Linux and the system Look&Feel on the other platforms. "**(system)**" is the cross-platform Look&Feel —AKA Metal— on Linux and the system Look&Feel on the other platforms.

Default: "**(default)**".

6.10.1. Features Options

Most features are preselected by default. If you don't need a feature, it's strongly recommended to uncheck the corresponding toggle and then restart **XXE**. Doing this simplifies the user interface and slightly improves **XXE** startup time.

Evaluate XMLmind XML Editor Professional Edition features

This check box is found only in XMLmind XML Editor Personal Edition™. This check box is initially checked to make it easy evaluating the features implemented by the **File** → **Import DOCX** menu entry and the **Convert Document** submenus.

However, if you are not evaluating XMLmind XML Editor Professional Edition and just want to use a free, streamlined, fully functional, XML editor, simply uncheck this check box, restart **XXE** and the **File** → **Import DOCX** menu entry and the **Convert Document** submenus will go away.

Feature names: `ImportDOCX`, `ConvertDocument`.

Enable the "Window|New Window" menu item

Check this toggle to enable the **Window** → **New Window** menu item [15].

Enabling this feature is recommended if you work on many documents at the same time.

Feature name: `NewWindow`.

Enable the "File|Folder" submenu

Check this toggle to enable the **File** → **Folder** submenu [17].

Enabling this feature is recommended if you routinely have to manipulate many files or if you need to search files by their names and/or by their XML contents or if your files are stored on remote servers.

Feature name: `OpenFolder`.

Enable the "Edit|Force Deletion" menu item

Check this toggle to enable the **Edit** → **Force Deletion** menu item [23] and also the "**Force Removal**" popup menu item [65] in the **Attributes** tool.

Enabling this feature is not recommended as it is almost never useful and can make your document invalid.

Feature name: `ForceDeletion`.

Enable the "Search|Web Search" submenu

Check this toggle to enable the **Search** → **Web Search** submenu [31].

Enabling this feature is recommended if you are looking for very convenient writing tools.

Feature name: `WebSearch`.

Enable the "View|Display Images" submenu

Check this toggle to enable the **View** → **Display Images** submenu [33].

Enabling this feature is recommended if your documents include many high-resolution images.

Feature name: `ImageViewportMode`.

In the styled view, render audio and video elements by embedding a media player

When this feature is enabled, XHTML 5, DITA and DocBook 5.1 audio and video elements are rendered by embedding a media player in the styled view. This media player is similar to those found in Web browsers.

Enabling this feature has no effect whatsoever if the Java™ runtime used to run **XXE** does not support JavaFX®, hence is older than version 1.8.0_40.

Also note that JavaFX® is not supported on Windows XP because Java™ 1.8 is not officially supported by Oracle® on this operating system.

Feature name: `MediaPlayer`.

Enable the "Tools|Use as Master Document" menu item

Check this toggle to enable the **Tools** → **Use as Master Document** menu item [36].

Enabling this feature is recommendedRecommended if you author modular documents as this feature makes it easy and safe creating and checking cross-references in module documents.

This feature is *required* if you use attributes `keyref` and/or `conkeyref` in your DITA documents.

Feature name: `MasterDocument`.

Enable the "Tools|Remark" submenu

Check this toggle to enable the **Tools** → **Remark** submenu [39].

Enabling this feature is recommended if you need to comment documents written by others.

Feature name: `Remark`.

Enable the "Tools|Revisions" submenu

Check this toggle to enable the **Tools** → **Revisions** submenu [40] and the **Options** → **Preferences**, **Tools**|**Revisions** group of options.

Enabling this feature is recommended if you need to review the changes made to your documents by others.

Feature name: `Revisions`.

Enable the "Tools|Record Macro" submenu

Check this toggle to enable the **Tools** → **Record Macro** submenu [44] and the **Tools** → **Execute Command** menu item [38].

Enabling this feature is recommended if you are a power user who wants to automate sequences of actions.

Feature name: `Macro`.

Enable all advanced tools including the Document Cache Indicator

Check this toggle to display in the status bar an indicator allowing to monitor the activity of the document cache of XMLmind XML Editor. Enabling this feature is recommended if you author modular documents and experience some unexplained slowness during validation. More information about this indicator in **Document cache** [89].

Enabling this feature also adds a few menu entries for some of the **XXE** configurations. For example, it adds CSS stylesheet "**Show info about included elements**" to the **View** menu when DITA, DocBook, XHTML documents are opened in **XXE**.

Feature name: `AdvancedTools`.

6.11. Advanced options

6.11.1. Cached data options

Uncheck the checkbox to disable the corresponding cache.

Click the **Clear** button to make the corresponding cache empty. Note that all the following caches are automatically cleared each time you install a new version of **XXE**.

If you are a consultant developing an XXE configuration on the behalf of a group of writers, you'll almost certainly need to disable all the following caches.

Quick Start cache

This cache exists only when XMLmind XML Editor is run as a desktop application. It does not exist when **XXE** is run as an applet or when it has been started using Java™ Web Start.

The lists of configuration objects discovered by XXE during its startup (`.xxe` configuration files, XML catalogs, translations of **XXE** messages, spell-checker dictionaries, all kinds of plug-ins, `customize.xxe_gui` files, etc) are compiled and cached for slightly faster subsequent startups.

Once these lists have been compiled and cached in directory *`XXE_user_preferences_dir`* `[106]/cache/start/`, a configuration file such as `docbook.xxe` is loaded only when this is really needed, that is, when you open a DocBook document. In other words, if you never open a DocBook document (and if you never use **File** → **New**) during your editing session, then the `docbook.xxe` configuration file will not be loaded at all.

Default state: checked.

This cache is automatically cleared when you install add-ons using **Options** → **Install Add-ons** and also when you upgrade XXE. If, for any reason you cannot install an add-on using **Options** → **Install Add-ons** and have to install this add-on manually by unzipping it in one of the two `addon/` directories, do not forget to clear the quick start cache before restarting XXE.

Password cache

XXE allows to open documents stored on WebDAV, FTP, SFTP, etc, servers. Such servers generally require the user to authenticate himself.

In order to do this, XXE prompts the user for his username and password. Once this is done, the credentials are always cached for subsequent use within the same editing session.

Moreover, if the user check the "**Remember these username and password**" checkbox of the authentication dialog box, the credentials are saved to the user preferences file for use in subsequent editing sessions.

The password cache cannot be disabled.

If you typed an incorrect username and/or password, you'll probably want to click the **Clear** button of this **Password cache**. Doing this forces XXE to prompt you again for your credentials.

Document cache

This document cache is needed because some of the components of **XXE** (e.g. the XPath `document()` function in *XMLmind XML Editor - Support of XPath 1.0*) have to quickly access the contents of documents which have not been explicitly opened in the XML editor by the user.

Default state: checked. Default size: 100.

Schema cache

When an instance document conforming to a grammar (DTD, W3C XML Schema or RELAX NG Schema) is opened in **XXE**, the grammar is checked for validity, then the instance document is checked for validity.

Checking a grammar for validity may be a lengthy operation for a medium-size W3C XML schema and for a large DTD such as DocBook. (This implies that this cache is not useful if you exclusively use small or medium size DTDs such as XHTML.)

Therefore, if the cache is enabled, **XXE** checks the grammar the first time it is used by an instance document and then caches it for subsequent uses. The grammar is cached

1. in memory,
2. on disk using a fast loading binary format (DTD, W3C XML Schema, but not RELAX NG Schema).

   The directory used to store the fast loading binary files is *XXE_user_preferences_dir* `[106]/cache/schema/`.

The cache records the location and date of the source files of the grammar. If these source files are removed or modified, the cache will automatically discard the obsolete cached grammar. Of course, this forces XXE to load an up-to-date grammar from the source files.

Note that the existence and date of remote source files (that is, grammar files located on a HTTP or FTP server) are *not* checked by the schema cache.

Default state: checked. Default size: 10.

6.11.2. Proxies options

This preference sheet is absent when XMLmind XML Editor has been started using Java™ Web Start. In such case, proxy servers must be configured using the Java™ Control Panel.

A proxy server is a service which allows clients such Web browsers or XMLmind XML Editor to make indirect network connections to other HTTP, FTP, etc, servers. A common proxy server is a caching Web proxy. This proxy provides a nearby cache of Web pages and files available on remote Web servers, allowing clients to access them quickly and reliably.

That's why, if you have problems listing or downloading available add-ons using **Options** → **Install Add-ons**, may be your organization uses a proxy server to make connections to the Internet and in such case, this proxy server needs to be declared using this preference sheet.

Direct connection to the Internet

Do not use the services of proxy servers, if any. Make direct connections to the Internet.

Default: not checked.

Use system settings

Use what has been specified in the ``control panel'' of your desktop/operating system. For example, on Windows, you'll find these settings in **Control Panel** > **Internet Options** > **Connections** tab > **LAN Settings**.

Default: checked.

Manual proxy configuration

Specify which proxy server to use for which network protocol. Please ask your network administrator to help you fill this form.

Default: not checked.

HTTP proxy
Specifies the host name of the proxy server to use when HTTP connections are made.

Default: none.

(HTTP proxy) **Port**
Specifies the port number of the proxy server to use when HTTP connections are made.

Default: 80.

HTTPS proxy
Specifies the host name of the proxy server to use when HTTPS connections are made.

Default: none.

(HTTPS proxy) **Port**
Specifies the port number of the proxy server to use when HTTPS connections are made.

Default: 443.

FTP proxy
Specifies the host name of the proxy server to use when FTP connections are made.

Default: none.

(FTP proxy) **Port**
Specifies the port number of the proxy server to use when FTP connections are made.

Default: 80.

SOCKS proxy
Specifies the host name of the SOCKS V4 or V5 proxy server to use when network connections of any sort are made.

Default: none.

(SOCKS proxy) **Port**
Specifies the port number of the SOCKS V4 or V5 proxy server to use when network connections of any sort are made.

Default: 1080.

Authenticate SOCKS user
Check this toggle if the SOCKS proxy server of your organization requires you to authenticate yourself before using its services.

Default: not checked.

(SOCKS) **User name**
Your name as a user of the SOCKS proxy server.

Default: the login name of the user.

(SOCKS) **Password**

Your password as a user of the SOCKS proxy server.

Default: the empty string.

No proxy for

When XMLmind XML Editor runs on these hosts, do not the services of proxy servers, if any. Make direct connections to the Internet.

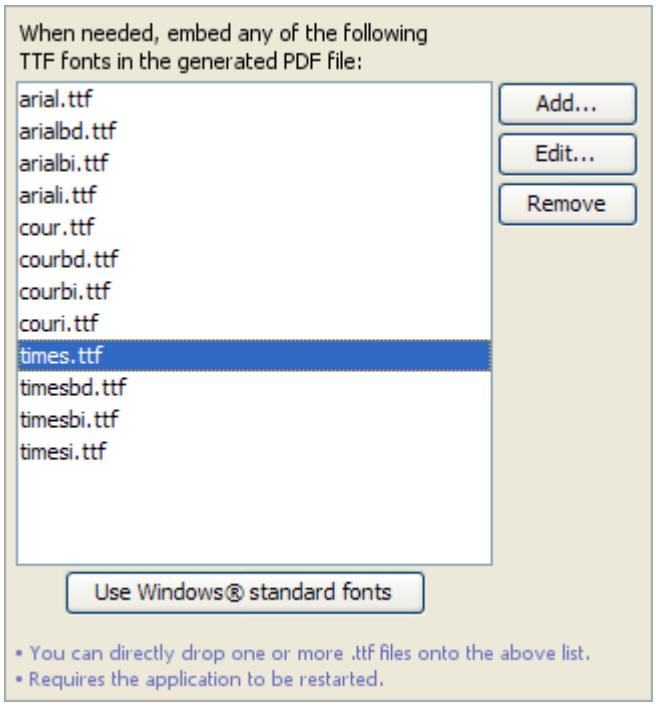
Default: "`localhost 127.0.0.1`".

6.12. Add-on options

6.12.1. Apache FOP options

The "**FOP**" preferences sheet is available only when add-on "**Apache FOP 1.x XSL-FO processor plug-in**" has been installed.

*Figure 9.9. The "**FOP**" preferences sheet*



By default, only the 14 built-in fonts: Times, Helvetica, Courier, Symbol and ZapfDingbats are used in the generated PDF. The above preferences sheet allows to specify which custom TrueType (`.ttf`) fonts are to be *embedded* in the generated PDF.

This facility is useful in the following two cases:

- The 14 PDF standard fonts (Helvetica, Times, Courier, etc), which are used by default by FOP, have glyphs only for the Western languages. If, for example, you convert a DocBook document written in Russian to PDF, the generated PDF will mainly contain the '#' placeholder character. Fortunately, widely available TTF fonts such as Microsoft® Arial, Times New Roman and Courier New or the DejaVu fonts have glyphs for almost all the languages of the world.

- Use fonts nicer than the 14 PDF standard fonts.

Procedure 9.1. How to use Times New Roman, Arial and Courier New instead of Times, Helvetica, Courier

1. Click **Use Windows® standard fonts**.

   Note that the **Use Windows® standard fonts** button is grayed if the Arial font is not found in the standard fonts folder of your system.

2. Click **OK**.

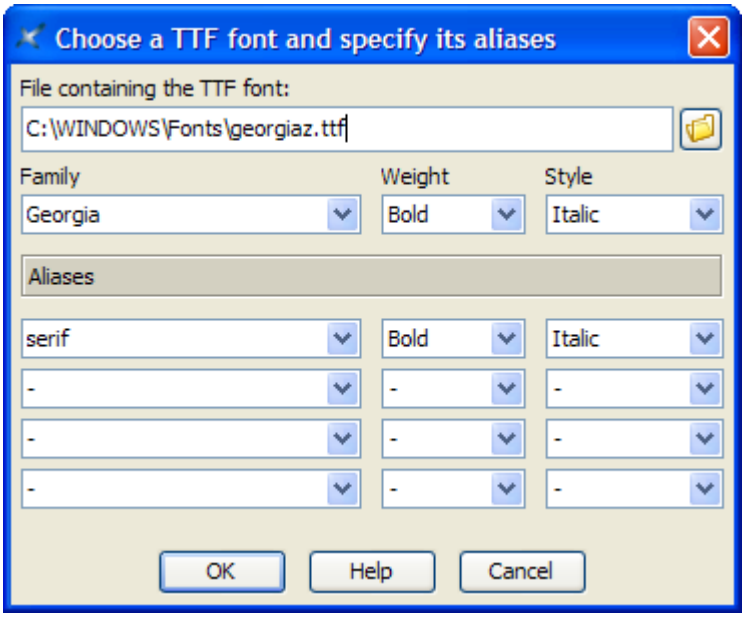
3. Restart XMLmind XML Editor.

Procedure 9.2. How to choose specific fonts (for example, you want to replace Times fonts by Georgia fonts)

1. Click **Add**.

   This displays the following dialog box:

   *Figure 9.10. The "**Choose a TTF font and specify its aliases**" dialog box*



   a. Choose the `.ttf` file containing font "Georgia".

   On Windows, for permissions reasons, there is no way to pick a font file from the "`C:\Windows\Fonts\`" folder using the standard file chooser. Therefore the only way to register with Apache FOP a font found in "`C:\Windows\Fonts\`" is to drag its file from the Windows file manager and to drop it onto the list found in the "**FOP**" preferences sheet (see above figure [131]).

   However, after you do this, do not forget to select each entry added by the drop action and then click **Edit** to possible change or complement what's have been automatically specified there.

b. Specify the following alias: serif.

c. Click **OK**.

2. Click **Add**.

a. Choose the `.ttf` file containing font "Georgia Bold".

b. Specify the following alias: serif, Bold.

c. Click **OK**.

3. Click **Add**.

a. Choose the `.ttf` file containing font "Georgia Italic".

b. Specify the following alias: serif, Italic.

c. Click **OK**.

4. Click **Add**.

a. Choose the `.ttf` file containing font "Georgia Bold Italic".

b. Specify the following alias: serif, Bold, Italic.

c. Click **OK**.

5. Click **OK**.

Doing this automatically creates a standard FOP configuration file in *`XXE_user_preferences_dir`*`/fop/fop.conf`. User preferences directory *`XXE_user_preferences_dir`* is documented in Section 6, “The "**Preferences**" dialog box” [106].

6. It is recommended to repeat the above steps in order to specify fonts replacing Helvetica, that is, fonts having a sans-serif alias and fonts replacing Courier, that is fonts having a monospace alias.

7. Restart XMLmind XML Editor.

Using in XXE an existing FOP configuration file

The location of an existing FOP configuration file may be specified to **XXE** by the means of Java™ system property or environment variable `XXE_FOP_CONFIG`. The value of this variable is an URL or an absolute file path. Examples: `-DXXE_FOP_CONFIG=http://localhost/~john/fop/fop.conf` (Java™ system property), `set XXE_FOP_CONFIG=C:\Users\john\misc\fop\fop.conf` (Windows environment variable).

6.12.2. RenderX XEP options

The "**XEP**" preferences sheet is available only when add-on "**RenderX XEP XSL-FO processor plug-in**" has been installed.

XEP preferences are almost identical to FOP preferences [130]. The 3 differences are:

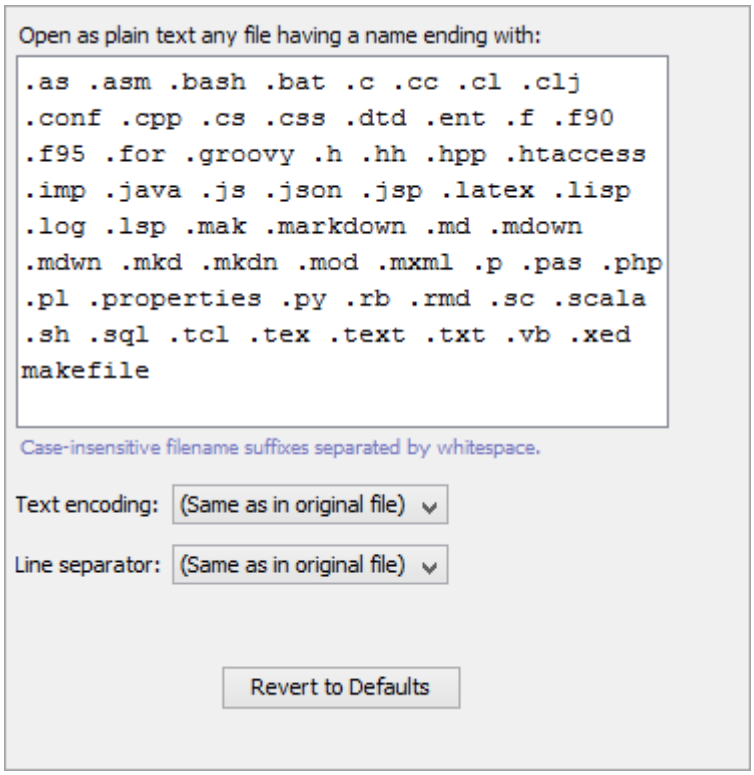
- It is possible to install both the "**RenderX XEP XSL-FO processor plug-in**" add-on and the "**Apache FOP 1.x XSL-FO processor plug-in**" add-on. When this is the case, by default, XEP supersedes FOP. The "**Use Apache FOP rather RenderX XEP**" check-box, found only in the XEP preferences sheet, lets you change this.
- Some fonts have licensing restrictions that forbid embedding them in a PDF file. RenderX XEP enforces these licensing restrictions, not Apache FOP. XMLmind XML Editor has currently no way to detect these licensing restrictions, therefore you may follow the above procedure and end up with glyphs still missing in the generated PDF.
- The standard XEP configuration file is created in *`XXE_user_preferences_dir`*`/xep/xep.conf`.

Using in XXE an existing XEP configuration file

The location of an existing FOP configuration file may be specified to **XXE** by the means of Java™ system property or environment variable `XXE_XEP_CONFIG`. The value of this variable is an URL or an absolute file path. Examples: `-DXXE_XEP_CONFIG=http://localhost/~john/xep/xep.conf` (Java™ system property), `set XXE_XEP_CONFIG=C:\Users\john\misc\xep\xep.conf` (Windows environment variable).

6.12.3. Text format options

*Figure 9.11. The "**Text format**" preferences sheet*



Open as plain text

Specifies the suffixes of the files that should be opened by **XXE** as plain text files (that is, *not XML files*). A file suffix is case insensitive. If a suffix does not start with a dot (e.g. "`makefile`") then this “suffix” is considered to a file base name and not a file suffix.

Text encoding

Specifies the text encoding (e.g. **Windows-1252** on Windows for a Western PC) to use when saving a plain text file to disk.

Notes:

- Choosing "**(Same as in original file)**" implies also saving the original Byte Order Mark (**BOM**) if a one was there when opening the text file in **XXE**.
- Special values **UTF-16BE_BOM**, **UTF-16LE_BOM** and **UTF-8_BOM** mean: save the text file to disk using respectively the `UTF-16BE`, `UTF-16LE` and `UTF-8` encodings and also add the corresponding **BOM** when doing so.

This preference is ignored when an *encoding directive* is found at the beginning of the text file to be saved.

For example, this CSS stylesheet will always be saved using encoding "`ISO-8859-1`":

```
@charset "iso-8859-1";

body {
    font-family: sans-serif;
}
```

Other example, this Markdown file will always be saved using encoding "`ISO-8859-1`":

```
<!-- -*- coding: iso-8859-1 -*- -->

Heading
=======

## Sub-heading

Paragraphs are separated
by a blank line.
```

(The above example works because **XXE** understands the GNU Emacs file variable called "`coding`".)

Line separator

Specifies the line separator (e.g. **CR+LF**, that is, "`\r\n`" , on Windows) to use when saving a plain text file to disk.

6.12.4. JSON format options

Edit JSON as **Text** or **XML**

If **Text** is selected, the **JSON** format add-on is made inactive and **JSON** files are treated by **XXE** as plain text files.

If **XML** is selected, the **JSON** format add-on is actived and **JSON** files are treated by **XXE** as if they were XML files.

The XML representation of **JSON** used by the **JSON** format add-on is documented in "*JSONx, an XML Encoding for JSON*".

JSON file indentation

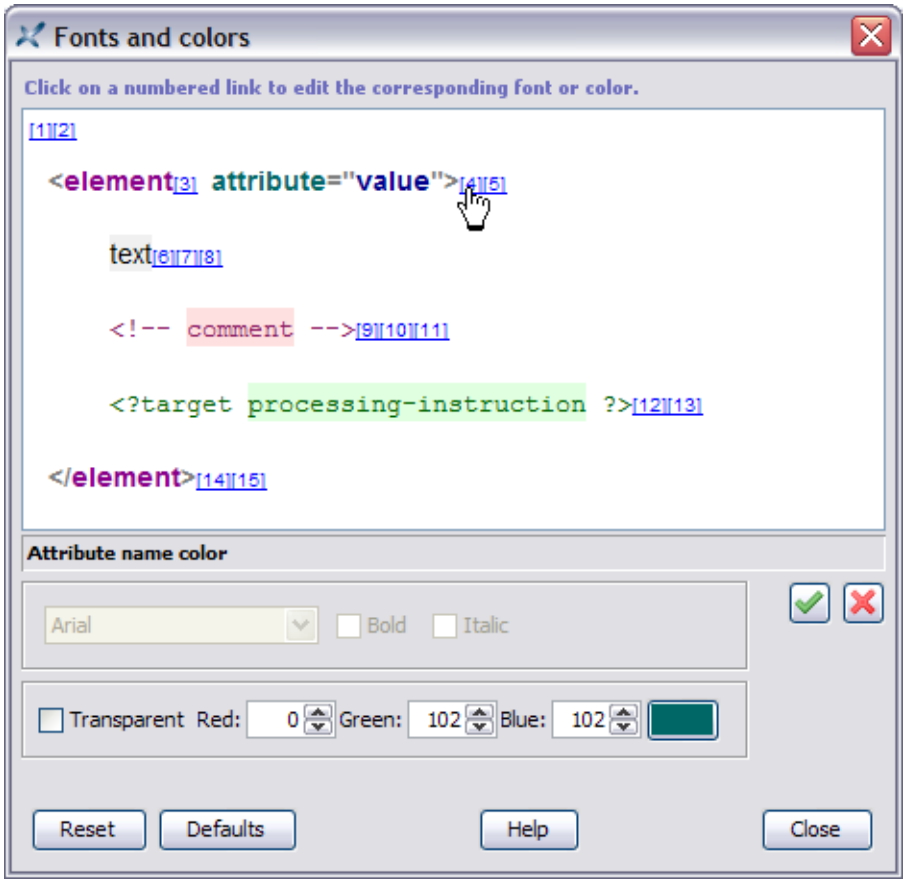
Specifies the line indentation used when saving **JSON** files. The recommended indentation for **JSON** files is 4 space characters. Value 0 means no indentation at all, resulting in a compact **JSON** file containing a single line of text.

- A **JSON** file has a "`.json`" filename suffix and is encoded in UTF-8 or UTF-16.
- The **JSON** format add-on always uses the UTF-8 encoding when saving **JSON** files to disk.

7. The "Tree view fonts and colors" dialog box

This dialog box is displayed by clicking "**Fonts and colors**" [118] in **Options** → **Preferences**, **View** section, "**Tree view settings**" subsection. It allows the specify the fonts and colors used to render a tree view.

Note that the *sizes* of the fonts used to render a tree view are not specified in this dialog box. Instead, they are specified by "**Base font size**" [118].



In order to specify a font or a color,

1. Click on a numbered link in the top pane.

In the case of the above screenshot, the user has clicked on **[4]** which corresponds to modifying "**Attribute name color**".

2. Doing this activates a font chooser or a color chooser in the bottom pane, depending on the kind of property being modified.

   In the case of the above screenshot, "**Attribute name color**" is a color, therefore the color chooser is activated and the font chooser is grayed out.

   Note that some colors are allowed to be transparent (background and border colors) and some are not (text colors). For example, if you specify **Transparent** for "**Attribute name color**", an error will be reported.

3. Specify a font or a color.

4. Click ✔ (**Apply**).

   The top, preview, pane is automatically updated to reflect your choice.

Other buttons:

Reset
: Discard all the changes. Revert to the initial fonts and colors.

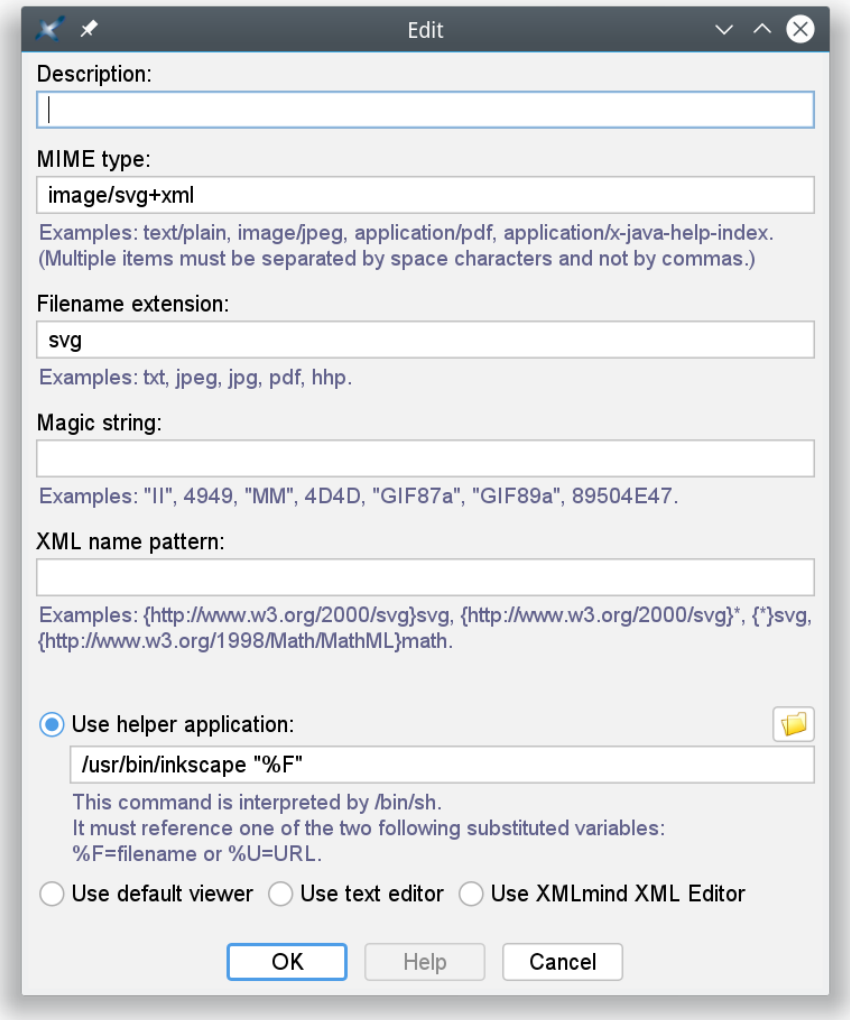
Defaults
: Discard all the changes. Revert to the default (that is, “factory”) fonts and colors.

Some of these settings also apply to the "XML source" view [34]

- Most foreground colors are also used by the "**XML source**" view.
- The font family specified in item [9] —comment, processing-instruction and verbatim text font— is used as the (only) font family of the "**XML source**" view.
- The font styles (bold, italic) specified in item [14] —markup font— are used as the font styles of the markup in the "**XML source**" view.

8. The "Helper Application Editor" dialog box

This dialog box allows to view or modify a file type listed in the "**Helper applications**" preferences sheet [121].



A file type is specified by *at least one* of the following characteristics:

MIME type

The official (or just well-known) formal name of the file type. Generally returned by Web servers to their client Web browsers. Non-registered MIME types typically start with string "`application/x-`".

A MIME type may end with a wildcard. Example: "`image/*`" matches "`image/gif`", "`image/jpeg`", etc.

Examples: `text/plain`, `image/jpeg`, `application/excel`, `application/x-java-help-index`.

Filename extension

If the filename or URL of a file ends with specified "`.extension`", then this file is detected as having this file type.

An extension may or may not start with a dot. This is unimportant because, in all cases, a leading dot would be automatically stripped.

Examples: `txt`, `jpeg`, `jpg`, `xls`.

Magic string

For some file formats, the first bytes of a file are always the same and therefore, can be considered as being the *signature* of this file type.

If a file starts with specified first bytes, then this file is detected as having this file type. This type of detection is supposed to work like magic, hence the name: ``magic string".

A magic string may be specified by a the hexadecimal representation of a sequence of bytes (example, one of the two TIFF magic strings: `4949`) or by a *quoted* sequence of ASCII characters (same example, one of the two TIFF magic strings: `"II"`).

Examples: TIFF: `"II"` or `4949`, `"MM"`or `4D4D`; GIF: `"GIF87a"`, `"GIF89a"`; PNG: `89504E47`; PDF: `"%PDF-"`.

XML name pattern

If the root element of an XML file has a name which matches specified pattern, then this XML file is detected as having this file type.

An XML name pattern follows this syntax:

```
( '{' namespace_URI? '}' )? local_part
```

One of *`local_part`* or *`namespace_URI`* may be equal to wildcard `"*"`

Examples: `{*}svg`, `{http://www.w3.org/1998/Math/MathML}:math`.

Each file type has an associated helper application. This helper application is assumed to be able to open files detected as having this type. A helper application may be a viewer or an editor.

Description

Description of the file type. Not mandatory, just recommended. This text is displayed in the **File types** list of the "**Helper applications**" preferences sheet [121].

MIME type

One or more MIME types (see definition [137] above) separated by spaces.

Filename extension

One or more extensions (see definition [137] above) separated by spaces.

Magic string

One or more magic strings (see definition [138] above) separated by spaces.

XML name pattern

One or more XML name patterns (see definition [138] above) separated by spaces.

Use helper application

This field must contain a command line interpreted by the native shell of the platform: `cmd.exe` on Windows and `/bin/sh` on the Mac and on Unix.

This command line must reference one of these two substituted variables: `%U` and `%F`. In principle, `%U` is replaced by the URL of the file to be opened by the helper application and `%F` is replaced by a filename. In practice, `%U` is just a *hint* meaning: the helper application can open URLs as well as filenames.

The **Choose Helper Application** button displays the standard file chooser in order to specify an application (e.g. a `.exe` or a `.bat` file on Windows). String `" "%F""` is automatically appended to the chosen application.

Helper applications on the Mac

When an *application*, that is, a folder having a name ending with hidden suffix "`.app`", containing a package bundle, has been selected by the user, the **Choose Helper Application** button automatically prepends "`open -W -n -a `" to chosen `.app` folder. Example: "`open -W -n -a /Applications/Inkscape "%F"`".

Options "`-W -n`" mean: start a new instance of the application and wait until this instance has exited. These options are required when the helper application is used to edit the content of an element, the content of an attribute or the whole document.

Use default viewer

Files having specified type are to be opened using the default viewer. The default viewer is specified in the "**Helper applications**" section of the "**Preferences**" dialog box [121].

Use text editor

Files having specified type are to be opened using the “default text editor”.

Which text editor?

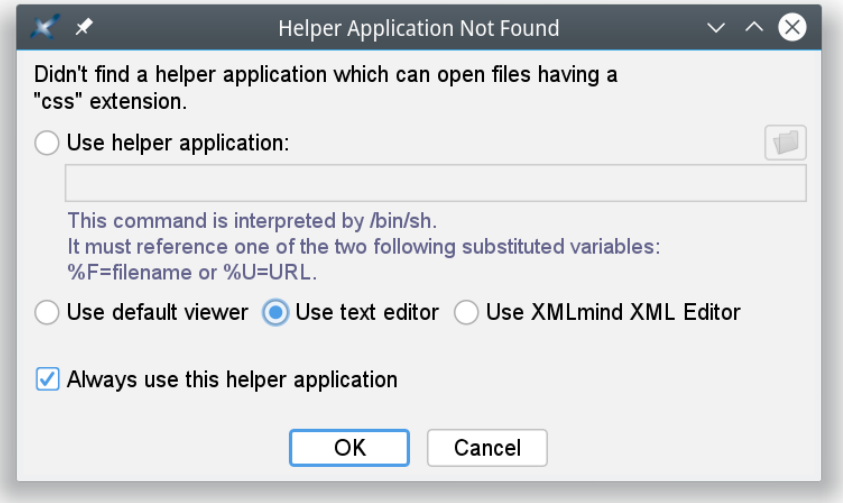
This text editor to be used is *the helper application associated to the "`text/plain`" MIME type*. If such association has not been defined, the standard text editor of the platform is automatically used (e.g. on Windows, it's `notepad`).

Use XMLmind XML Editor

Use **XXE** as an “internal” helper application to open files having specified type. For example, associating **XXE** to `.mml` files lets you edit the MathML documents referenced in a DocBook document.

The "Helper Application Not Found" dialog box

When you attempt to open a non-XML file using some of the tools[6] of **XXE** for which no helper application has been associated, the following dialog box is displayed.



If "**Always use this helper application**" is checked, which is often the case by default, this dialog box automatically associates a file type to specified helper application. For example, in the case of the above screenshot, files having a `.css` extension are associated to the default text editor.

9. The "Save Document As Template" dialog box

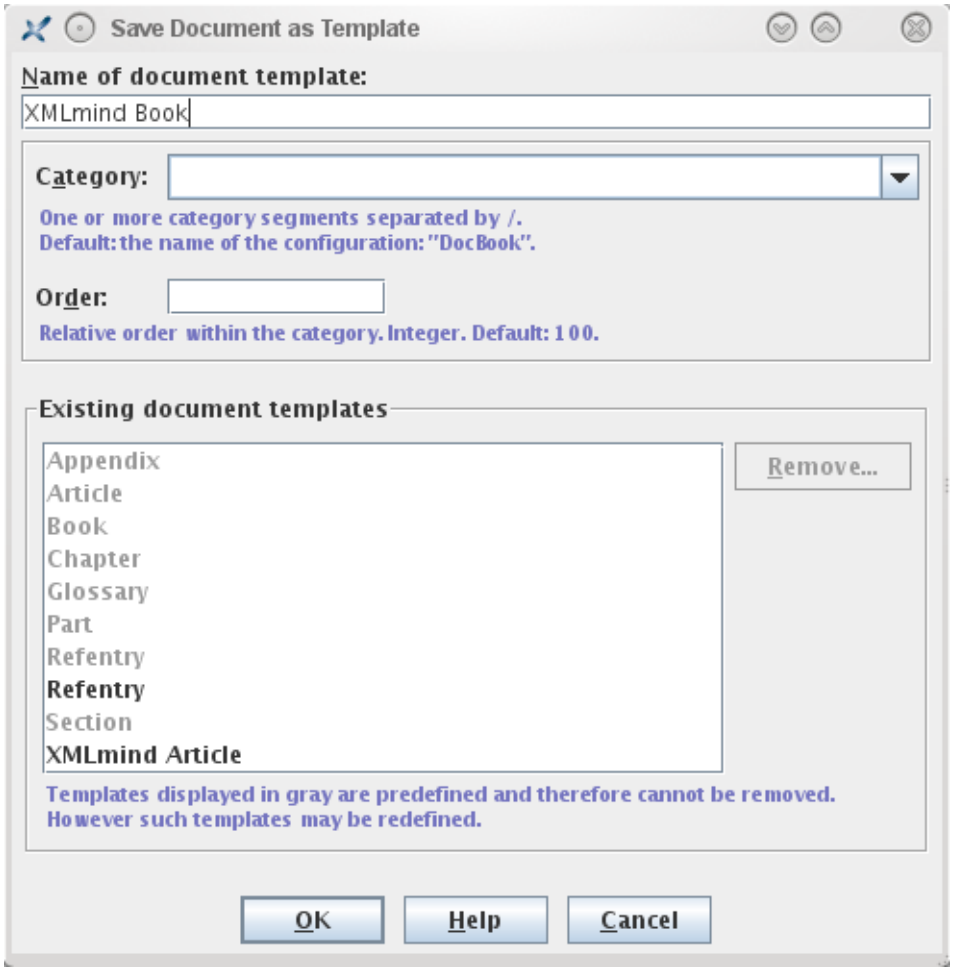
This dialog box is displayed by menu item **Options** → **Customize Configuration** → **Save Document as Template** [46]. Basically, it allows the user to specify a name for the newly created document template.

The name of a document template may contain any character, including spaces.

The name of a document template may be identical to the name of one of the existing document templates. This allows to replace an existing document template by a new one.

[6] For example:

- the **Open** entry of the contextual menu of the "**Browse Files**" tool [72],
- any of the entries of the **Tools** → **Helper Application** menu [42].



In the above screen shot, we can see that:

- The user is creating a new document template called "`XMLmind Book`".
- Previously, the user has replaced predefined template called "`Refentry`" by a "`Refentry`" document template of its own.
- Previously, the user has created another document template called "`XMLmind article`".

9.1. Organizing the content of the File New dialog box

Optional document template properties **Category** and **Order** allow to better organize the content of the **File** → **New** dialog box.

Category

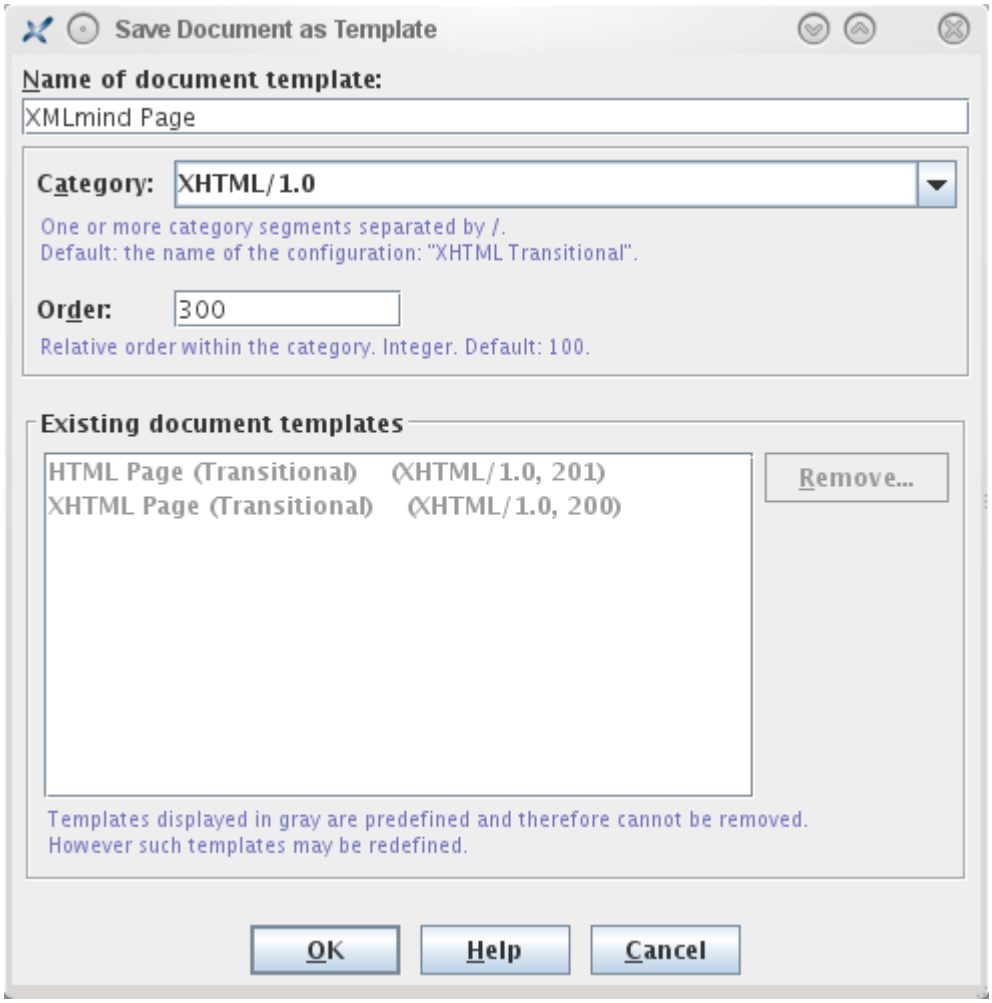
Specifies the category of the new document template. A category consists in one or more segments separated by character `'/'`. By default, the category of a document template is the name of the configuration in which this template has been specified.

Order

Specifies the relative order of the new document template within its category. Default value is 100.

For example, stock category "`XHTML/1.0`" includes all the templates specified in stock configurations "`XHTML Strict`" and "`XHTML Transitional`". These stock document templates are "`XHTML Page`

`(Strict)`", "HTML `Page (Strict)`", "`XHTML Page (Transitional)`" and "`HTML Page (Transitional)`". Let's suppose a user wants to create a new document template template conforming to the Transitional DTD.



In the above screen shot, we can see that:

- The user is creating a new document template called "`XMLmind Page`".
- The user wants it to be found in the same category ("`XHTML/1.0`") as the stock XHTML 1.0 templates.
- The user wants it to be found *after* the "`HTML Page (Transitional)`" stock template (its order, 300, is greater than 201).
- Note that the above dialog box does not list all the document templates belong to category "`XHTML/1.0`", but instead it lists all the document templates specified the the configuration called "`XHTML Transitional`".

9.2. Removing a previously defined document template

This dialog box also allows to remove user-defined document templates. In order to do this, simply click on a template in the "**Existing document templates**" list and then click on the **Remove** button.

Note that this operation takes immediate effect: clicking on the **Cancel** button of the "**Save Document As Template**" dialog box will not cancel the deletion of the template[7].

You can only remove the document templates you have created. You cannot remove predefined (grayed) document templates.

In the above screen shot, notice how removing user-defined template called "Refentry" would restore the predefined template having the same name.

10. The "Save Selected Element as Template" dialog box

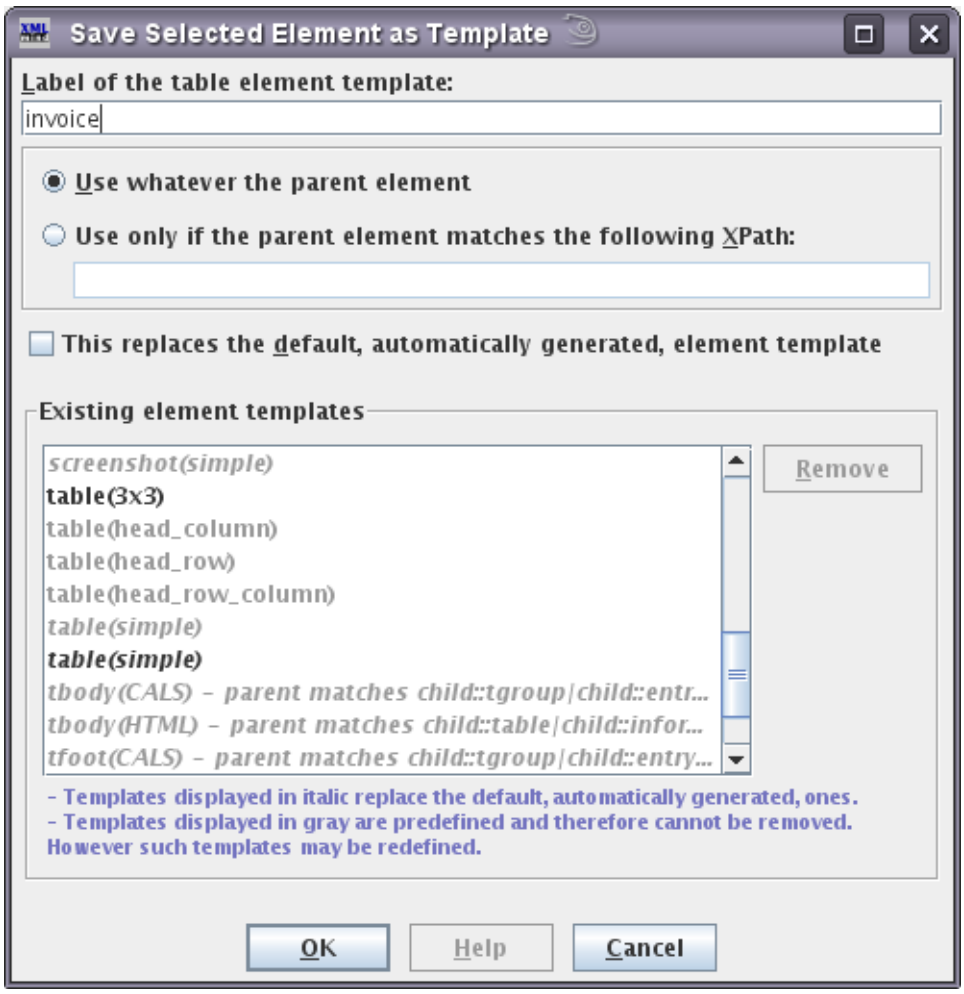
This dialog box is displayed by menu item **Options**+**Customize Configuration** → **Save Selected Element as Template** [47]. Basically, it allows the user to specify a *label* for the newly created element template.

Giving a label to an element template is needed to support different templates for the same element type. For example, in a DocBook document, select a `table` element and use **Options**+**Customize Configuration** → **Save Selected Element as Template** to label it as "L1". Then select another `table` element and use the same menu item to label it as "L2". This gives you two `table` element templates, one called "L1" and the other called "L2".

The label of an element template must be a valid `NMTOKEN`. It cannot contain whitespace.

The element name and label of a template may be identical to the element name and label of one of the existing templates. This allows to replace an existing template by a new one.

[7] The **Cancel** button of the "**Save Document As Template**" dialog box just cancels the action of creating a new template.



In the above screen shot, we can see that:

- The user is creating a new `table` element template called "invoice".

  After creating this template, it will be displayed in the "**Existing element templates**" list as "table(invoice)", "table" being the element name of the template and "invoice" being its label.

- Previously, the user has created another `table` element template labelled "3x3".

- Previously, the user has replaced predefined "table(simple)" template by a "table(simple)" template of its own.

10.1. Advanced options

Use only if the parent element matches the following XPath

Normally an element template is suggested by the **Edit** tool whatever the parent element where this template is to be inserted. But in some cases, generally with W3C XML schemas and with RELAX NG schemas and not with DTDs, you'll want the template to be usable only in certain contexts.

Example: your schema defines two `title` elements: a *job title* which is a possible child element of `person` and `author` elements and the optional title (i.e. a *caption*) of `table` and `figure` elements.

You intend to define several job title templates. Example: element template "title(arch)" is defined as `<title>Software Architect</title>`.

Of course, you don't want to see "title(arch)" listed by the **Edit** tool when you are about to give a title to a `table`.

In such case:

1. Click on the "**Use only if the parent element matches the following XPath**" radio button.
2. Type XPath expression "`person|author`" in the text field below the radio button.

This means: suggest "title(arch)" but only when the template is to be inserted in a `person` or an `author` element.

Note that a very small subset of XPath is supported here:

XPath 1.0 subset

The XPath 1.0 subset is the one defined in "XML Schema Part 1: Structures, Identity-constraint Definitions", except that absolute XPaths (`/foo/bar`, `//bar`, etc) are also supported.

```
XPath      ::=    Path ( '|' Path )*
Path       ::=    ('/'|'//')? ( Step ('/'|'//') )* ( Step | '@' NameTest )
Step       ::=    '.' | NameTest
NameTest   ::=    QName | '*' | NCName ':' '*'
```

Both abbreviated syntax and non-abbreviated syntax are supported.

This replaces the default, automatically generated, element template

This option is best explained by an example.

Out of the box, if you insert a `table` element in a DocBook document, in fact, you insert predefined `table` template called "table(simple)".

The "table(simple)" template has two rows and two columns which makes it much more useful that the default, automatically generated, `table` template which just contains a single cell[8].

Note that when a *single* named template such as "table(simple)" replaces the default, automatically generated, element template, the **Edit** tool does not display its label. For example, in the case of the DocBook `table`, the **Edit** tool displays "table"[9] and not "table(simple)".

Now, what if you prefer to use a `table` having 3 rows and 3 columns plus a `thead` header as your default table?

1. Create the table by inserting the rows, cells and header you want.
2. Select this table.
3. Use **Options** → **Customize Configuration** → **Save Selected Element as Template**.

[8]The default, automatically generated, element template is the simplest, valid, element allowed by the schema.

Most of the time this heuristic gives good results. However in some cases, the default, automatically generated, element template, is, well, too simple to be useful.

[9]There is no ambiguity and this is shorter to type.

4. The "**Existing element templates**" list shows you that a predefined template already replaces the default, automatically generated, one. You can see this because "table(simple)" is displayed in italic.

   You need to label your own template like the existing one in order to replace it. Therefore specify "simple" in the **Label of the element template** field.

5. Click on check box "**This replaces the default, automatically generated, element template**".

10.2. Removing a previously defined element template

This dialog box also allows to remove user-defined element templates. In order to do this, simply click on a template in the "**Existing element templates**" list and then click on the **Remove** button.

Note that this operation takes immediate effect: clicking on the **Cancel** button of the "**Save Selected Element As Template**" dialog box will not cancel the deletion of the template[10].

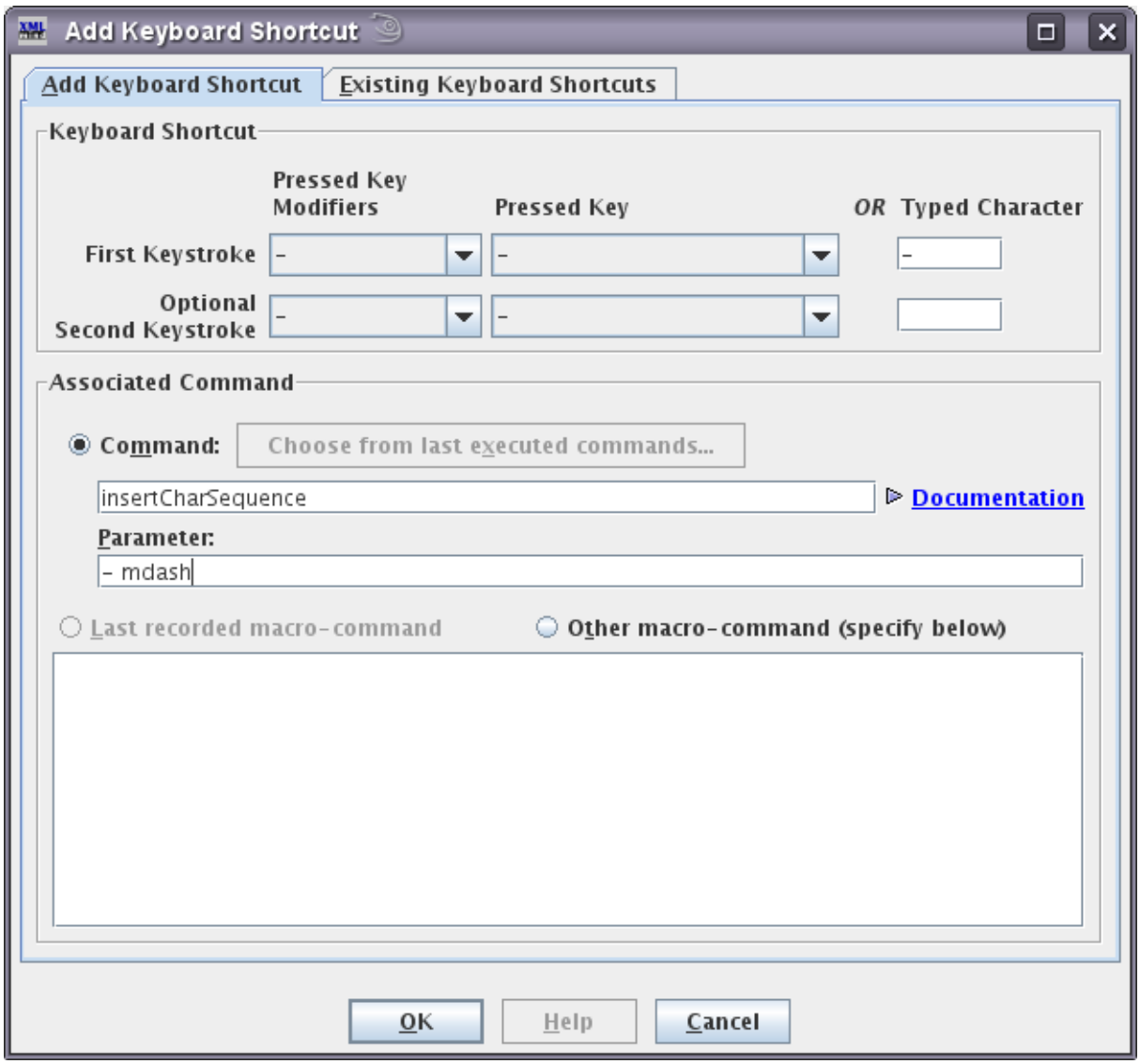
You can only remove the element templates you have created. You cannot remove predefined (grayed) element templates.

In the above screen shot, notice how removing user-defined "table(simple)" template would restore the predefined template having the same element name and label.

11. The "Add Keyboard Shortcut" dialog box

This dialog box is displayed by menu item **Options** → **Customize Configuration** → **Add Keyboard Shortcut** [47]. It allows the user to specify a keyboard shortcut that will invoke a predefined command, last recorded macro-command or a user-defined macro-command.

[10]The **Cancel** button of the "**Save Selected Element As Template**" dialog box just cancels the action of creating a new template.



In the above screen shot, the user adds a shortcut for command "`insertCharSequence`" with parameter "`- mdash`". This command will be invoked after character "`-`" (dash) is typed.

Procedure 9.3. Adding a keyboard shortcut

1. Specify a keystroke. A keystroke is either:

   - A *typed character* specified the **Typed Character** field.

     A typed character is any character you can type using any combination/sequence of pressed keys. A typed character often does not correspond to a key of your keyboard. For example, on an US keyboard, there is no **$** key. You need to press on key **4** and on key **Shift** at the same time in order to type the '$' character.

   - OR a *pressed key* specified using the **Pressed Key** combo box.

A pressed key must correspond to a key actually found on your keyboard. For example, do not choose key **$** on an US keyboard because there is no such key. On the other hand, you may choose it on a French keyboard which has a **$** key[11].

You'll often want to also specify the modifier keys (**Ctrl**, **Shift**, etc) which must be pressed at the same time as the main key specified in the **Pressed Key** combo box. Any combination of modifier keys may be chosen using the **Pressed Key Modifiers** combo box.

2. Optionally specify a second keystroke. In such case, the command will be invoked by the sequence: first keystroke immediately followed by second keystroke.

3. Specify the command invoked by your custom shortcut. There are three possible options for this:

   Command

   The command invoked by your custom keyboard shortcut is a predefined command (documented in Chapter 6, *Commands written in the Java™ programming language* in *XMLmind XML Editor - Commands*).

   You may choose one of the last executed commands by using the list displayed by the "**Choose from last executed command**" button.

   Last recorded macro

   The command invoked by your custom keyboard shortcut is the last recorded macro-command (see **Tools** → **Record Macro** [44]).

   Other macro-command

   The command invoked by your custom keyboard shortcut is a custom macro-command that you'll type in the text area below the "**Other macro-command**" radio button.

   It is recommended to restrict yourself to specifying one of the last executed commands or last recorded macro. Attempting to use the other options is harmless, but unless you know what you are doing (for example, because you have posted a support request to the `<xmleditor-support@xmlmind.com>` public mailing list), the commands specified the hard way will probably not work.

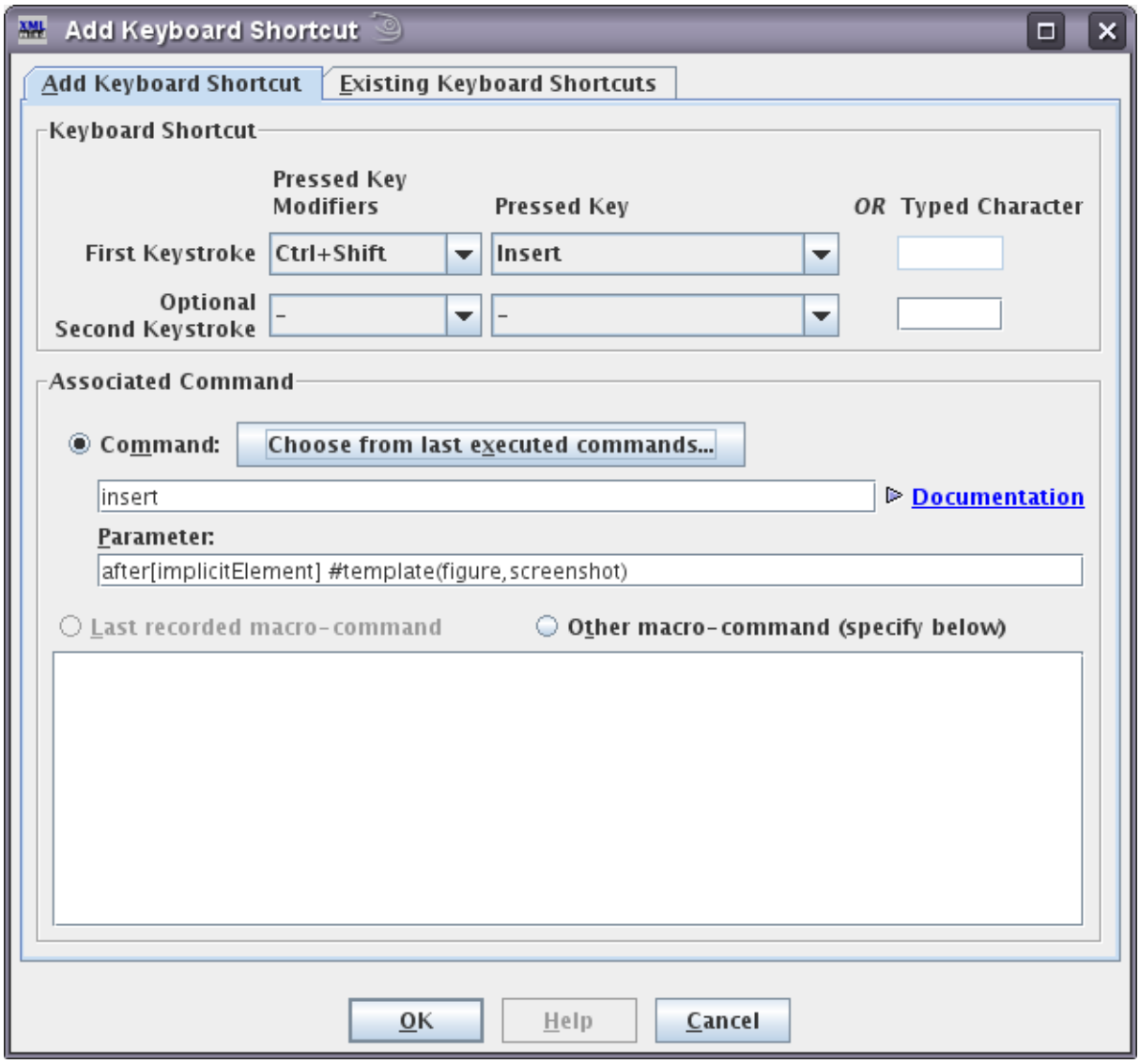
   However, if you decide to follow the hard way, you must know that:

   - The **Command** field, allowing to specify the name of a command, supports auto-completion.
   - Clicking on the **Documentation** link will open in the Web browser the reference manual in *XMLmind XML Editor - Commands* describing the command of interest[12]. This is almost always needed in order to specify in the **Parameter** field a valid parameter for the command.
   - If you type a custom macro-command in the text area, do not bother declaring namespace prefixes. Consider that "`http://www.xmlmind.com/xmleditor/schema/configuration`" is declared as being the default namespace, that "`cfg`" is declared as a prefix for namespace "`http://www.xmlmind.com/xmleditor/schema/configuration`" and that all namespace prefixes declared in the document being edited (see **Tools** → **Declare Namespace** [38]) are in scope also here.

4. Click **OK**.

[11] Suprising, isn't it?
[12] This requires an access to the Internet if XMLmind XML Editor is deployed using Java™ Web Start.

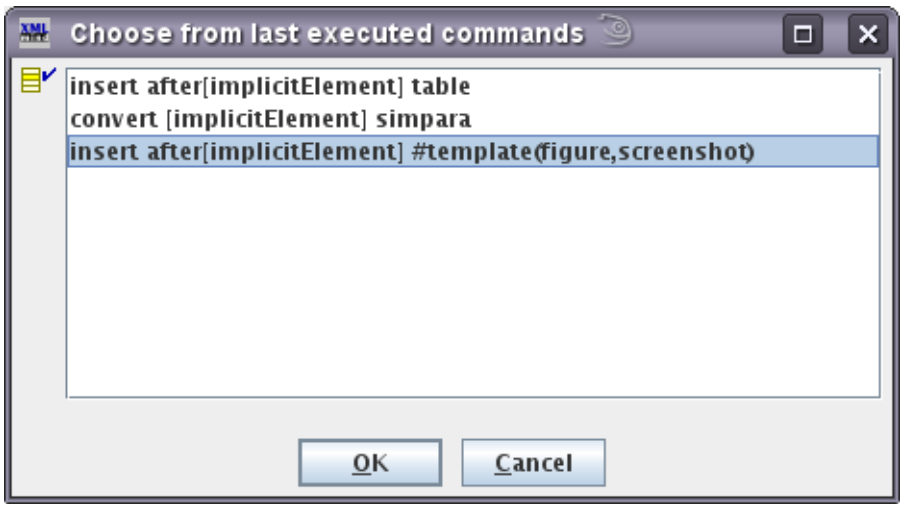
Other example:



In the above screen shot, the user adds a shortcut for command "`insert after[implicitElement] #template(figure,screenshot)`". This command will be invoked when the Insert key will be pressed at the same time as the Control and Shift keys.

Previously, by using the **Edit** tool, the user has inserted in the document being edited a `figure` element containing a `screenshot` child element[13]. Having done this now allows her/him to pick this command from the list displayed by the "**Choose from last executed command**" button, instead of typing the command name and its parameter.

[13]She/he picked `figure(screenshot)` from the list of elements suggested by the **Edit** tool.



Many keystrokes may not be used to define custom keyboard shortcuts. Clicking on the "**Existing Keyboard Shortcuts**" tab will give you an idea of this fact.

Examples:

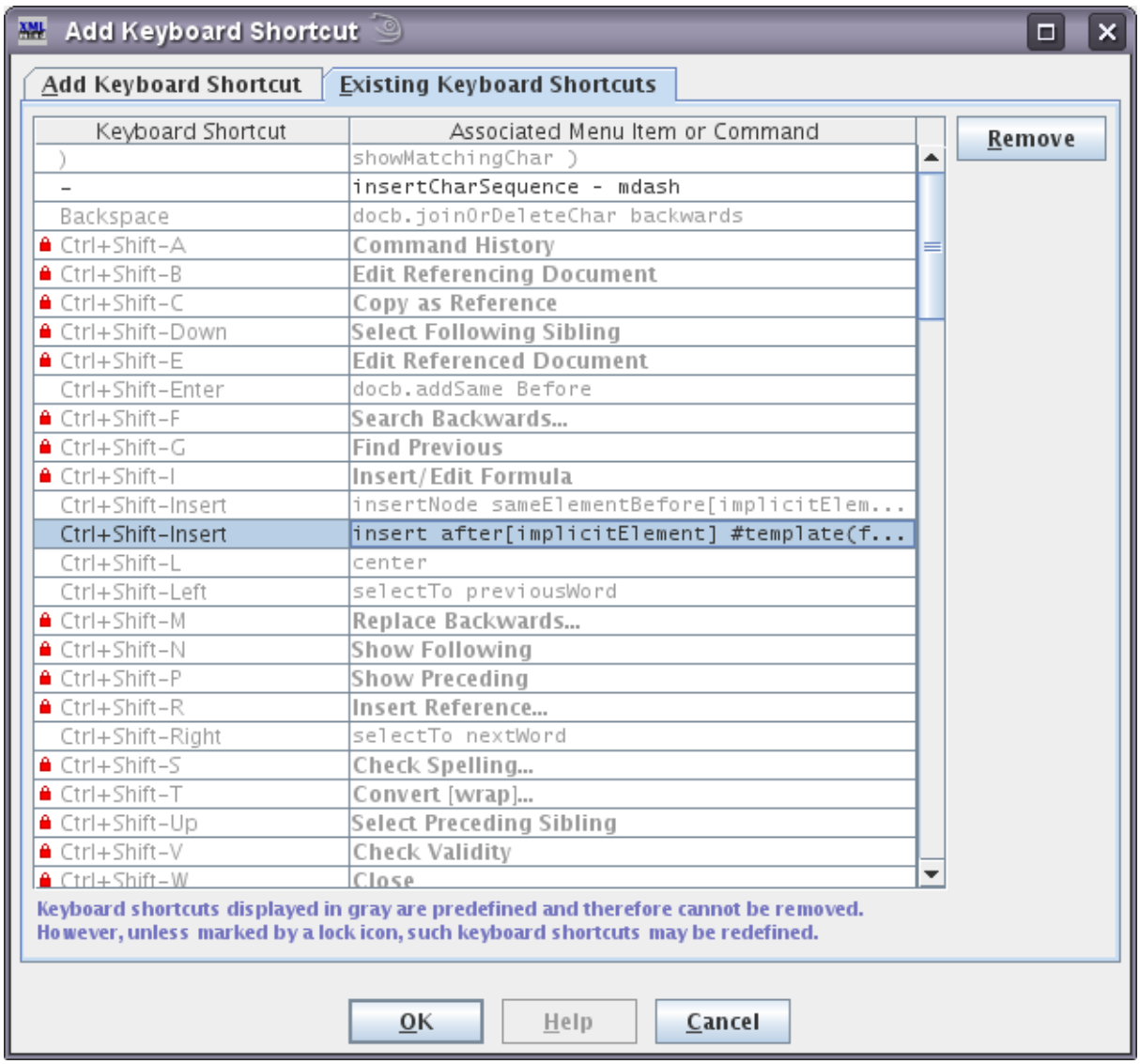
- You may not use **Ctrl**+**O** because this keystroke is reserved for menu item **File** → **Open**.
- You may not use **Ctrl**+**I** because this keystroke is reserved for menu item **Edit** → **Insert**.
- You cannot use single keystroke **Esc** because this would prevent existing keyboard shortcuts **Esc** g (bound to menu item **Select** → **Find Element**), **Esc** x (bound to menu item **Tools** → **Execute**) , etc, from working[14].
- You cannot use keystroke **Home**, for example followed by a second **Home** keystroke, because existing keyboard shortcut **Home** (bound to command "`moveDotTo lineBegin`") would prevent your custom shortcut from working.

Moreover certain keystrokes are reserved by Java™ (e.g. Tab, **Alt**) or by the operating system (e.g. there are loads of these on the Mac). Attempting to use these keystrokes for your custom keyboard shortcuts will almost certainly not work.

11.1. Removing a previously defined keyboard shortcut

This dialog box also allows to remove user-defined keyboard shortcuts.

[14] Using **Esc** followed by a second, carefully chosen, keystroke should work fine.



In the above screen shot, notice how removing user-defined keyboard shortcut **Ctrl**+**Shift**+**Insert** (invoking command "`insert after[implicitElement] #template(figure,screenshot)`") would restore the predefined keyboard shortcut (invoking command "`insertNode sameElementBefore[implicitElement]`").

Procedure 9.4. Removing a keyboard shortcut

1. Click on the "**Existing Keyboard Shortcuts**" tab.

2. Select the keyboard shortcut to be removed by clicking on it.

   You can only remove the keyboard shortcuts you have created. You cannot remove predefined (grayed) keyboard shortcuts.

3. Click on the **Remove** button.

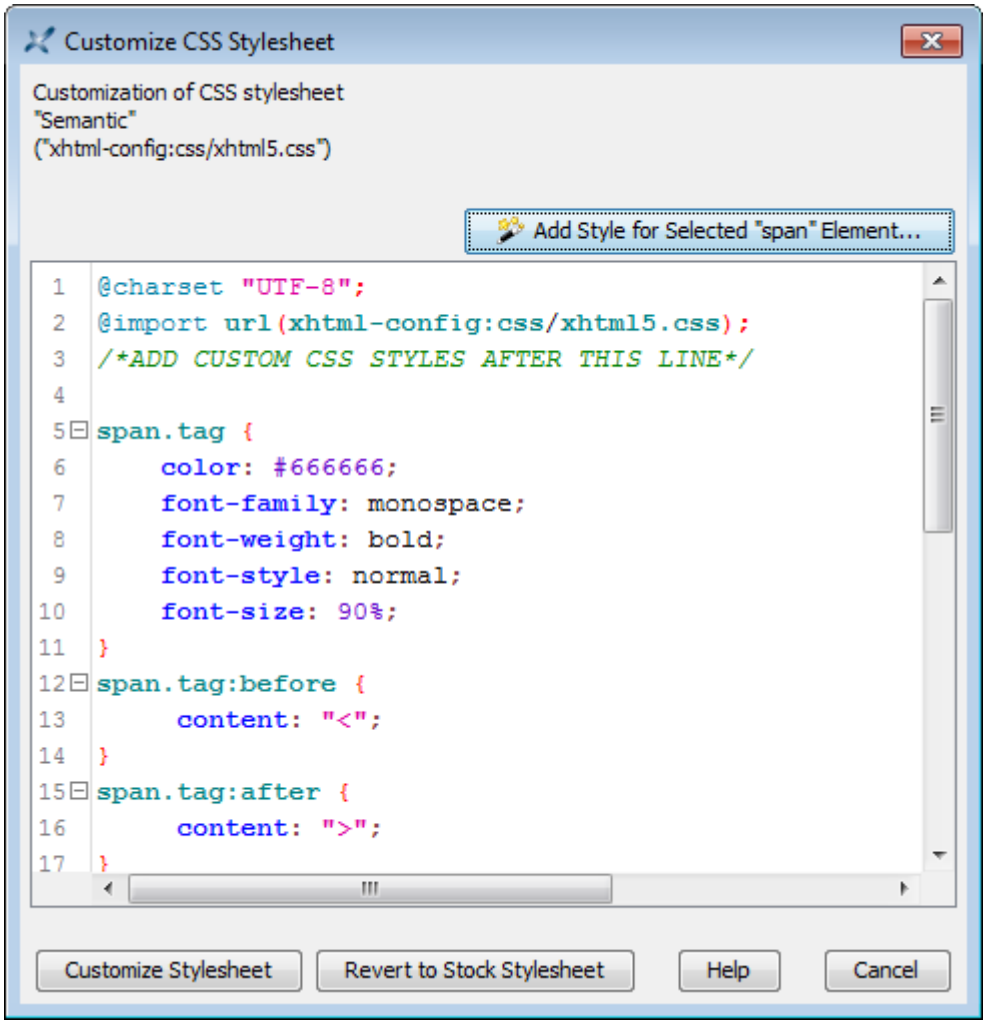
   Note that this operation takes immediate effect: clicking on the **Cancel** button of the "**Add Keyboard Shortcut**" dialog box will not cancel the deletion of the keyboard shortcut[15].

[15]The **Cancel** button of the "**Add Keyboard Shortcut**" dialog box just cancels the action of creating a new keyboard shortcut.

12. The "Customize CSS Stylesheet" dialog box

This dialog box is displayed by menu item **Options** → **Customize Configuration** → **Customize CSS Stylesheet** [47]. It allows customizing the CSS stylesheet being used by the active styled document view.

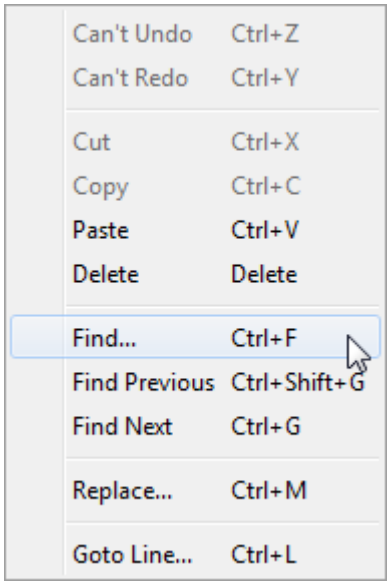
*Figure 9.12. The "**Customize CSS Stylesheet**" dialog box*



This dialog box lets the user type one or more CSS rules after the 3 first lines, *which must not be removed or altered*.

Note that a right click in the text area of this dialog box displays a useful popup menu.

*Figure 9.13. Right-click popup menu of the "**Customize CSS Stylesheet**" dialog box*



Clicking "**Customize Stylesheet**" checks the syntax of specified custom CSS stylesheet and immediately applies the changes to all styled document views using this stylesheet (called "`Semantic`" in the case of the above figure).

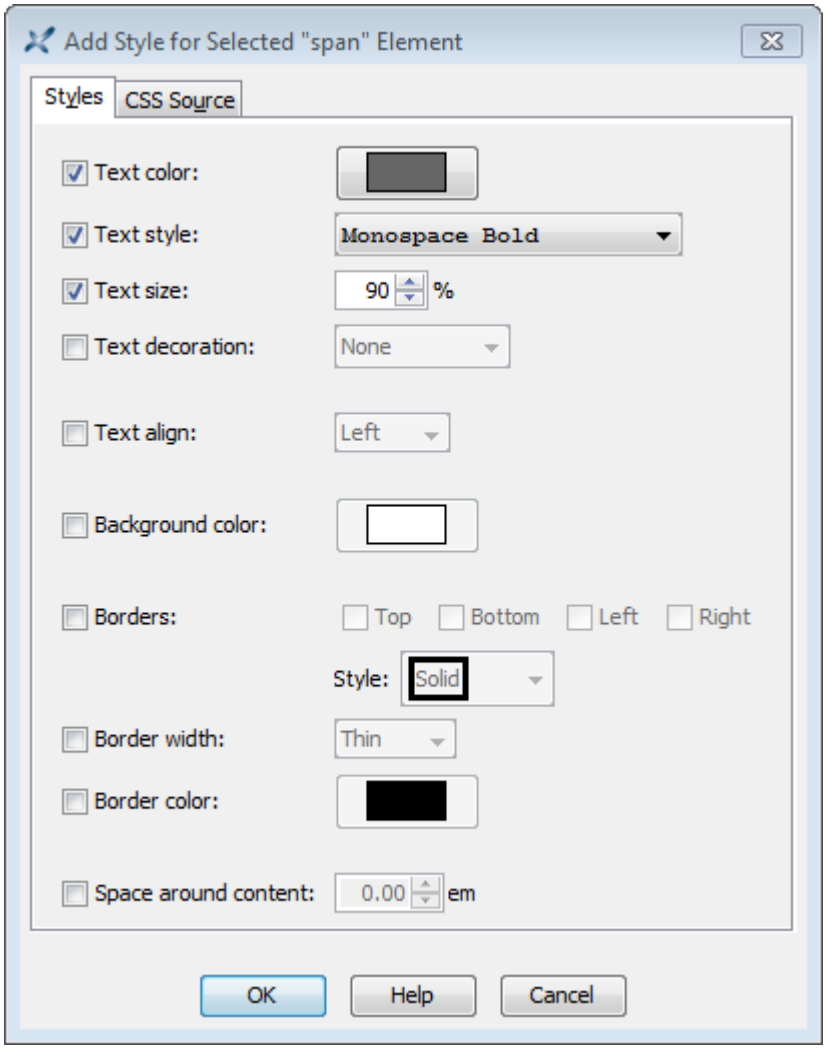
Clicking "**Revert to Stock Stylesheet**" lets the user revert to the stock CSS stylesheet (called "`Semantic`" found in stock "`xhtml-config:css/xhtml5.css`" in the case of the above figure).

12.1. Quickly customizing the style of selected element

The "**Add Style for Selected Element**" button lets the user quickly customize the screen style of the element being implicitly or explicitly selected. Clicking this button displays a style editor, which may be used to quickly and easily specify one or more CSS style properties.

Clicking **OK** adds a new *CSS rule* at the end of the text area of the "**Customize CSS Stylesheet**" dialog box. The body of the rule comprises the properties specified using the style editor, while the selector of the rule is simply the name of selected element. However when selected element is an XHTML element having a `class` attribute, this selector is *`element_name.first_class_name`*. In practice, this means that you may have to modify by hand this automatically determined selector in order to make it more specific.

Figure 9.14. An easy to use style editor

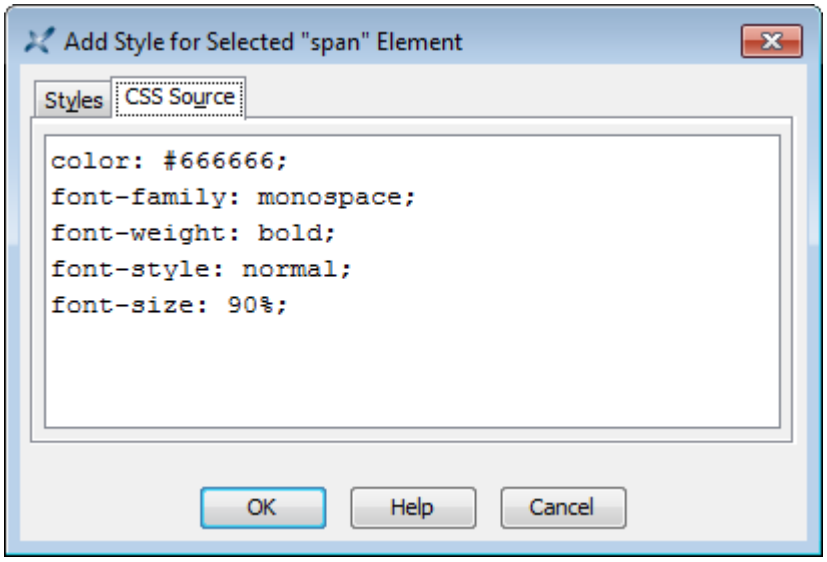


No prior knowledge of CSS is required to use the **Styles** tab of this style editor: simply check the checkbox corresponding to the style (e.g. "**Text color**") you want to add, then possibly adjust the style using the button, combobox or numeric field found next to the checkbox.

In the case of the above figure, selected XHTML `span` (having a "`tag`" `class` attribute) element is given a dark gray text color, a fixed-width bold font and a slightly smaller text size.

Clicking the "**CSS Source**" tab lets the user hand edit the CSS properties specified using the **Styles** tab. The figure below shows you the CSS properties corresponding to the dark gray text color, fixed-width bold font and slightly smaller text size chosen using the **Styles** tab depicted in the above figure.

*Figure 9.15. The "**CSS Source**" tab of the style editor*



When the "**CSS Source**" tab is selected, its content is used *as is* (of course after a successful syntax check) to specify the screen style of selected element. This allows typing CSS properties not supported by the **Styles** tab, for example, `text-indent`, `margin` or even `display`.

The style editor is easy to use however, in some cases, you'll have to workaround some of the limitations of XMLmind XML Editor in terms of CSS support in *XMLmind XML Editor - Support of Cascading Style Sheets (W3C CSS)*. Examples:

- You may have to check "**Border width**" in addition to "**Borders**" in order to successfully add a border around an element.
- The box decoration properties (borders, space around content, etc) are not supported for inline elements. The background color is the only property supported for such elements. A workaround consists in typing "`display: inline-block;`" in the "**CSS Source**" tab.

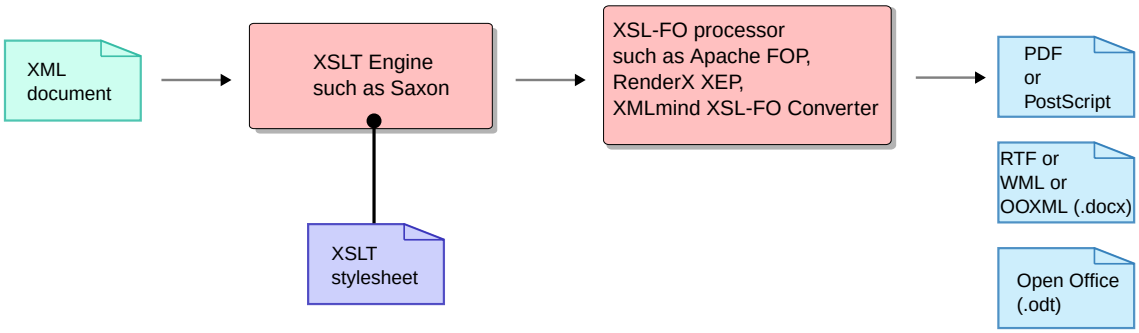
13. The "Customize Document Conversion Stylesheets" dialog box

Using this dialog box is a relatively simple way to influence the layout and style of the deliverable (PDF, RTF, HTML, etc) which results from the document conversion.

13.1. Introduction

The document being edited is converted to other formats by the means of *XSLT stylesheets*.

Figure 9.16. Converting an XML document to an XSL-FO based format



This dialog box allows to:

- Select an XSLT stylesheet other the default one.

  Example: when converting a DocBook document to HTML, the default XSLT stylesheet generates HTML and does not support profiling. this dialog box allows to select any of the following alternate XSLT stylesheets: "**Generate HTML; profiling stylesheet**", "**Generate XHTML**", "**Generate XHTML; profiling stylesheet**".

- Create a custom XSLT stylesheet on the fly.

  A custom XSLT stylesheet created this way merely imports the stock XSLT stylesheet. Example:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">
  <xsl:import href="docbook-config:xsl/fo/docbook.xsl"/>
  <!-- REDEFINE PARAMETERS AND ATTRIBUTE-SETS HERE -->
</xsl:stylesheet>
```

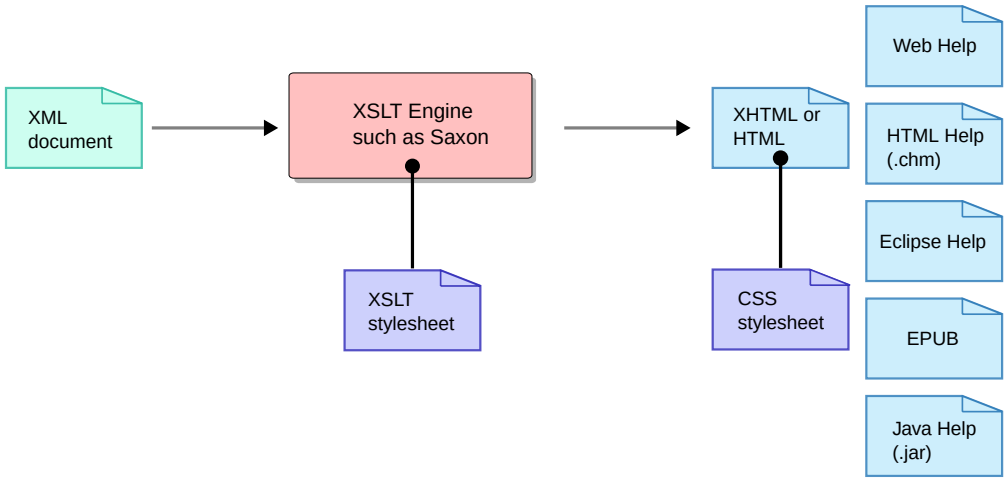
  After such custom XSLT stylesheet has been created, the user is expected to add to it a number of parameters and attribute-sets.

- Invoke a specialized editor —XMLmind XSL Customizer [163]— to modify a user-created XSLT stylesheet.

However, when the document being edited is converted to an *HTML-based format* (Web Help, EPUB, HTML Help, etc)[16], the HTML pages which are automatically generated by the aforementioned XSLT stylesheets are styled mainly by *CSS stylesheets*.

[16] As opposed to XSL-FO based formats (PDF, RTF, .docx, .odt, etc).

Figure 9.17. Converting an XML document to an HTML-based format



When this is the case, this dialog box allows additionally to:

- Select a CSS stylesheet other the default one.
- Create a custom CSS stylesheet on the fly.

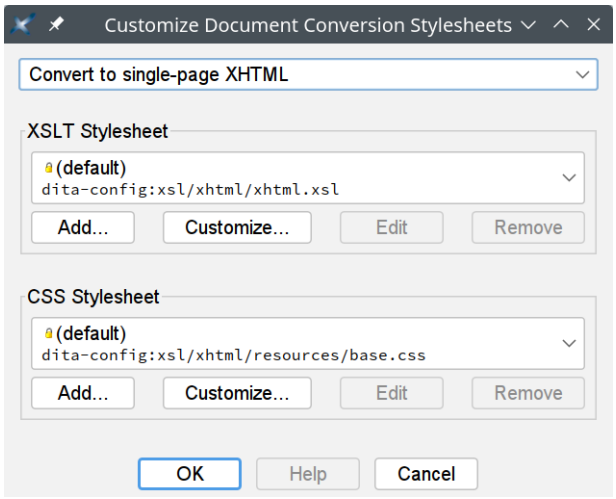
  A custom CSS stylesheet created this way merely imports the stock CSS stylesheet. Example:

  ```
  @charset "UTF-8";
  @import url(base.css);
  /* ADD YOUR STYLES HERE */
  ```

  After such custom CSS stylesheet has been created, the user is expected to add to it a number of styles.
- Invoke a helper application [121] (generally, a text editor) to modify a user-created CSS stylesheet.

13.2. Reference

*Figure 9.18. The "**Customize Document Conversion Stylesheets**" dialog box showing the stylesheets used when a DITA map is converted to single-page XHTML*



"**XSLT stylesheet**" frame:

The combobox

This combobox allows to select:

- the default stock XSLT stylesheet,
- an alternate stock XSLT stylesheet (if any),
- a custom XSLT stylesheet added using the **Add** button,
- a custom XSLT stylesheet created using the **Customize** button.

Add

Add an existing custom XSLT stylesheet. This button displays a dialog box allowing to specify the filename of a custom XSLT stylesheet and also to give it a short title (a title is required).

Customize

Create a custom XSLT stylesheet based on the currently selected stock XSLT stylesheet.

It's not possible to create a customization of a custom XSLT stylesheet.

A custom XSLT stylesheet created this way merely imports the stock XSLT stylesheet. Example:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">
  <xsl:import href="docbook-config:xsl/fo/docbook.xsl"/>
  <!-- REDEFINE PARAMETERS AND ATTRIBUTE-SETS HERE -->
</xsl:stylesheet>
```

After such custom XSLT stylesheet has been created, the user is expected to click **Edit** to populate it with parameters and attribute-sets.

Edit

If the custom XSLT stylesheet has been created using the **Customize** button, this button starts the "XMLmind XSL Customizer" application [163] in order to edit it. Otherwise, this button starts a helper application [121] allowing to edit XSLT files.

It's not possible to edit a stock XSLT stylesheet.

Clicking **Edit** while keep the Shift key pressed allows to start a helper application rather than the "XMLmind XSL Customizer" application.

Remove

Remove the currently selected custom XSLT stylesheet from the combobox. Of course, clicking **Remove** does not delete the XSLT file.

It's not possible to remove a stock XSLT stylesheet.

"**CSS stylesheet**" frame (this frame is absent when the output format is based on XSL-FO):

The combobox

This combobox allows to select:

- the default stock CSS stylesheet,
- an alternate stock CSS stylesheet (if any),

- a custom CSS stylesheet added using the **Add** button,
- a custom CSS stylesheet created using the **Customize** button.

Add

Add an existing custom CSS stylesheet. This button displays a dialog box allowing to specify the filename of a custom CSS stylesheet and also to give it a short title (a title is required).

Customize

Create a custom CSS stylesheet based on the currently selected stock CSS stylesheet.

It's not possible to create a customization of a custom CSS stylesheet.

A custom CSS stylesheet created this way merely imports the stock CSS stylesheet. Example:

```
@charset "UTF-8";
@import url(base.css);
/* ADD YOUR STYLES HERE */
```

After such custom CSS stylesheet has been created, the user is expected to click **Edit** to populate it with styles.

Edit

Starts a helper application [121] allowing to edit CSS files.

It's not possible to edit a stock CSS stylesheet.

Remove

Remove the currently selected custom CSS stylesheet from the combobox. Of course, clicking **Remove** does not delete the CSS file.

It's not possible to remove a stock CSS stylesheet.

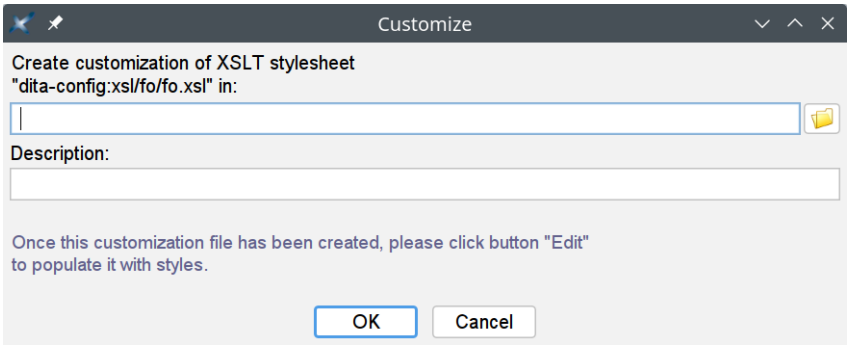
Example 9.4. When converting a DITA map to PDF, use a 12pt base font size and give a light gray background to all the topic titles

1. Open a DITA map in XMLmind XML Editor.

2. Select menu item **Options** → **Customize Configuration** → **Customize Document Conversion Stylesheets**. This displays the "**Customize Document Conversion Stylesheets**" dialog box.

3. Select "**Convert to PDF, PostScript**" in the top combobox.

4. In the "**XSLT stylesheet**" frame, click **Customize**. This displays the following dialog box:



a. Specify the filename of the XSLT stylesheet which is to contain your customization of the stock `dita-config:xsl/fo/fo.xsl` XSLT stylesheet.

b. Also required: give a short title to your customization in the **Description** field.

5. In the "**XSLT stylesheet**" frame, click **Edit**. This displays the main window of the "XMLmind XSL Customizer" application [163].

   a. Select the **Attribute Sets** tab and click **Add** to add attribute `background-color=#CCCCCC` to the attribute-set called `topic-title`. How to do this is detailed in the example below [169].

   b. Select the **Parameters** tab and click **Add** to specify the following parameter: `base-font-size=12pt`. How to do this is detailed in the example below [170].

   c. Click toolbar button **Save** to save your changes.

   d. Click toolbar button **Quit** to close "XMLmind XSL Customizer" window.

6. Click **OK** to close the the "**Customize Document Conversion Stylesheets**" dialog box.

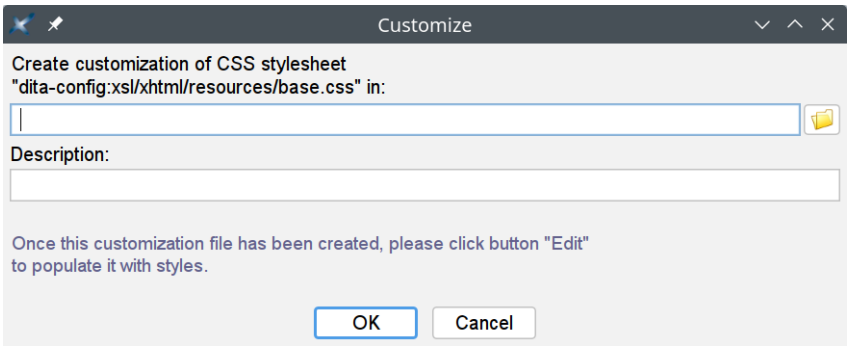
Example 9.5. When converting a DITA map to single-page XHTML, use a 12pt base font size and give a light gray background to all the topic titles

1. Open a DITA map in XMLmind XML Editor.

2. Select menu item **Options** → **Customize Configuration** → **Customize Document Conversion Stylesheets**. This displays the "**Customize Document Conversion Stylesheets**" dialog box.

3. Select "**Convert to single-page XHTML**" in the top combobox.

4. In the "**CSS stylesheet**" frame, click **Customize**. This displays the following dialog box:



   a. Specify the filename of the CSS stylesheet which is to contain your customization of the stock `dita-config:xsl/xhtml/resources/base.css` XSLT stylesheet.

   b. Also required: give a short title to your customization in the **Description** field.

5. In the "**CSS stylesheet**" frame, click **Edit**. This displays the "**Helper Application Not Found**" dialog box[17]. This dialog box allows to specify which editor is to be used to edit a CSS file. By default, the "**Helper Application Not Found**" dialog box suggests to use a text editor.

   a. Replace the line:

[17] Unless the user has already specified which is the helper application she/he prefers to edit a CSS stylesheet. This kind of preference is specified by using **Options** → **Preferences**, **Tools**|**Helper Applications** section. More information in Section 6.8.5, “Helper applications options” [121].

```
/* ADD YOUR STYLES HERE */
```

by:

```
body {
    font-size: 12pt;
}

.part-title,
.chapter-title,
.appendices-title,
.appendix-title,
.section1-title,
.section2-title,
.section3-title,
.section4-title,
.section5-title,
.section6-title,
.section7-title,
.section8-title,
.section9-title,
.topic-title {
    background-color: #CCCCCC;
}
```

Make sure not to modify the `@import` line.

b. Save your changes.

Make sure to use an UTF-8 encoding for the save file.

c. Quit the text editor.

6. Click **OK** to close the the "**Customize Document Conversion Stylesheets**" dialog box.

How to automatically select a set of custom stylesheets when converting a given document?

The "**Customize Document Conversion Stylesheets**" dialog box lets a user create and select custom XSLT or CSS stylesheets which are used when converting XML documents to formats such PDF, RTF, EPUB, etc.

The custom XSLT or CSS stylesheets selected using the "**Customize Document Conversion Stylesheets**" dialog box are used for all the documents having the same document type (e.g. all DocBook documents or all XHTML 5 documents). Moreover, a custom stylesheet selection made this way is persistent across editing sessions.

Now let's suppose you want to use a custom XSLT stylesheet and a custom CSS stylesheet when converting DITA maps to Web Help and you want to use a custom XSLT stylesheet when converting DITA maps to PDF. However, you want to use these custom stylesheets *only when converting DITA maps containing tutorials*. You consider that the stock XSLT and CSS stylesheets are just fine for the other kinds of DITA documents, for example, for reference manuals.

Here's how to do this:

1. Use the "**Customize Document Conversion Stylesheets**" dialog box, select "**Convert To Web Help**" and declare your custom XSLT stylesheet `webhelp_tutorials.xsl`.

   Specify "`Tutorial Styles`" in the **Description** field of the custom XSLT stylesheet chooser dialog box [159].

   Do *not* select `webhelp_tutorials.xsl`, as this would instruct **XXE** to use this custom XSLT stylesheet for *all* DITA map documents.

2. Use the "**Customize Document Conversion Stylesheets**" dialog box, select "**Convert To Web Help**" and declare your custom CSS stylesheet `webhelp_tutorials.css`.

   Specify "`Tutorial Styles`" in the **Description** field of the custom CSS stylesheet chooser dialog box [160].

   Do *not* select `webhelp_tutorials.css`.

3. Use the "**Customize Document Conversion Stylesheets**" dialog box, select "**Convert To PDF, PostScript**" and declare your custom XSLT stylesheet `fo_tutorials.xsl`.

   Specify "`Tutorial Styles`" in the **Description** field of the custom XSLT stylesheet chooser dialog box [159].

   Do *not* select `fo_tutorials.xsl`.

   Make sure to specify exactly the same description for all the custom XSLT or CSS stylesheets belonging to the same stylesheet set.

4. In all the DITA maps containing tutorials you'll author, insert the following processing-instruction:

   ```
   <?xxe-stylesheet-set Tutorial Styles?>
   ```

   as a direct child of the root element (e.g. the DITA `map` or `bookmap`).

The position as a child of the root element does not matter. For example, you may insert this processing-instruction at the beginning or at the end of the `map` root element.

```
<map xml:lang="en-US">
  <?xxe-stylesheet-set Tutorial Styles?>
  <title>Tutorial: how to fix gizmos</title>
  ...
</map>
```

From now, when converting a DITA map containing `<?xxe-stylesheet-set Tutorial Styles?>` to WebHelp or PDF, **XXE** will automatically use `webhelp_tutorials.xsl`, `webhelp_tutorials.css`, `fo_tutorials.xsl`.

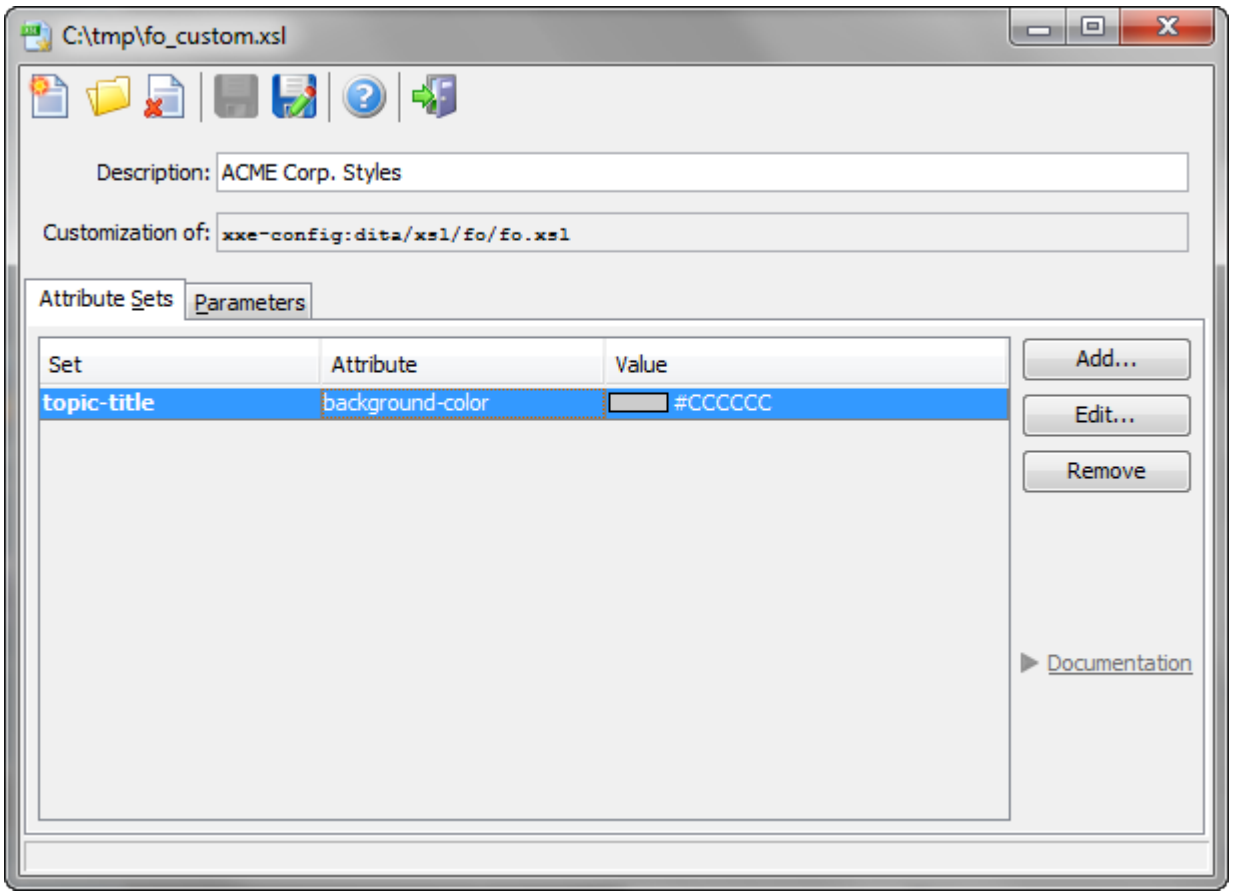
What if you a DITA map containing `<?xxe-stylesheet-set Tutorial Styles?>` to EPUB or DOCX? Remember that you did not use "**Customize Document Conversion Stylesheets**" to declare custom XSLT or CSS stylesheets called "`Tutorial Styles`" for these output formats. Well, it will work just fine. When converting your tutorial to EPUB or DOCX, processing-instruction `<?xxe-stylesheet-set Tutorial Styles?>` will be silently ignored.

13.3. The "XMLmind XSL Customizer" application

13.3.1. Introduction

The "XMLmind XSL Customizer" application is a companion application embedded in XMLmind XML Editor and XMLmind XSL Utility.

Figure 9.19. "XMLmind XSL Customizer" main window



This application is invoked by clicking the **Edit** XSLT stylesheet button found in XMLmind XML Editor and XMLmind XSL Utility. It allows to modify a custom XSLT stylesheet created by clicking the **Customize** XSLT stylesheet button found in XMLmind XML Editor and XMLmind XSL Utility.

A custom XSLT stylesheet created this way merely imports the stock XSLT stylesheet. Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<?stylesheet-label ACME Corp. Styles?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:import href="dita-config:xsl/fo/fo.xsl"/>
  <!-- REDEFINE PARAMETERS AND ATTRIBUTE-SETS HERE -->
</xsl:stylesheet>
```

"XMLmind XSL Customizer" is not an XSLT editor. However it allows to:

- add or change attributes in some of the *attribute sets* supported by the XSLT stylesheet,
- change the value of some of the *parameters* supported by the XSLT stylesheet,

and this, *without prior knowledge of XSLT*.

What are attribute sets and parameters?

XSLT stylesheets are often (but not always) parameterized by the means of:

- Parameters. DocBook XSL stylesheet example:

```
<xsl:param name="paper.type"❶>USletter❷</xsl:param>
```

  ❶ The name of the parameter is "`paper.type`".

  ❷ The value of the parameter is literal string "`USletter`".

- Attribute sets. DocBook XSL stylesheet example:

```
<xsl:attribute-set name="monospace.verbatim.properties"❶
                   use-attribute-sets="verbatim.properties monospace.properties"❷>
  <xsl:attribute name="text-align">start</xsl:attribute>❸
  <xsl:attribute name="wrap-option">no-wrap</xsl:attribute>❹
</xsl:attribute-set>
```

  ❶ The name of the attribute set is "`monospace.verbatim.properties`".

  ❷ This attribute set includes two other attribute sets: `verbatim.properties` and `monospace.properties`.

  ❸ This attribute set directly contains attribute `text-align="start"`.

  ❹ This attribute set directly contains attribute `wrap-option="no-wrap"`.

  An attribute set can contain any attribute. However attribute sets are mainly used in XSLT stylesheets which generate XSL-FO. The XSL-FO intermediate file generated by the XSLT stylesheet is then processed by programs such as Apache FOP, RenderX XEP, Antenna House XSL Formatter, XMLmind XSL-FO Converter, etc, in order to create the deliverable: PDF, PostScript, RTF, `.docx`, `.odt`, etc. The attribute sets are used in this case because they are the only way to influence the look of the deliverable. Such attribute sets contain standard XSL-FO presentation attributes (very similar to the CSS properties): `color`, `font-family`, `line-height`, `margin-left`, etc.

13.3.2. Reference

Toolbar buttons

New

Create a customization of an existing XSLT stylesheet. This button displays the standard file chooser dialog box allowing to choose the XSLT stylesheet for which a customization is to be created.

Open

Open a custom XSLT stylesheet previously created by clicking the **New** button. This button displays the standard file chooser dialog box allowing to choose the custom XSLT stylesheet to be opened.

Close

Close currently opened XSLT stylesheet.

Save

Save the changes made to currently opened XSLT stylesheet.

Save As

Save currently opened XSLT stylesheet to a different file.

Help

Display this online help.

Quit

Close "XMLmind XSL Customizer" main window.

Because "XMLmind XSL Customizer" is an (embedded) application and not a modal dialog box, you can keep it open while converting an XML document in XMLmind XML Editor or in XMLmind XSL Utility. This allows to experiment with attribute sets and parameters until you are satisfied with the result of the conversion.

Information fields about the currently opened XSLT stylesheet

Description:

Short description of the currently opened custom XSLT stylesheet. XMLmind XML Editor requires a custom XSLT stylesheet to have such description.

Customization of:

Read-only text field: URI of the stock XSLT stylesheet for which the currently opened XSLT stylesheet is a customization.

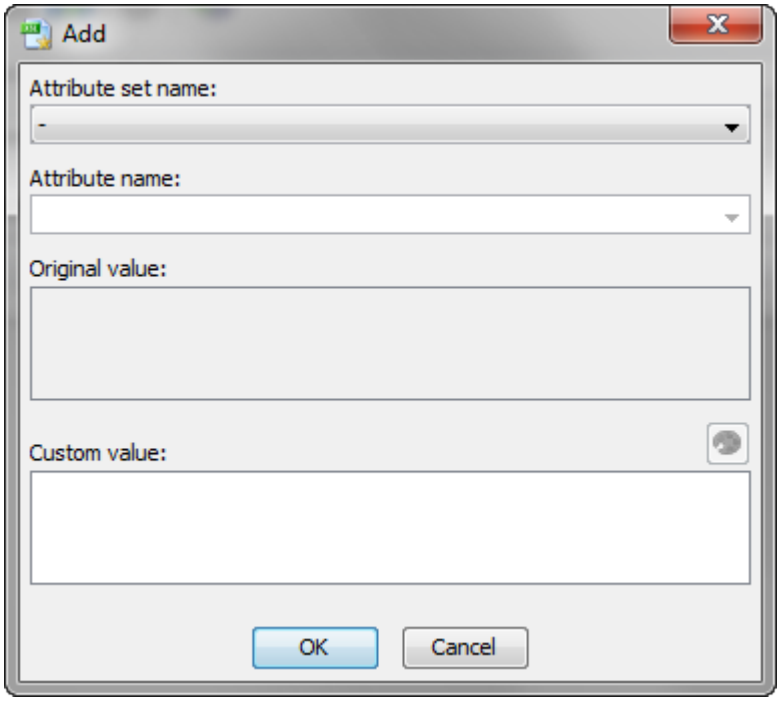
Attribute Sets tab buttons

Add

Add an attribute to one of the attribute sets supported by currently opened XSLT stylesheet.

This button displays the **Add**/**Edit** attribute dialog box. How to use this dialog box is described in the example below [169].

*Figure 9.20. The **Add/Edit** attribute dialog box*



XMLmind XSL Customizer is designed for users who cannot “program in XSLT”. These users are expected to enter literal values, not XSLT elements, in the **Custom value** field. For example, they are expected to enter something like:

```
20pt
```

as the value of the `font-size` attribute, and not something like:

```
<xsl:value-of select="$body.font.master * 2"/>
<xsl:text>pt</xsl:text>
```

However, if the user happens to know what she/he is doing and nevertheless enters one or more XML nodes in the **Custom value** field, then XMLmind XSL Customizer will ask her/him to confirm that this is really what she/he wants and make it work.

Edit

Modify currently selected attribute.

This button displays the **Add/Edit** attribute dialog box.

Remove

Remove currently selected attribute.

Documentation

Start the web browser and make it display the page containing the documentation about currently selected attribute. This button is disabled (grayed) when such documentation is not available. For now, only the DocBook XSL Stylesheets provide some documentation for their attribute sets.

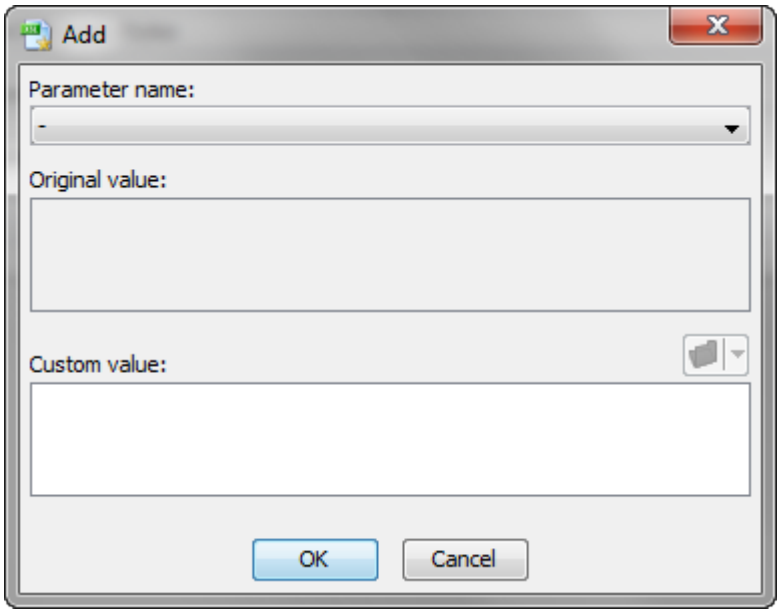
Parameters tab buttons

Add

Set one of the parameters supported by currently opened XSLT stylesheet.

This button displays the **Add**/**Edit** parameter dialog box. How to use this dialog box is described in the example below [170].

*Figure 9.21. The **Add**/**Edit** parameter dialog box*



XMLmind XSL Customizer is designed for users who cannot “program in XSLT”. These users are expected to enter literal values, not XSLT elements, in the **Custom value** field. For example, they are expected to enter something like:

```
40pt
```

as the value of the *`body.start.indent`* parameter, and not something like:

```
<xsl:value-of select="$body.font.master * 4"/>
<xsl:text>pt</xsl:text>
```

However, if the user happens to know what she/he is doing and nevertheless enters one or more XML nodes in the **Custom value** field, then XMLmind XSL Customizer will ask her/him to confirm that this is really what she/he wants and make it work.

Edit

Modify currently selected parameter.

This button displays the **Add**/**Edit** parameter dialog box.

Remove

Remove currently selected parameter.

Documentation

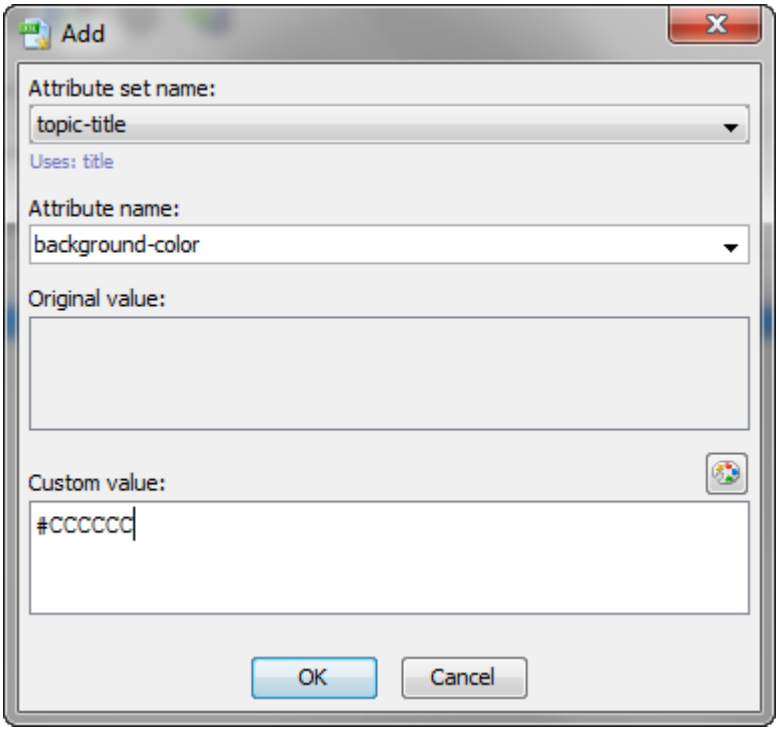
Start the web browser and make it display the page containing the documentation about currently selected parameter. This button is disabled (grayed) when such documentation is not available.

Example 9.6. When converting a DITA map to PDF, give a light gray background to all the topic titles

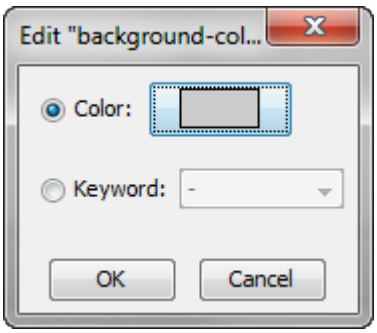
This is specified by adding attribute `background-color=#CCCCCC` to the attribute-set called `topic-title`.

1. Select the **Attribute Sets** tab.
2. Click **Add**. This displays the **Add/Edit** attribute dialog box.
3. In the "**Attribute set name:**" combobox, select `topic-title`.
4. In the "**Attribute:**" combobox, type "`background-color`" or select this commonly used attribute from the drop down list.

   The "**Original value:**" read-only text field remains empty, indicating that the stock XSLT stylesheet does not specify attribute `topic-title/background-color`.
5. In the "**Custom value:**" field, type "`#CCCCCC`".



Or more simply, click "**Edit style attribute**" and use the `background-color` editor to specify a light gray color.



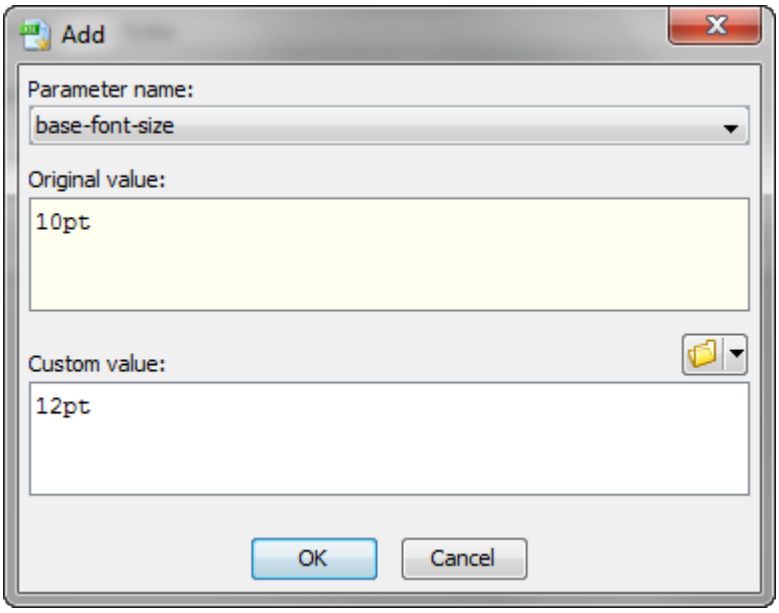
6. Click **OK** to close the dialog box.

Example 9.7. When converting a DITA map to PDF, use a 12pt base font size

This is specified by setting parameter `base-font-size` to `12pt`.

1. Select the **Parameters** tab.
2. Click **Add**. This displays the **Add/Edit** parameter dialog box.
3. In the "**Parameter name:**" combobox, select `base-font-size`.

   The "**Original value:**" read-only text field changes to "`10pt`", which is the value of parameter `base-font-size` specified in the stock XSLT stylesheet.
4. In the "**Custom value:**" field, type "`12pt`".



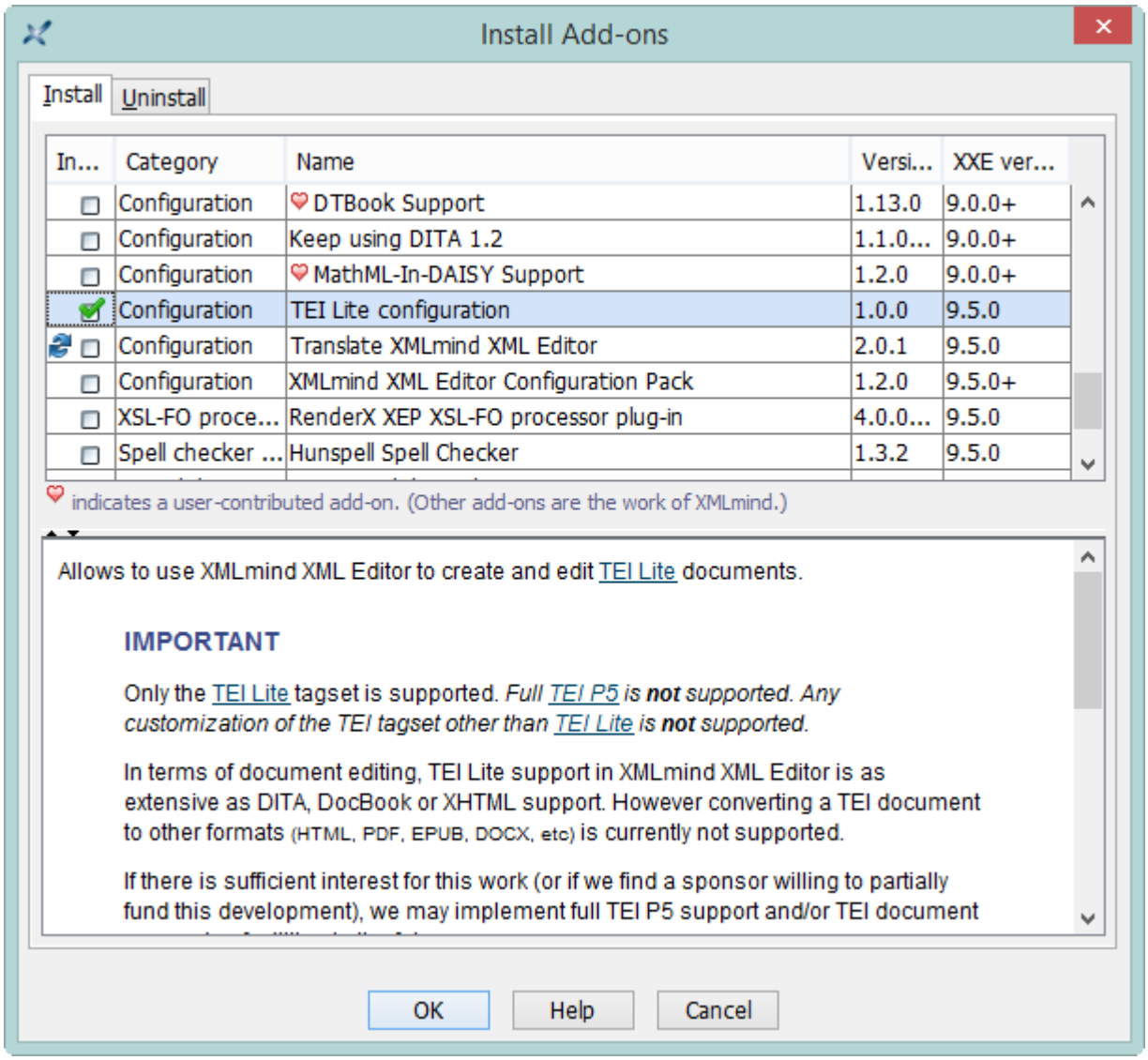
5. Click **OK** to close the dialog box.

14. The "Install Add-ons" dialog box

This dialog box is not available if **XXE** has been deployed using Java™ Web Start or as an applet.

The "**Install Add-ons**" dialog box is displayed by selecting menu item **Options** → **Install Add-ons** [46]. It is used to install, uninstall or upgrade add-ons.

*Figure 9.22. The "**Install Add-ons**" dialog box*



XXE add-on manager assumes that proxy servers, if any, have been properly specified using the Proxies section [128] of the **Preferences** dialog box.

If a problem occurs when listing the add-ons available for download and/or when uninstalling/installing the add-ons, use the "**Show Message Log**" button [89] of the status bar, category "**Install Add-ons**", to see logged error messages.

Procedure 9.5. Install an add-on

1. Select the **Install** tab.

2. Click on the checkbox of the add-on to be installed (looks like this ☐, when clicked looks like this ☑).

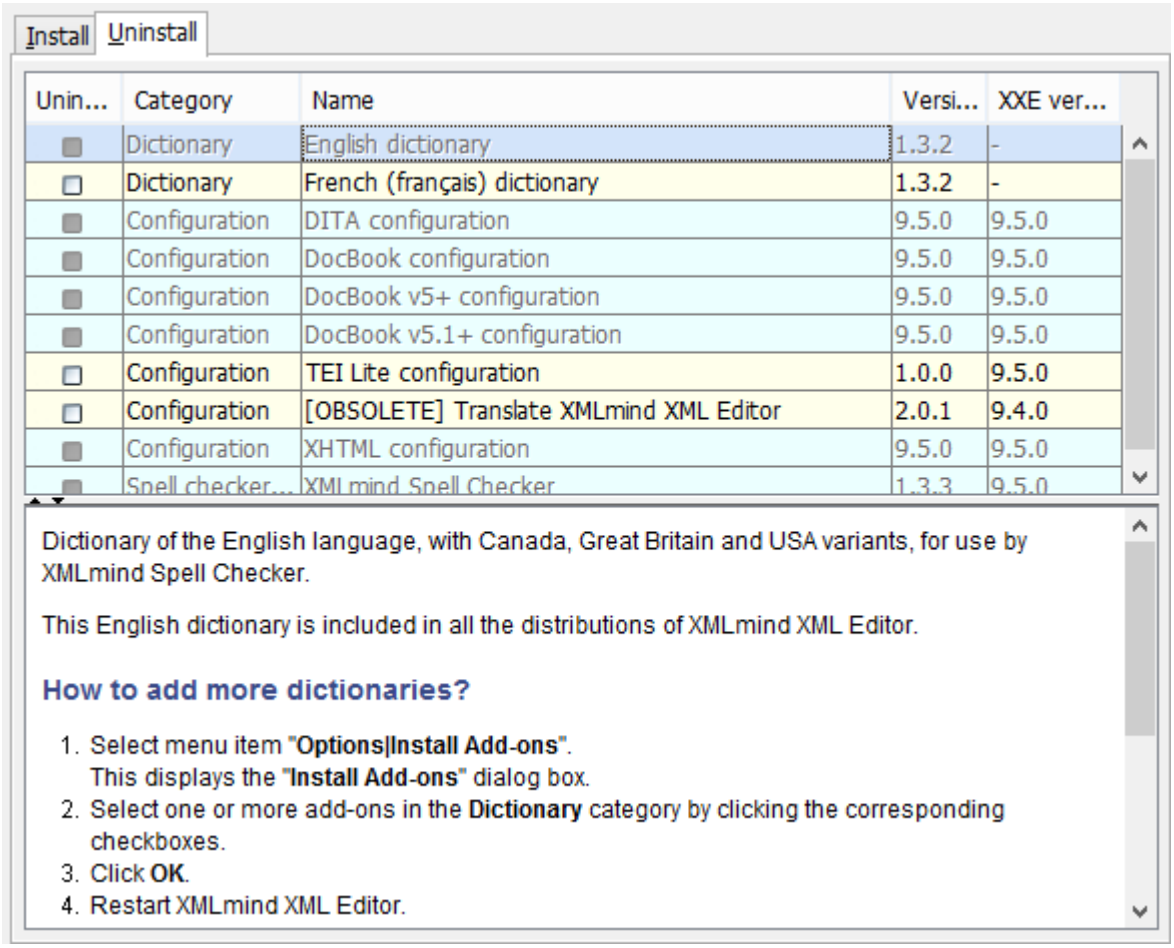
   If an add-on is not listed in the the **Install** tab, it's either because this add-on is already installed[18] or because it is not available for the edition or version of **XXE** you are running or for your platform (e.g. not available for Linux).

3. Click **OK**.

4. Restart **XXE** as instructed by the dialog box.

Procedure 9.6. Uninstall an add-on

*Figure 9.23. The **Uninstall** tab*



1. Select the **Uninstall** tab.

2. Click on the checkbox of the add-on to be uninstalled (looks like this ☐, when clicked looks like this ☑).

- The rows of the **Uninstall** add-on table are given a *light yellow background* when the corresponding add-on is installed in the user preferences directory

[18] In such case, this add-on is listed in the **Uninstall** tab.

(*XXE_user_preferences_dir*/addon/), and a *light cyan background* when the corresponding add-on is installed in **XXE** installation directory (*XXE_install_dir*/addon/).

- In order to make automatically upgrading add-ons each time you upgrade **XXE** a simple and reliable process, it is not possible to install, uninstall or upgrade any of the bundled add-ons found inside the **XXE** installation directory. That's why all light cyan rows are disabled (grayed).
- If a light yellow row is disabled (grayed), it's either because you have insufficient privileges to uninstall the corresponding add-on, or on the Mac and Linux, because the subdirectory of *XXE_user_preferences_dir*/addon/ containing this add-on is a *symlink*.

3. Click **OK**.
4. Restart **XXE** as instructed by the dialog box.

Procedure 9.7. Upgrade an add-on

1. Select the **Install** tab.
2. Click on the checkbox of the add-on to be upgraded (looks like this).

- There is no need to also explicitly uninstall the corresponding add-on because this is done automatically for you. However it is harmless to do so.
- If the row corresponding to the add-on to be upgraded is disabled (grayed), it's either because you have insufficient privileges to first uninstall this add-on, or on the Mac and Linux, because the subdirectory of *XXE_user_preferences_dir*/addon/ containing this add-on is a *symlink*.

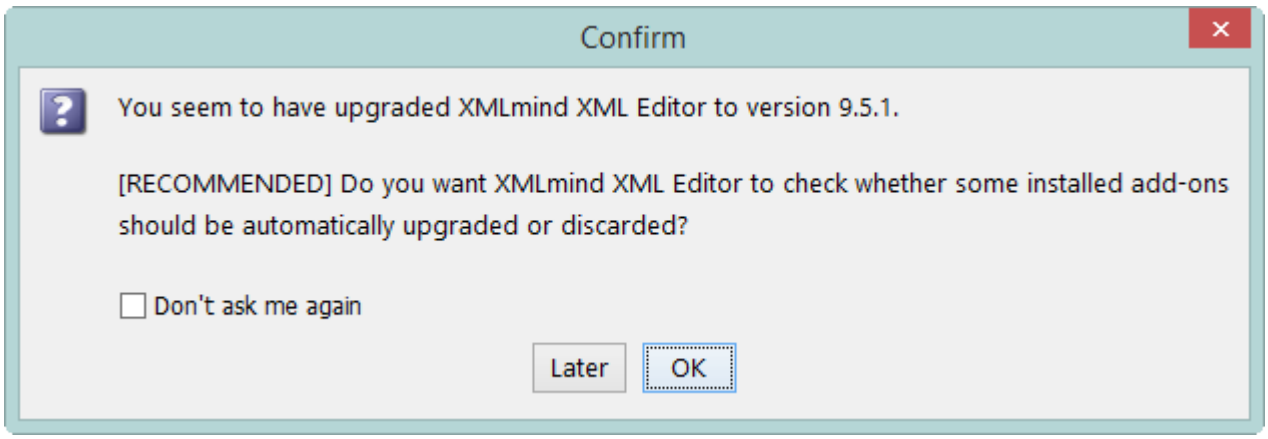
3. Click **OK**.
4. Restart **XXE** as instructed by the dialog box.

14.1. *Automatically* upgrading installed add-ons

If you hold the Shift key and click the **Options** → **Install Add-ons** [46] menu item, you'll *automatically* upgrade user-installed add-ons (of course, if any and if needed to).

However, doing this should not be required as user-installed add-ons are automatically upgraded too the first time you'll start **XXE** after upgrading the application.

The user is simply asked to confirm that she/he accepts to proceed with this automatic task.



If the user clicks **Later**, **XXE** will keep “nagging her/him” each time the application is started until all obsolete add-ons are upgraded or uninstalled (either automatically or “by hand”, using the "**Install Add-ons**" dialog box). An expert user who really wants to keep obsolete add-ons installed on her/his computer may choose to select "**Don't ask me again**".

14.2. How to “*manually*” install, uninstall or upgrade an add-on

Add-ons are found in the two following `addon/` directories:

1. Bundled add-ons (such as "**XMLmind Spell Checker**" and "**DITA configuration**") are found in *`XXE_install_dir`*`/addon/` (or any of its subdirectories).

2. Optional add-ons (such as "**Hunspell Spell Checker**" and "**TEI Lite configuration**") are found in *`XXE_user_preferences_dir`*`/addon/` (or any of its subdirectories), where **XXE** user preferences directory is:
   - `$HOME/.xxe11/` on Linux,
   - `$HOME/Library/Application Support/XMLmind/XMLEditor11/` on the Mac,
   - `%APPDATA%\XMLmind\XMLEditor11\` on Windows. Example: `C:\Users\john\AppData\Roaming\XMLmind\XMLEditor11\`.

     If you cannot see the "**AppData**" directory using Microsoft Windows File Manager, turn on **Tools** → **Folder** → **Options** → **View** → **File and Folders** → **Show hidden files and folders**.

When “*manually*” installing an add-on, you may choose either of the two above `addon/` directories. However, it is recommended to use *`XXE_user_preferences_dir`*`/addon/` because this way, your add-ons will be automatically upgraded the next time you'll upgrade **XXE**.

Procedure 9.8. “Manually” install an add-on

1. Visit this "**Add-ons**" page using your web browser to download the add-on distribution.

2. Unzip the add-on distribution in either *`XXE_user_preferences_dir`*`/addon/` (recommended) or *`XXE_install_dir`*`/addon/`.

   For example, unzipping `bidi_support-`*`N_N_N`*`.zip` in *`XXE_user_preferences_dir`*`/addon/` creates subdirectory *`XXE_user_preferences_dir`*`/addon/bidi_support/`.

3. Clear *the Quick Start cache* [127].

How to clear the Quick Start cache?

The simplest and quickest is to delete “by hand” directory *XXE_user_preferences_dir*/cache/ (if it exists).

Alternatively, this can be done from within **XXE** by selecting **Options** → **Preferences** and then clicking the **Clear** button found after the "**Quick Start cache**" toggle in the **Advanced**|**Cached Data** section [126].

4. Restart **XXE** and check installed add-ons using **Options** → **Install Add-ons** [46].

Procedure 9.9. “Manually” uninstall an add-on

1. Delete the subdirectory of addon/ containing the distribution of the add-on.

   For example, uninstall the "**Bidi Support**" add-on by deleting subdirectory *XXE_user_preferences_dir*/addon/bidi_support/.

2. Clear *the Quick Start cache* [127]. See note [175] above explaining how to do this.

3. Restart **XXE** and check installed add-ons using **Options** → **Install Add-ons** [46].

Procedure 9.10. “Manually” upgrade an add-on

1. Delete the subdirectory of addon/ containing the distribution of the old version of add-on.

2. Visit this "**Add-ons**" page using your web browser to download the distribution of the new version of the add-on.

3. Unzip the distribution of the new version of the add-on in either *XXE_user_preferences_dir*/addon/ (recommended) or *XXE_install_dir*/addon/.

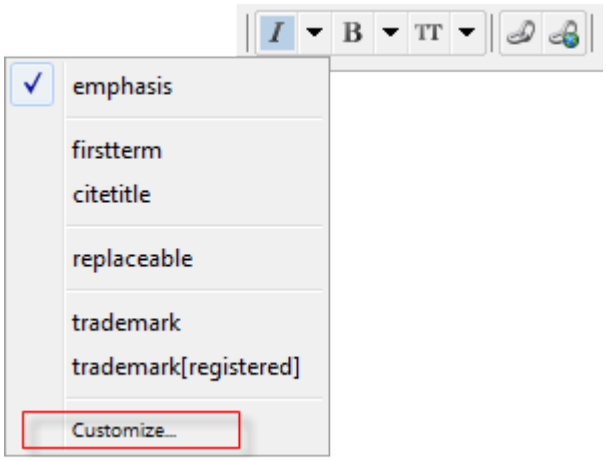
4. Clear *the Quick Start cache* [127]. See note [175] above explaining how to do this.

5. Restart **XXE** and check installed add-ons using **Options** → **Install Add-ons** [46].

15. Dialog box allowing to edit “text style” menu items

This dialog box is displayed when you select the **Customize** menu item found at the end of a “text style” menu. Text style menus are displayed when you click the arrow button found next to a “text style” toggle. The XHTML, DocBook and DITA tool bars all start with a number of text style toggles. More information about text style toggles below [177].

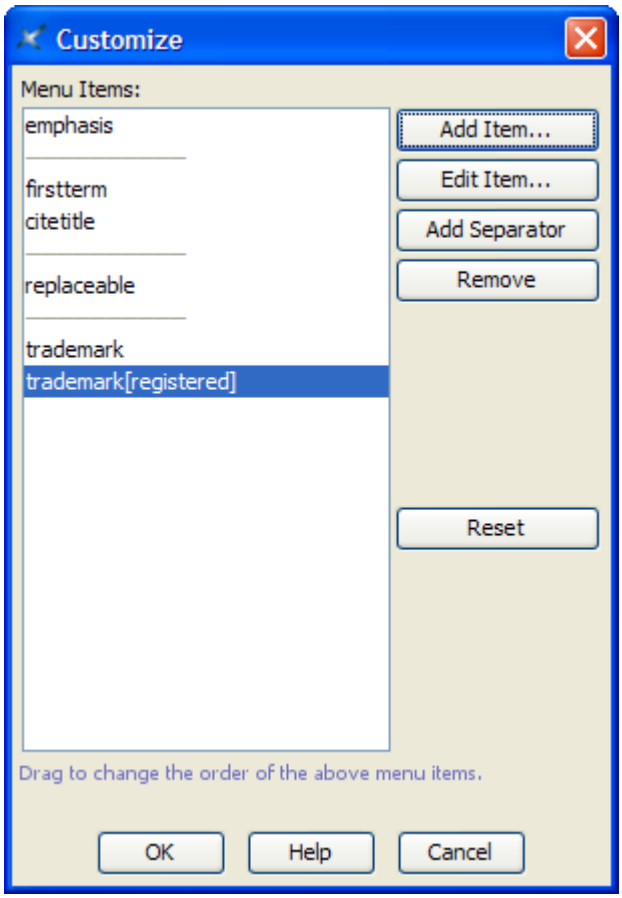
Figure 9.24. Toggles found at the beginning of the DocBook tool bar



In the above screenshot, the caret is inside an `emphasis` element and the user clicked the arrow button next to a “italic text style” toggle.

The **Customize** menu item displays a dialog box allowing to edit the items of a “text style” menu:

Figure 9.25. Customizing the items of the DocBook' “italic text style” menu



Add Item

Add an item before selected item if any; at the end of the menu otherwise. Clicking this button displays the item editor dialog box [178].

When possible, select in your document an instance of the element you would like to add and then copy it to the clipboard (**Ctrl**+**C**). This is the quickest and safest way to add an item to a “text style” menu, as the item editor dialog box [178] has a handy "**Paste Element**" button.

Edit Item

Edit selected item. Clicking this button displays the item editor dialog box [178].

Add Separator

Add a separator before selected item if any; at the end of the menu otherwise.

Remove

Remove selected item or separator.

Reset

Restore the stock menu items.

Notice that the order of items within the menu may be modified by dragging a label in the list.

About “text style” toggles

The XHTML, DocBook and DITA tool bars all start with a number of “text style” toggles. These toggles emulate the behavior of the **Bold**, **Italic**, **Underline**, etc, toggles found in the tool bars of almost all word-processors.

Note that, by emulating what does a word-processor, these toggles behave very differently from the conversion buttons found in the previous versions of the XHTML, DocBook and DITA tool bars:

- Such toggles add a given “text style” to text spans not having this text style and remove a given text style to text spans already having this text style.
- Such toggles do not require some text to be explicitly selected. If the caret is in the middle of a word, then the toggle acts on this word. If the caret is before or after a word, the toggle acts on this text location by inserting an empty text span having or on the contrary, not having, the corresponding text style.

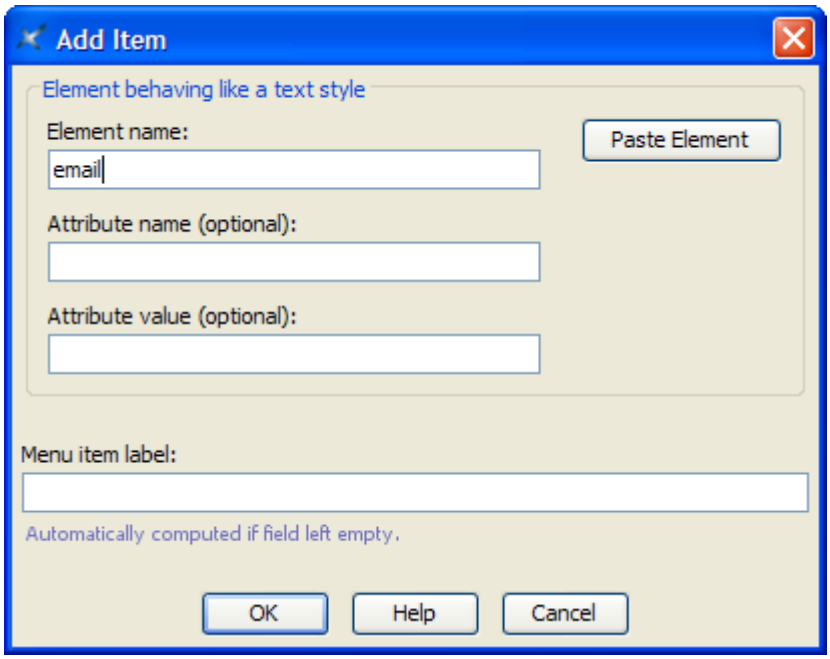
The main difference with word-processors is that **XXE**'s “text style” toggles supports the node selection in addition to the text selection. For example, if you want to apply a “text style” to an element in its entirety, suffice to explicitly select it. This is more handy that selecting all the text this element contains.

A word-processor supports just a few text styles. A document type such as DocBook supports dozens of different text styles. (In XML, what we call a *text style* is an inline element having a mixed content.) That's why you'll often find next to a “text style” toggle, an arrow button displaying a menu of “text style” checkboxes. Moreover the entries of this “text style” menu can be easily customized by the end-user by the means of the **Customize** dialog box [175].

Such quick and easy customization is really needed in the case of document types such as XHTML, DocBook and DITA. For example, the DocBook “bold text style” menu contains checkboxes for `emphasis[@role="bold"]`, `abbrev`, `guilabel`, `guibutton`, `guimenuitem`, `guisubmenu`, `keycap`, `keysym`. But what if a user never uses the `keycap` and `keysym` elements and often uses the `application` element? Answer: suffice for her/him to customize the entries of the “bold text style” menu.

15.1. Dialog box allowing to edit a “text style” menu item

Figure 9.26. The menu item editor



Try to use the "Paste Element" button as much as possible

The quickest and safest way to add an item to a “text style” menu is to click the "**Paste Element**" button. However in order to do that, you'll have to first select in your document an instance of the element you would like to add and then copy it to the clipboard (**Ctrl**+**C**).

Element name

The qualified name[19] of the element behaving like a text style. *Make sure to specify an inline element having a mixed content.*

DocBook 5 example: `email`. Notice that there is no need to specify a namespace prefix as http://docbook.org/ns/docbook is generally the default namespace of DocBook 5 elements.

Attribute name

In some cases, the element you want to specify has a required attribute. This required attribute *must* be specified in this text field.

DocBook 5 example: an external `link` element has a required `xlink:href` attribute.

Attribute value

In some cases, the element you want to specify has a specific attribute value which characterizes it. This attribute value *must* be specified in this text field and the attribute name *must* be specified in the **Attribute name** text field.

DocBook 5 example: element: `trademark`, attribute: `class`, attribute value: `registered`.

[19] Use the same namespace prefixes as those declared in your documents. Such namespace prefixes may be listed by selecting **Tools** → **Declare Namepace** [38].

Menu item label

The label of the menu item. Must be less than 40 characters long.

If you leave this field empty, a label will be automatically computed out of the element name, attribute name and attribute value. In the case of the above DocBook 5 examples, the automatically computed labels are: `email`, `link[xlink:href]` and `trademark[registered]`.

16. The media player dialog box

A DITA Topic or DocBook 5.1 document containing media elements is expected to be converted to *XHTML 5 based formats*, that is, XHTML 5 multi-page or single-page, Web Help containing XHTML 5 pages or EPUB 3.

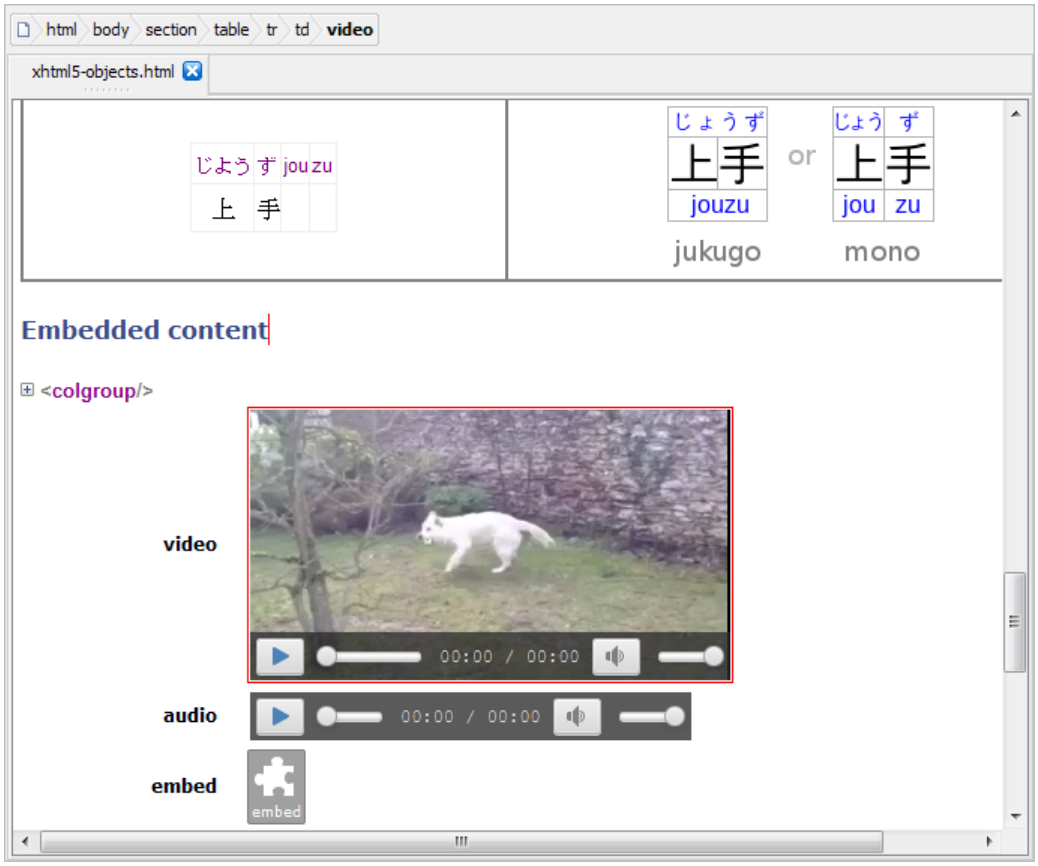
In practice, this means that you'll have to use **Options** → **Customize Configuration** → **Customize Document Conversion Stylesheets** [47] to select the XSL stylesheets which target XHTML 5 based formats.

16.1. Adding audio or video to your document

After opening an XHTML 5, DITA Topic or DocBook 5.1 document in **XXE**, you'll find an "insert/add media object" toolbar button allowing to add a media element (e.g. XHTML 5 `audio` and `video` elements) to your document.

Once properly configured, a media element is rendered by embedding a *media player* in the styled view. This media player is similar to those found in Web browsers.

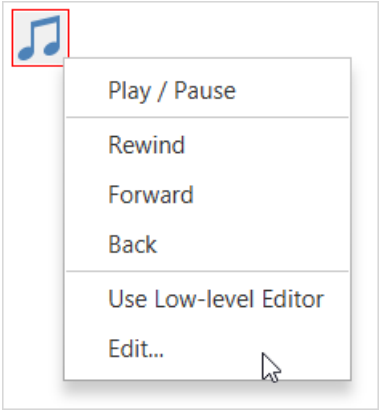
Figure 9.27. Two embedded media players, one for video and the other for audio, both having controls



This feature, which is based on JavaFX®, will not work if the Java™ runtime used to run **XXE** is older than version 1.8.0_40. Moreover, this feature may be explicitly disabled by unchecking **Options** → **Preferences**, **General**|**Feature**, "**In the styled view, render audio and video elements by embedding a media player**" [125].

A newly inserted media player is not configured yet (e.g. does not point to any media file) and just looks like a stock audio or video icon.

Figure 9.28. The right-click popup menu of a media player (audio only, no controls)



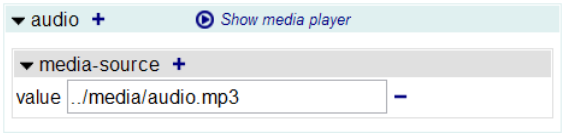
In order to configure a media player, you may either:

- double-click inside it;
- OR right-click in it and select the **Edit** menu item;
- OR drag and drop a media file onto it.

The three above actions all display the media player dialog box.

An alternate way to edit a media element

Selecting the "**Use Low-level Editor**" item in the right-click popup menu of a media player replaces the media player by an “object view” of the media element. For example:



This “object view” may be used as a low-level media element editor. It may be used for example to add `track` child elements to a `video` element, something that the media player dialog box cannot do (see below [181]).

The “plus button” lets you pick an attribute to be added from the list of available attributes[20] . The “minus button” removes the corresponding attribute.

When done with this low-level editor, the "**Show media player**" button may be used to switch back to the media player.

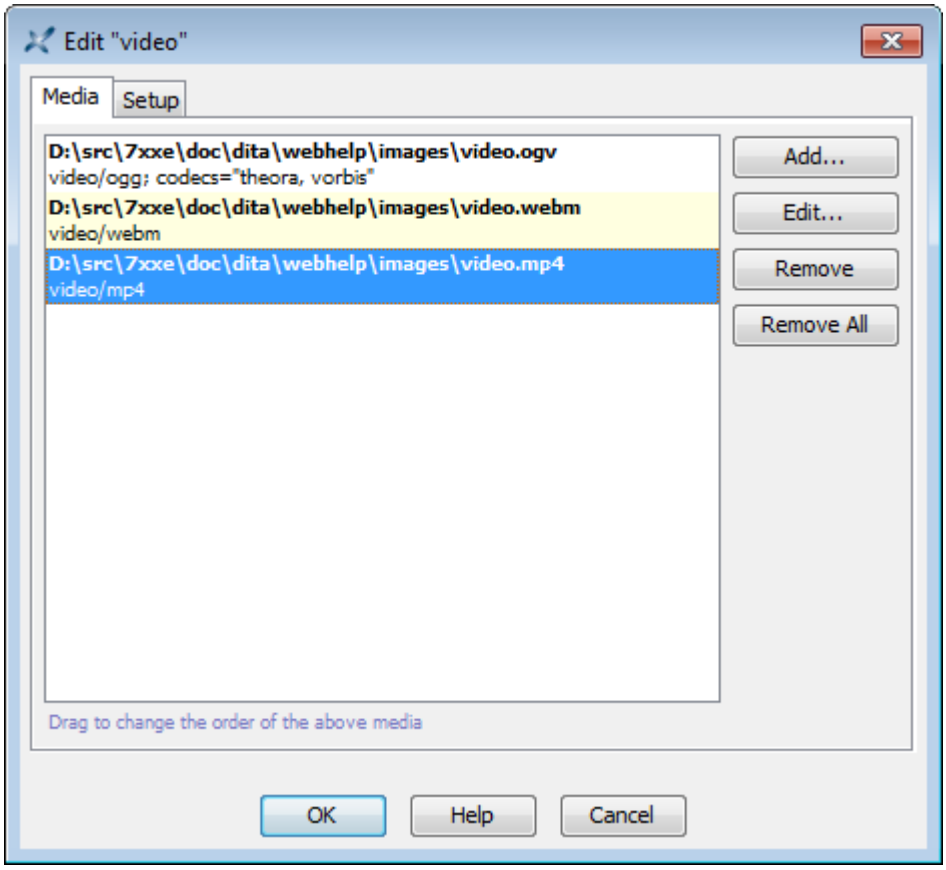
16.2. Using the media player dialog box

The dialog box has two tabs. First tab allows to specify one or more media files. The other tab allows to configure the media player.

[20] You may not see *all* the available attributes in this list. This list is filtered the same way the attribute table of the **Attributes** tool is. More information in Section 3.1.1, “Reducing the number of attributes displayed by the attribute table” [65].

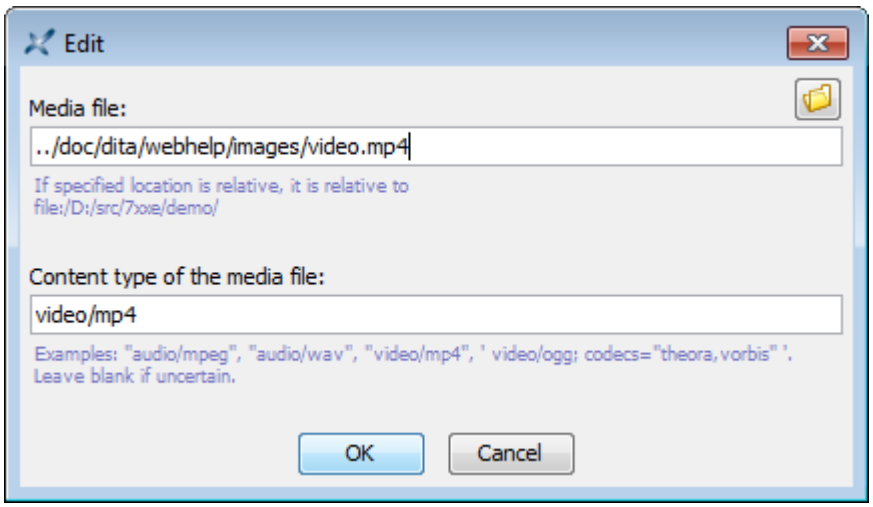
16.2.1. The Media tab

*Figure 9.29. The **Media** tab of the media player dialog box*



- Click **Add** to add a media file.
- Click **Edit** to modify selected media file.
- Click **Remove** to remove selected media file.
- Click **Remove All** to remove all media files.
- Use drag and drop to reorder the list of media files. The media file which is actually played by the embedded media player is the first one in the list having a supported format (see supported formats below [183]).

Clicking **Add** or **Edit** displays the following dialog box:



- In the "**Media file**" field, specify the absolute or relative *URL* of a media file. Make sure to specify a file having a content type matching the underlying media element. For example, do *not* specify a video file for a media player used to render an XHTML 5 `audio` element.
- In the "**Content type**" field, specify the precise content type of the media file. Leave blank if you are not sure of what this content type could be.

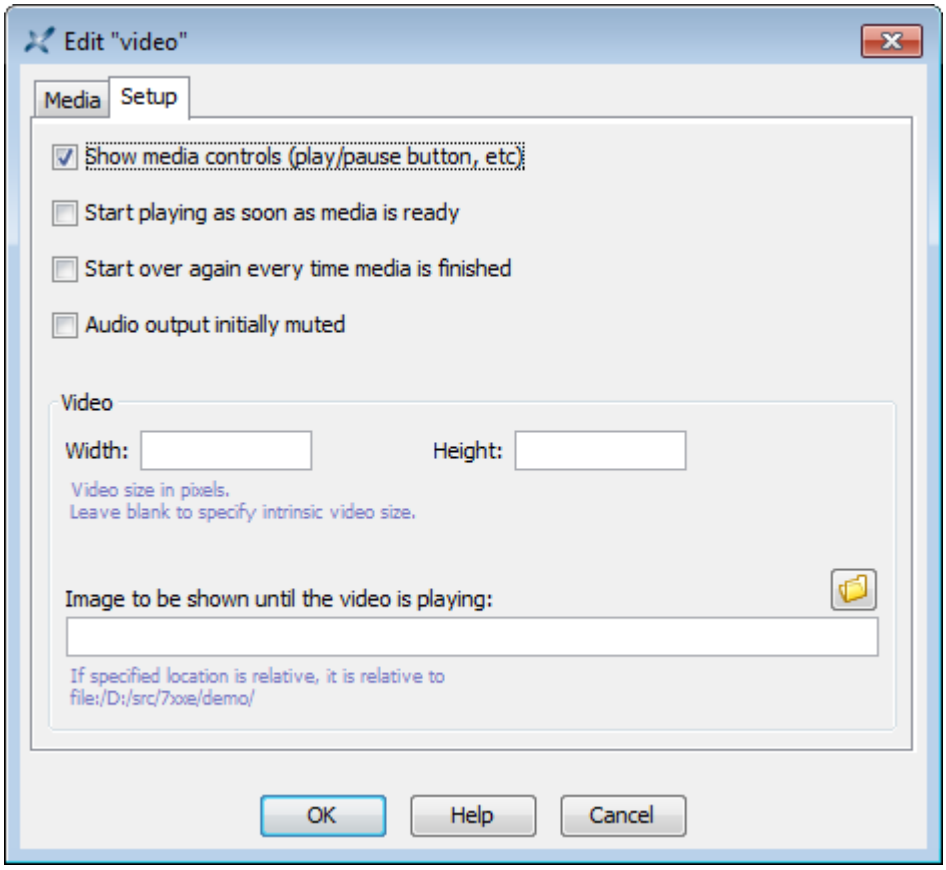
Specify alternate audio and video formats whenever possible

The embedded media player only supports playing few audio and video formats, mainly `.mp3` (`audio/mpeg`) , `.wav` (`audio/x-wav`), `.m4a` (`audio/x-m4a`), `.mp4` (`video/mp4`).

This should not prevent you from specifying alternate audio and video formats like `.ogg` (`audio/ogg`), `.webm` (`video/webm`), etc. Such formats may be the only ones supported by the application which will be used to open your deliverable (example: Apple iBook used to open the EPUB 3 book resulting from the conversion of a DITA document). So it is important to specify alternate audio and video formats whenever possible.

16.2.2. The Setup tab

*Figure 9.30. The **Setup** tab of the media player dialog box*



The following checkboxes allows to configure the embedded audio or video player:

- **Show media controls (play/pause button, etc)**. Corresponds to the the `controls` attribute of an XHTML 5 `audio` or `video` element.

  Note that when the media player has not been given any control button, it's still possible to use it by right-clicking in it and selecting the **Play**/**Pause**, **Rewind**, **Forward** and **Back** items found in the popup menu. See Figure 9.28, “The right-click popup menu of a media player (audio only, no controls)” [180].

- **Start playing as soon as media is ready**. Corresponds to the the `autoplay` attribute of an XHTML 5 `audio` or `video` element.

- **Start over again every time media is finished**. Corresponds to the the `loop` attribute of an XHTML 5 `audio` or `video` element.

- **Audio output initially muted**. Corresponds to the the `muted` attribute of an XHTML 5 `audio` or `video` element.

The **Video** frame is absent when the media player is used to render an audio element.

Width

Width of the embedded media player. Strictly positive integer expressed in pixels.

The default width and height are those of the media actually played by the media player, that is, the intrinsic video size.

The media player embedded in **XXE**'s styled view will always look too small on the HiDPI screen of a Linux computer.

As already explained, the media player is implemented using the JavaFX® technology and for now, JavaFX is HiDPI-aware only on Windows and on the Mac.

Height

Height of the embedded media player. Strictly positive integer expressed in pixels.

Note that it's possible to specify only the width of the media player or only its height or both its width and height.

Image to be shown until the video is playing

Corresponds to the `poster` attribute of an XHTML 5 `video` element. The absolute or relative URL of a PNG or JPEG image file.

Unless above **Width** and/or **Height** have been specified, the media player is given the size of this “poster image”.

17. The "Edit Image Map" dialog box

An image map is an image having “hot areas”. A “hot area” is a region of the image which behaves as a hyperlink to an internal or external destination.

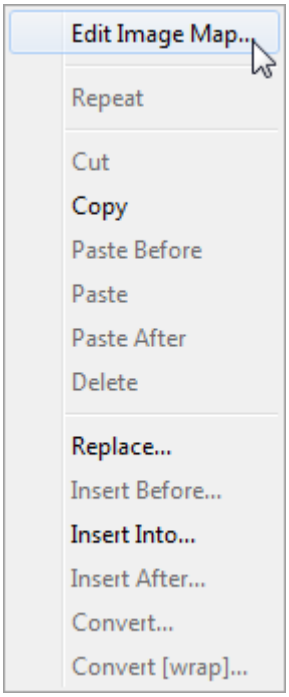
The following elements may be used to add image maps to your XML documents: XHTML `map`, DITA `imagemap`, DocBook `imageobjectco` (contained in a `mediaobject` or `inlinemediaobject` element).

17.1. Adding an image map to your document

The following procedure applies to XHTML, DITA topic and DocBook documents.

1. Select an entry corresponding to an image map from the popup menu displayed by the "**Add image**" toolbar button.
2. Specify which image file to use, for example, by right-clicking the image placeholder and then selecting "**Set Image**" from the contextual popup menu.
3. Right-click anywhere inside the newly inserted image map element element (e.g. XHTML `map`) and select "**Edit Image Map**" from the contextual popup menu.

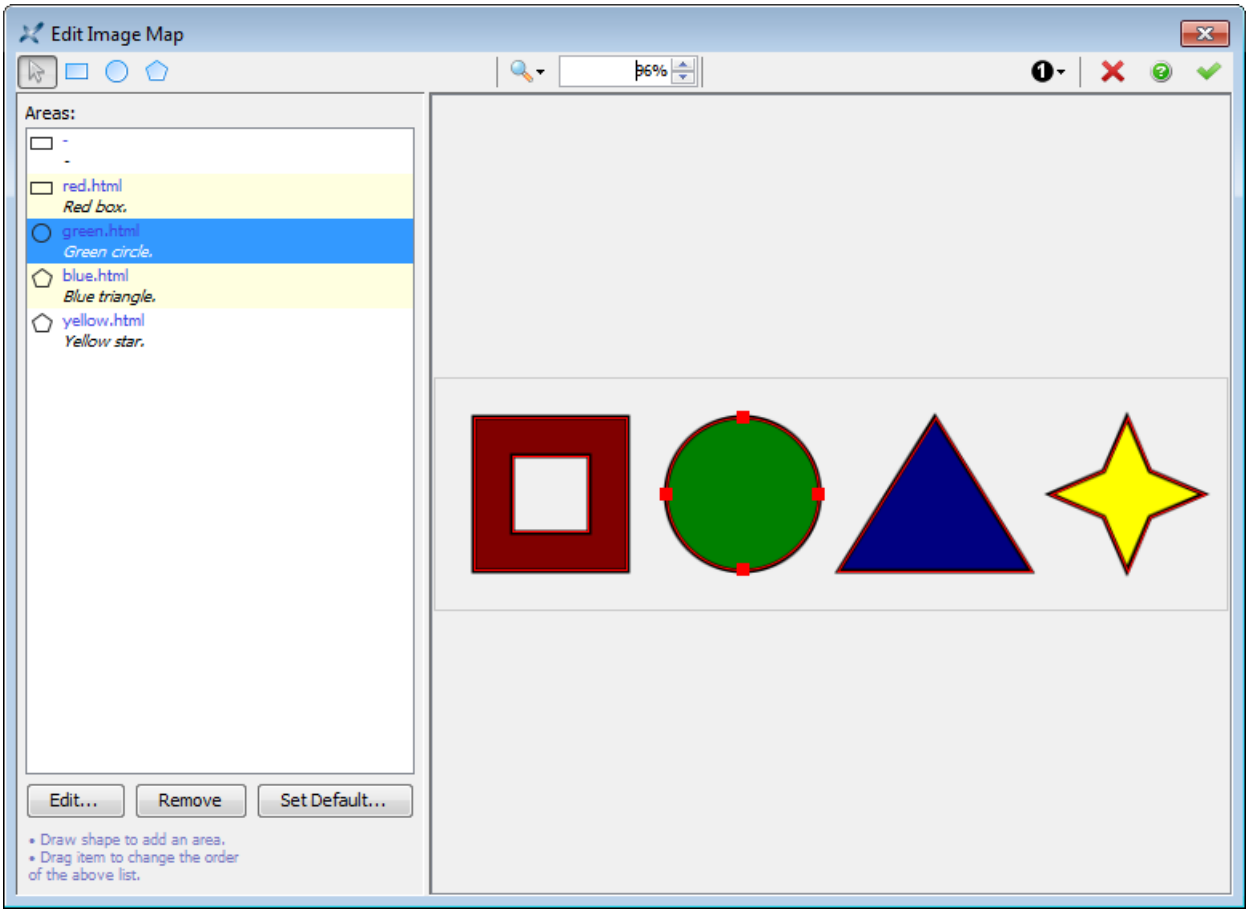
*Figure 9.31. The right-click, contextual, popup menu has an "**Edit Image Map**" entry*



This displays an image map editor.

4. Use the image map editor to add “hot areas” to your image.

*Figure 9.32. The "**Edit Image Map**" dialog box*



How to use this image map editor is explained below [187].

17.2. Using the image map editor

An image area is added as follows:

1. Click the **Rectangle**, **Circle** or **Polygon** button to select the corresponding drawing tool.

2. Draw a shape as you would do it in any vector graphics editor.

   Note that in the case of the **Polygon** tool, you'll have to right-click anywhere and select "**Finish**" in the contextual popup menu to close the polygon. More information in Section 17.2.1, “The toolbar of the image map editor” [188].

   Doing this adds an area to the **Areas** list found at the left of the image map editor. The newly added area is automatically selected. The **Select** tool is also automatically selected, that is, the current “drawing mode” is automatically abandoned.

3. Optionally adjust the position and size of the selected shape as you would do it in any vector graphics editor.

4. Optionally specify a link target for the new added area. This is done by clicking button **Edit** which displays the The "**Edit image map area**" dialog box. More information in Section 17.2.2, “Specifying the link target and optionally the link text of selected area” [189].

The order of areas in the Areas list is significant

- In case some areas overlap, it's important to know that:
  - areas are drawn in reverse order, from last area to first one;
  - when used as hyperlinks, areas are considered in order, from first one to last one.

    Example: in the case of the above figure, the first area is a transparent square found in the middle of the second area, a larger red square.

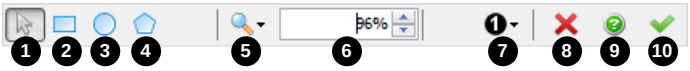
    First area, the transparent square, is a “dead area”, that is, not an hyperlink. Second area, the red square, is a “hot area”, an hyperlink pointing to `red.html`.

    Because the “dead area” is found before its enclosing “hot area” in the **Areas** list, clicking in the middle of the red square has no effect, while clicking in any red location will cause the `red.html` link to be followed.

    Switching the position of these dead and hot areas in the **Areas** list[21] would have to effect of making the dead area completely useless.

- The "**Set Default**" button may be used to specify the hyperlink, if any, associated to any part of the image which is *not* included in the areas found in the **Areas** list. This default area —the entire image— is always added at the very end of the **Areas** list. It does not make sense to add it elsewhere.

17.2.1. The toolbar of the image map editor



❶ Select, move or resize area.

❷ Add rectangular area.

Click on the first corner of the rectangle and drag to specify the opposite corner of the rectangle.

❸ Add circular area.

Click on the center of the circle and drag to specify the radius of the circle.

DocBook `imageobjectco` element only supports rectangular areas. Circular and polygonal areas are not supported.

❹ Add polygonal area.

Drag to draw at least two sides of the polygon then right-click anywhere and select "**Finish**" in the contextual popup menu to close the polygon.

DocBook `imageobjectco` element only supports rectangular areas. Circular and polygonal areas are not supported.

❺ Displays the **Zoom** menu.

❻ Displays and also allows to adjust the current zoom level.

[21] **Areas** list items may be moved by using drag and drop.

The zoom level may also be changed by using the mouse wheel while pressing the **Ctrl** key (Cmd key on the Mac).

❼ Displays the "**Create and use an annotated variant of image file**" menu. More information about this facility in Section 17.3, “Creating and using an annotated variant of image file” [190].

❽ The **Cancel** button.

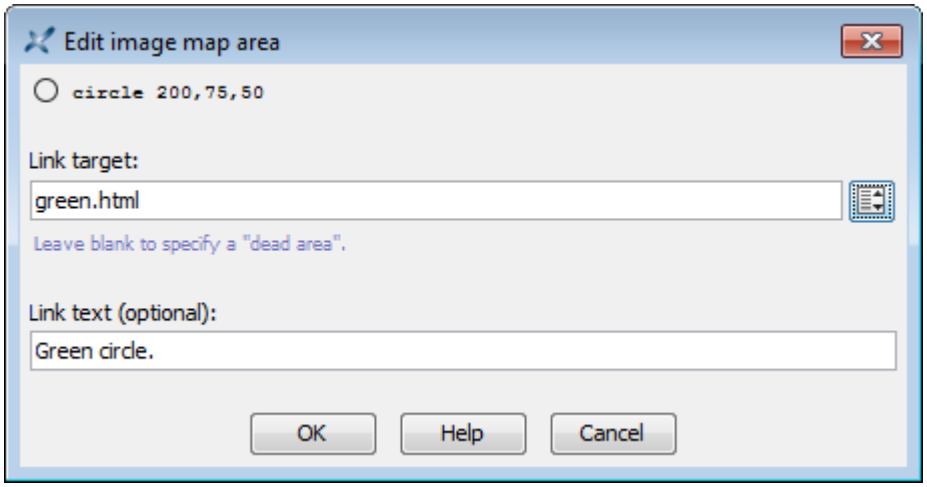
❾ The **Help** button.

❿ The **OK** button.

17.2.2. Specifying the link target and optionally the link text of selected area

The "**Edit image map area**" dialog box is displayed by clicking the **Edit** or "**Set Default**" buttons.

*Figure 9.33. The "**Edit image map area**" dialog box*



- The "**Link target**" field contains the target of the link. This link target may be a destination which is internal to the document being edited or external to it, typically a Web page.

  Click the "**Choose link target**" button to specify an internal link. This button displays the same dialog box [103] as the "**Set Link Target**" entry found in the contextual popup menu after right-clicking an hypertext element (e.g. XHTML `a`).

  Drag a link location from a Web browser and drop it into the "**Link target**" field to specify an external link.

  DocBook 4 `imageobjectco` element only supports internal links.

  DocBook has a facility —**DocBook** → **Link callouts** in *XMLmind XML Editor - DocBook Support*— allowing to automatically link areas and items found in a `calloutlist` element. With this facility, there is no need to use the "**Edit image map area**" dialog box.

- The "**Link text**" field contains an optional description of the link target. This description is ignored unless a link target has been specified.

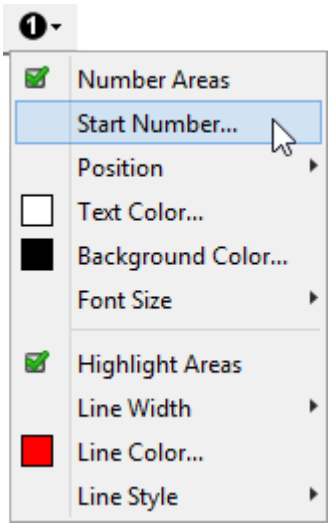
  DocBook 4 `imageobjectco` element does not support link text.

17.3. Creating and using an annotated variant of image file

Normally, the “hot areas” drawn using the image map editor are not visible to the user. For example, if you convert a DITA `imagemap` to XHTML, the Web browser will simply display the image as is. The user will understand that the image has “hot areas” only when she/he'll hover the mouse pointer above these “hot areas”.

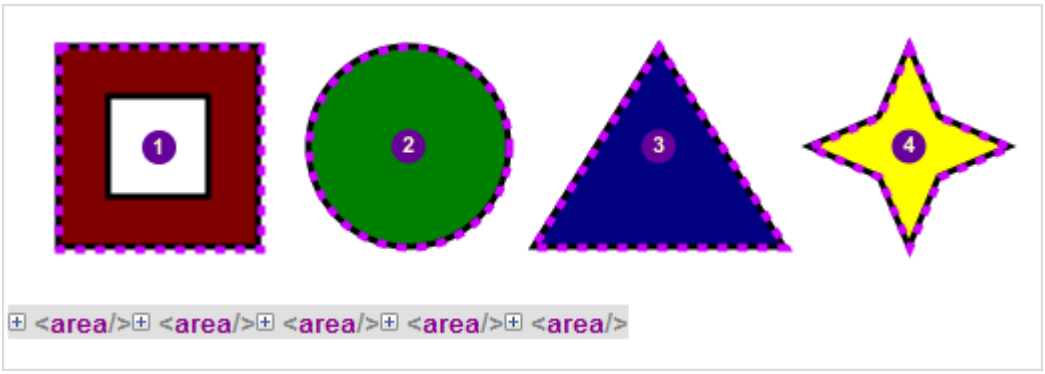
However, in some cases, making the “hot areas” visible and/or giving them numbers so they can be referred to by number may be useful. This can be done by clicking the "**Create and use an annotated variant of image file**" button and checking the "**Highlight Areas**" and/or "**Number Areas**" checkboxes.

*Figure 9.34. The popup menu displayed by clicking the "**Create and use an annotated variant of image file**" toolbar button*



The other entries of the above menu let the user customize the appearance of the visual marks she/he has chosen to add to the image. In the following figure, the user has chosen yellow on purple numbers using a 9pt font size and thick, dashed, purple highlights.

Figure 9.35. Example of annotated image preceding an XHTML `map`



Note that these visual marks are not rendered by the image map editor itself, as this would make drawing new shapes more difficult. Instead, it works as follows: after checking the "**Highlight Areas**" and/or "**Number Areas**" checkboxes and clicking **OK**, an SVG image embedding the original image and adding the chosen visual marks above this image is automatically generated.

For example, if a DITA `imagemap` contains `<image href="images/imagemapworld.jpg">`, this facility automatically generates SVG file "`imagemapworld.jpg.marked.svg`" and replaces the value of the `href` attribute by `images/imagemapworld.jpg.marked.svg`.

Note that the format of the image to be annotated must be GIF, PNG, JPEG or SVG, otherwise the "**Create and use an annotated variant of image file**" toolbar button is disabled (grayed).

Image annotations are created *only for areas having link targets*

"Dead areas", that is, areas having no link targets, are ignored by this facility.

Therefore, if you use tool bar button "**Add image**" in *XMLmind XML Editor - DocBook Support* and select "**mediaobject(calloutlist)**" to add a `mediaobject` element containing `imageobjectco` element to your document, you must proceed as follows:

1. Specify an image file, for example, by right-clicking the image placeholder and then selecting "**Set Image**".

2. Right-click anywhere inside the newly inserted `imageobjectco` element and select "**Edit Image Map**".

3. Use the image map editor to add "hot areas" to your image. *Do not bother setting the links of any of these hot areas* because there is a way to do this automatically.

4. Add one `callout` per hot area to the `calloutlist`. The number and order of the callouts must match the number and order of the hot areas because this is how the correspondence between a hot area and a `callout` is established. *Do not bother setting the links of any of these callouts* because there is a way to do this automatically.

5. Use **DocBook** → **Link callouts** in *XMLmind XML Editor - DocBook Support* to link the hot areas to the corresponding `callout` elements (and the other way round of course).

6. Now that the **DocBook** → **Link callouts** command has automatically added link targets to all areas, you may use the "**Create and use an annotated variant of image file**" facility.

Appendix A. Command line options

1. Command line

```
xxe [advanced_option]*
    [ [ -read ] file_or_URL_to_be_opened ]* [ -compare ]
    | -last
    | -new config template save_file_or_URL
    | -new save_file_or_URL
```

Options:

file_or_URL_to_be_opened

Open specified document in normal read/write mode.

If the "**Enable the 'File|Folder' Submenu**" [125] feature has been enabled, *file_or_URL_to_be_opened* may also specify the location of a folder or a Zip archive (of any kind: `.zip`, `.jar`, `.epub`, `.docx`, `.xlsx`, `.odt`, etc). When this is the case, a "**Browse Files**" tool [72] is opened to allow browsing the contents of this folder or Zip archive.

Examples:

```
xxe doc.xml
xxe .
xxe C:\Users\john
xxe zipped/docsrc.zip
xxe report.docx
```

`-read` *file_or_URL_to_be_opened*

Open specified document in read-only mode.

Ignored when *file_or_URL_to_be_opened* specifies the location of a folder or a Zip archive.

`-compare`

This option must be specified just after at least two *file_or_URL_to_be_opened* arguments. This option automates the comparison of two revisions of the same initial document for which the comparison of revisions has been enabled. (Background information about the compare revisions feature: the **Tools** → **Revisions** submenu [40], the **Compare** tool [85].)

Examples:

```
xxe doc.xml~ doc.xml -compare
xxe -read ../old_docs/art23.xml art23.xml -compare
xxe art23.xml -read ../old_docs/art23.xml -compare
```

`-last`

Forces **XXE** to reopen the last document opened during the preceding editing session. Note that this option cannot be used when one or more *file_or_URL_to_be_opened*s have been specified.

See also the "Automatically reopen last opened documents [123]" option for a more comprehensive alternative.

`-new` *`save_file_or_URL`*

Creates a new document having specified filename or URL.

Note that this option does not create the corresponding save file. You'll still have to use **File** → **Save** or **File** → **Save As** to actually save the newly created document. However, when this option is used, **File** → **Save** or **File** → **Save As** will display a file chooser dialog box properly initialized using *`save_file_or_URL`*.

The document template used to create the new document is determined as follows:

- If the file extension of *`save_file_or_URL`* is used only for a single document template (e.g. "`.ditaval`") then this document template is used to create the new document.
- Otherwise, if the file extension of *`save_file_or_URL`* does not correspond to any template (e.g. "`.docb`") or corresponds to several templates (e.g. "`.xml`") then the **File** → **New** dialog box is displayed to let the user choose a document template.

Examples:

```
-new ../common/filter.ditaval

-new doc.xml
```

Advanced options:

`-putpref` *`key value`*

Adds or replace preference specified by *`key`*/*`value`* to the set of the user's preferences.

If *`value`* starts with "`<?xml?>`", then this prefix is removed and XML character entities are unescaped in *`value`*. This allows to easily specify preference values containing “unusual characters”. For example `-putpref MyKey "<?xml?>Line 1
Line 2"` adds preference key `MyKey` with a value equals to "`Line 1`" and "`Line 2`" separated by a newline character ("`
`").

The set of the user's preferences is normally modified using the Preferences dialog box [106].

For example, in the case of the Encoding [109] option, the preference key is `encoding`.

```
xxe -putpref encoding Windows-1252
```

The reference of all preference keys is found below [194].

`-putprefs` *`property_file_or_URL`*

Similar to `-putpref` except that several key/value pairs may be read from specified property file. Example:

```
xxe -putprefs /etc/xxe/preferences.properties
```

`-delpref` *`key`*

Removes preference specified by *`key`* from the set of the user's preferences.

`-auth` credentials

This option can be used to specify authentication credentials for a given server. This allows to connect to the specified server without interactively asking the user to enter a username and a password.

String *credentials* consists in 6 fields: *host*, *port*, *prompt*, *scheme*, *username*, *password*, in that order, separated by a newline character ('\n'). Fields *host*, *port*, *prompt*, *scheme* can be left empty, which means: match any. The UTF-8 bytes of the string are then encoded in base-64.

Command-line utility *XXE_install_dir*/bin/authvalue allows to generate such encoded string. Example: encode string "\n\nDocument Store\n\nvictoria\nsecret":

```
/opt/xxe/bin$ java -cp xxe.jar com.xmlmind.netutil.SimpleAuthenticatorModule \
victoria secret - "Document Store"
CgpEb2N1bWVudCBTdG9yZQoKanZpY3RvcmlhCnNlY3JldA==

/opt/xxe/bin$ xxe -auth CgpEb2N1bWVudCBTdG9yZQoKanZpY3RvcmlhCnNlY3JldA== \
http://www.acme.com/docstore/push_up.xml &
```

Command-line utility **authvalue** is auto-documented. Type **authvalue**, then press Enter to display a short documentation explaining how to use this utility.

-open *file_or_URL_to_be_opened*, -print *file_or_URL_to_be_opened*

First *file_or_URL_to_be_opened* may be preceded by -open or -print, which are ignored. This may be useful when XXE is started by Java™ Web Start. Example: "javaws http://www.acme.com/xxe/xxe.jnlp -open /docs/doc.xml".

2. Preference keys

Key	Type	Default	Corresponding Option
activeInputMethodClient	boolean	false	Use integrated input method support [113]
addOpenLines	boolean	true	Add open lines [110]
addonSource	servers|directory|both	servers	Download add-ons from these servers [122]
addonListURLs	list of URLs separated by newline characters ('\n')	See Download add-ons from these servers [122].	When addonSource=servers, this key specifies the list of "servers". See Download add-ons from these servers [122].
addonDir	the filename of an existing directory	none	When addonSource=directory, this key specifies the local directory. See Search add-ons in this directory [123].
allowAdvancedXInclude	boolean	false	Allow advanced use of XInclude [114]
appendSuggestionMode	none, auto or manual	manual	Append mode [113]
asyncImageLoading	no, yes or remote	remote	Load images in a background task [118]

Key	Type	Default	Corresponding Option
`asyncLabelLoading`	`no`, `yes` or `remote`	`remote`	When rendering a DITA map stored on a remote drive, the titles of the topics referenced by this map are loaded asynchronously by default. This considerably speeds up[a] opening a DITA map stored on a slow remote drive such as the Google Drive™. This feature is controlled by the user preference called `asyncLabelLoading`. There is no user interface which may be used to change the value of this user preference. If you need to change its value, please use **xxe**'s command-line options `-putpref` and `-delpref` [193]. Example: start **XXE** using "**`xxe -putpref asyncLabelLoading yes`**".
`autoCheckForUpdates`	boolean	true	Check for Updates [49]
`autoCheckForUpdates.interval`	positive integer	86400 (24 hours)	Specifies the minimal amount of time, in seconds, between two automated checks. See Check for Updates [49].
`autoCheckForUpdates.after`	integer between 10 and 14400	120 (2 minutes)	Specifies the amount of time, in seconds, after which the automated check is started. See Check for Updates [49].
`autoDiffSupport`	false|true|revisions	false	When a document is created or opened: [120]
`autoMasterDocument`	boolean	true	**Use as Master Document** [36]. See also **Maps (e.g. DITA map,**

Key	Type	Default	Corresponding Option
			DocBook assembly) are automatically made master documents [119].
`autoReopenDocuments`	boolean	`false`	**Automatically reopen last opened documents** [123]
`autoReopenState`	string	`none`	**Automatically reopen last opened documents** [123]
`autoSaveInterval`	`-1000000`-`1000000` (number of operations; negative means disabled)	-300	Max. modifications before saving [110]
`autoSaveTimeout`	10-86400 (number of seconds)	30	Idle time (seconds) before saving [111]
`autoSelectAttribute`	`never`, `placeholder` or `required`	`placeholder`	Automatically select an attribute [114]
`autoShowContentModel`	boolean	false	Corresponds to the auto-update option of the dialog box displayed by **Help** → **Show Element Reference** [49] menu item. This option automatically updates the dialog box to always show the content model of selected element.
`autoShowValidityPane`	enumeration(`NONE`, `SEMANTIC_WARNING`, `SEMANTIC_ERROR`, `INVALID_REFERENCE`, `INVALID_DATA`, `INVALID_STRUCTURE`)	INVALID_DATA	**Automatically show Validity tool** [119]
`autoSpellCheck`	enumeration (`never`, `always`, `seeConfig`)	`seeConfig`	Activate if this is specified in the configuration file [120]
`button2PastesSystemSelection`	boolean	`false`	Clicking with middle button pastes system selection [113]
`charsSavedAsEntityRefs`	specification as in In addition to characters outside the encoding, save the following	empty string	In addition to characters outside the encoding, save the following characters as entity references [109]

Key	Type	Default	Corresponding Option
	characters as entity references [109]		
`copyDocumentTemplate`	boolean	`false`	Immediately save newly created document [106]
`cursiveFontFamily`	font family name	`Serif`	"cursive" font family [117]
`defaultBackground`	3 0-255 integers separated by spaces (specify red, green, blue)	255 255 255	Default background [117]
`defaultFontIsSerif`	boolean	`false`	Default font family [117]
`defaultFontSize`	8-24 (CSS points)	11	Default font size [117]
`defaultForeground`	3 0-255 integers separated by spaces (specify red, green, blue)	0 0 0	Default text color [117]
`displayScaling`	100-400 (%) or -1 which means: use system settings	-1	**Display scaling** [123]
`documentCache`	integer between 0 and 5000. Specify 0 to disable the cache.	`100`	**Document cache** [127]
`dontIndentUnconstrainedDocs`	boolean	`false`	Do not indent documents not conforming to a DTD or schema [110]
`editPane.inspectorMode`	boolean	`true`	When the inspector mode is turned on (which is the default), the **Edit** tool does not clear its state when the editing context is changed. On the contrary, it automatically updates its list to continuously show you what elements are allowed given the current editing context.
`encoding`	any encoding supported by Java™ or "`ORIGINAL_ENCODING`"	UTF-8	Encoding [109]
`ensureSaveFileHasExtension`	boolean	`true`	Ensure that a save filename has an extension [111]

Key	Type	Default	Corresponding Option
`fantasyFontFamily`	font family name	`Serif`	"fantasy" font family [117]
`filterDuplicateIDs`	boolean	`true`	**Filter duplicate ID errors found in modular documents** [119]
`ftpProxyHost`	host name or host address	none	**FTP proxy** [129]
`ftpProxyPort`	integer	80	(FTP proxy) **Port** [129]
`guessIgnorableWhitespace`	boolean	`true`	When no DTD or schema, guess ignorable white space [106]
`headerBegin` [`footerBegin`]	text	empty [`%F`]	Begin [112]
`headerColor` [`footerColor`]	3 0-255 integers separated by spaces (specify red, green, blue)	128 128 128 [128 128 128]	Color [112]
`headerEnd` [`footerEnd`]	text	empty [`%P`]	End [112]
`headerMiddle` [`footerMiddle`]	text	empty [empty]	Middle [112]

`helperApplications.entries`	string
	The format of this string is:
	`entries --> ( entry ('\n' entry)* )?`
	
	`entry --> mime_types ':' extensions ':'`
	`magic_strings ':'`
	`xml_name_patterns ':'`
	`command`
	
	`mime_types --> ( mime_type (',' mime_type)* )?`
	`extensions --> ( extension (',' extension)* )?`
	`magic_strings --> ( magic_string (',' magic_string)* )?`
	`xml_name_patterns --> ( xml_name_pattern (',' xml_name_pattern)* )?`
	
	`In any of the above fields,`
	`character ':' must be escaped using "\072"`
	`and character ',' must be escaped using "\054".`
	
	`magic_string --> HEXADECIMAL_NUMBER`
	
	`xml_name_pattern --> ( '{' namespace_URI? '}' )? local_part`
	`One of local_part or namespace_URI`
	`may be equal to "*".`
		see File types [121]	File types [121]

Key	Type	Default	Corresponding Option
`helperApplications.defaultViewer`	string	see Default viewer [121]	Default viewer [121]
`highlightAllHits`	boolean	`false`	Highlight all [71]
`horizontalSplit`	boolean	`false`	**Split Document Area Horizontally** [48]
`httpProxyHost`	host name or host address	none	**HTTP proxy** [129]
`httpProxyPort`	integer	80	(HTTP proxy) **Port** [129]
`httpsProxyHost`	host name or host address	none	**HTTPS proxy** [129]
`httpsProxyPort`	integer	443	(HTTPS proxy) **Port** [129]
`indent`	integer (negative means not indented)	2	Indentation [110]
`ignoreStyleSheetPI`	boolean	`false`	Ignore <?xml-stylesheet?> [107]
`infoAboutReadOnlyInclusions`	boolean	`true`	Inform about non-editable document parts [107]
`locale`	*`language`* (e.g. `fr`) or *`language_COUNTRY`* (e.g. `fr_CA`) or - (means default)	-	**Locale** [124]
`lockLocalDocuments`	boolean	`false`	Lock documents stored on the local filesystem [107]
`lockMode`	`shared`, `exclusive` or `none`	`exclusive`	Preferred lock mode [108]
`lockOwner`	string	empty string which implies *`user_name@host_name`*	Identifier as a lock owner [108]
`lockTimeout`	integer (number of seconds)	86400 (that is, 24 hours)	After specified number of seconds, a lock is to be automatically removed by the server (typically WebDAV) hosting the locked document. A negative or null value may be used to create locks which should never expire.

Key	Type	Default	Corresponding Option
			Note that there is no need to create permanent locks as XXE automatically “refreshes” the lock each time the locked document is being saved. One notable exception is Subversion, which when used through WebDAV, does not support locks having a finite lifetime. This user preference cannot be specified using the GUI of XXE. You need to use the `-putpref` command-line option [193] for that.
`lookAndFeelClassName`	Java™ class name of a Look&Feel or "`-`" (an alias for "`default`") or "`default`" (the FlatLaf Look&Feel, light theme, on Linux and the system Look&Feel on the other platforms) or "`system`" (the cross-platform Look&Feel —AKA Metal— on Linux and the system Look&Feel on the other platforms)	-	**Style** [124]
`makeBackupFiles`	boolean	`true`	Before saving, make a backup copy of the file [111]
`masterDocumentList`	list of document URLs separated by newline characters	-	**Use as Master Document** [36]. See also Except for the following maps [119].
`maxLineLength`	positive integer	78	Max. line length [110]
`maxUndo`	1-100	20	Max. undo actions [114]
`monospaceFontFamily`	font family name	`Monospaced`	"monospace" font family [117]

Key	Type	Default	Corresponding Option
nonMasterDocumentList	list of document URLs separated by newline characters	-	**Use as Master Document** [36]. See also **Master documents** [119].
nonProxyHosts	list of host names, host addresses and domain names (e.g. .acme.com) separated by spaces	"localhost 127.0.0.1"	**No proxy for** [130]
onlyShowSuggestions	boolean	false	In the choice list, only show suggestions [114]
proxySettings	none|system|application	system	**Use system settings** [128]
quickStartCache	boolean	true	**Quick Start cache** [127]
saveCharsAsEntityRefs	boolean	true	Save characters outside encoding as entity references [109]
schemaCache	integer between 0 and 100. Specify 0 to disable the cache.	10	**Schema cache** [127]
sansSerifFontFamily	font family name	SansSerif	"sans-serif" font family [117]
serifFontFamily	font family name	Serif	"serif" font family [117]
singleInstance	boolean	false	**Use a unique instance of XMLmind XML Editor** [123]
singleInstancePort	1-65535	49987	**Port** [123]
socksPassword	*encoded* string	the empty string	(SOCKS) **Password** [130]
socksProxyAuthentication	boolean	false	**Authenticate SOCKS user** [129]
socksProxyHost	host name or host address	none	**SOCKS proxy** [129]
socksProxyPort	integer	1080	(SOCKS proxy) **Port** [129]
socksUser	string	the login name of the user	(SOCKS) **User name** [129]
sortDiagnosticsBySeverity	boolean	false	If true, instructs the **Validity** tool [71] to list errors from most severe ones to least severe ones. The default

Key	Type	Default	Corresponding Option
			is to list errors in document order.
`treeViewFontSize`	8-24 (CSS points)	10	Base font size [118]
`treeViewFontsAndColors`	(without a space after ';') "*`markupFont`*`;` *`textFont`*`;` *`verbatimTextFont`*`;` *`backgroundColor`*`;` *`readOnlyColor`*`;` *`markupColor`*`;` *`elementNameColor`*`;` *`attributeNameColor`*`;` *`attributeValueColor`*`;` *`textColor`*`;` *`commentColor`*`;` *`piColor`*`;` *`textBorderColor`*`;` *`commentBorderColor`*`;` *`piBorderColor`*"	(without a space after ';') "`SansSerif-Bold; SansSerif; Monospaced; ffffff; e0f0f0; 808080; 8b008b; 008b8b; 00008b; 000000; 8b1c62; 006400; f0f0f0; ffe0e0; e0ffe0`"	Fonts and colors [118]
`underlineHeader` [`overlineFooter`]	boolean	`true`	Underline [Overline] [112]
`unselectedFeatures`	list of feature names (`OpenFolder`, `NewWindow`, etc), separated by newline characters	`ForceDeletion`, `IncludeTool`, `DeveloperTools`, `DocumentCacheTool`.	Section 6.10.1, "Features Options" [124]
`updateInclusionsOnSave`	boolean	`true`	Automatically update references in modular documents [111]
`useHelperDocumentType`	boolean	`true`	When no DTD or schema, simulate a DTD [107]
`useNativeFileChooser`	boolean	`true` on the Mac, `false` on the other platforms	Use the native file chooser in preference to the multi-platform file chooser [106]
`useURLChooser`	boolean	`false`	**Use the URL Chooser** [15]
`viewSettings`	string	none	Preference saved by the items of the **View** [32] menu. See also Section 35, "viewSettings" in *XMLmind XML Editor - Configuration and Deployment*.

	The format of this string is:
	`entries --> entry [ '\n' entry ]*`
	`entry --> document_URL '\n' view_settings`
	`view_settings --> center_pane '|' top_pane '|' right_pane '|'`		

Key	Type	Default	Corresponding Option
	bottom_pane '|' left_pane pane --> stylesheet_name|'-' ';' size ';' add_order ';' text_size ';' display_images ';' show_tags ';' tree_view_details size --> DOUBLE_BETWEEN_0_AND_1 *(1 for center_pane)* add_order --> POSITIVE_INTEGER *(0 for center_pane)* text_size --> INTEGER_BETWEEN_8_AND_24 display_images --> image|thumbnail|box show_tags --> no|yes|all tree_view_details --> POSITIVE_INTEGER_IN_HEXADECIMAL_FORMAT No *stylesheet_name* at all means: hidden pane. "-" means: tree view. Example: "file:/home/john/doc.xml\nDocBook;1.0;0;12;box;no;|;;;;;;|;;;;;;|;;;;;;|;;;;;;".		
warnAboutAdvancedXInclude	boolean	false	Warn about advanced use of XInclude [114]
wrapLongWords	boolean	false	Wrap words wider than available space [116]
wrapSearch	boolean	false	Wrap search [71]

[a]Provided that the document cache is given a sufficient capacity. See "**Document cache size**" in the **Options** → **Preferences**, **Advanced**|**Cached data** preferences form [126].

3. Environment variables

The following environment variables are convenient to use while customizing or extending **XXE**. However, these variables are not really meant to be used in production.

Notes:

- All the scripts used to start **XXE** (that is, `xxe`, `xxe-c.bat` and `xxe.jstart`, `xxeconvert`, `xxeconvert.bat`) automatically define *system properties* corresponding to the following environment variables. Therefore it's also possible to specify the following variables as system properties rather than environment variables. See other supported system properties below [204].
- You cannot use the following environment variables on the Mac[1] due to limitations in Apple's Java™ launcher.

Variable name	Value	Description
XXE_ADDON_PATH	List of directory filenames separated by semi-colons (';').	Do not forget to clear, or even disable, the Quick Start cache [127] before using XXE_ADDON_PATH.

[1]This is the case only with `XMLmind.app`, the application bundle contained in the `.dmg` distribution. If you are a local guru or a consultant you may prefer to download and install the `xxe-*.zip` distribution rather than the `xxe-*.dmg` distribution.

After unzipping this archive in a directory of your choice, XMLmind XML Editor may be started using the *XXE_install_dir*`/bin/xxe` shell script. Unlike `XMLmind.app` which leverages Apple's Java™ launcher, the `xxe` shell script makes it easy working with environment variables such as `XXE_GUI`, `XXE_ADDON_PATH`, etc.

Variable name	Value	Description

		All the directories referenced in this list are recursively scanned by XXE during its startup to dynamically discover add-ons.
		
		Linux example:
		
		`~$ XXE_ADDON_PATH="my_docbook;+" xxetool convert \`
		`docb.toHTML doc.xml -u out`
		
		More info in chapter "The lookup phase during XXE startup" of Section 1, “Dynamic discovery of add-ons” in *XMLmind XML Editor - Configuration and Deployment*.

`XXE_FOP_CONFIG`	URL or absolute file path of a user-defined FOP configuration file	Specifies the location of a FOP configuration file. More information in Using in **XXE** an existing FOP configuration file [132].
`XXE_PREFS_DIR`	Relative or absolute filename of a directory. A relative filename is relative to the current working directory.	Specifies a custom user preferences directory. This directory is created if it does not already exist. Linux example: `~$ XXE_PREFS_DIR=test xxe &`
`XXE_VIEWPORT_SCROLL_MODE`	`simple|backing-store|blit`	Specifies which scrolling mode to use everywhere in **XXE**. Linux example: `~$ XXE_VIEWPORT_SCROLL_MODE=simple xxe &`
`XXE_XEP_CONFIG`	URL or absolute file path of a user-defined XEP configuration file	Specifies the location of a XEP configuration file. More information in Using in **XXE** an existing XEP configuration file [133].

4. System properties

In principle, there is no need to use the following, very low-level, system properties.

Property name	Value	Description
XSL_FO_PROCESSORS	List of XSL-FO processor names separated by semi-colons (';').	This property contains the names of all XSL-FO processor plug-ins. Examples: "`FOP;XFC`", "`XEP`". For use by some XSLT style sheets.
XXE_ADDON_DOWNLOAD_LOCATION	List of "`.xxe_addon`" URLs separated by semi-colons (';').	Clicking on the **Reset** button of the "**Install add-on**" preferences of the **Preferences** dialog box resets the list of download URLs to the value specified by this property.

Property name	Value	Description
		If this property is not set, by default, the list of download URLs is: • http://www.xmlmind.com/xmleditor/_usercontrib/list.xxe_addon • `http://www.xmlmind.com/xmleditor/_download/list-${XXE_VERSION}.xxe_addon` (for example, variable `${XXE_VERSION}` is replaced by "`10_8_1`" if the version of the running XML Editor is v10.8.1) • `http://www.xmlmind.com/archive/xmleditor/${XXE.VERSION}/list-${XXE_VERSION}.xxe_addon`
XXE_CONFIRM_CREDENTIALS_DELAY	Strictly positive number of milliseconds	Workaround for the following problem: "when opening a document stored on a server requiring user authentication (e.g. a WebDAV server), a user who typed an incorrect username or password is never prompted again for her/his credentials". Example: `-DXXE_CONFIRM_CREDENTIALS_DELAY=1000`, means wait for at least 1000 milliseconds before considering that the credentials specified by the user have been accepted by the server. When system property `XXE_CONFIRM_CREDENTIALS_DELAY` has been specified to an appropriate number of milliseconds, a user who typed an incorrect username or password is automatically prompted again for her/his credentials. However, this workaround, based on a heuristic, may be tricky to configure and may be not usable for some protocols. For example, it seems to work for `HTTP` and it gets in the way for `FTP`.
XXE_CLIENT	Any (example: `1`; suffice to *set* this property).	Set by **xxeserver** to specify that it's XMLmind XML Editor Web Edition and not the desktop application which is running.
XXE_DAV_TRACE	Any (example: `1`; suffice to *set* this property).	Causes the WebDAV client to print DAV requests and responses on the console.
XXE_DAV_USE_SIMPLE_REFS	Any (example: `1`; suffice to *set* this property).	Causes the WebDAV client to use absolute paths rather than full URIs in the `Destination` and `If` headers. Doing this is allowed by RFC 4918, but not by (obsolete) RFC 2518. This may be useful when operating through a reverse proxy that does rewrite the `Host` request header, but not WebDAV-specific headers.

Property name	Value	Description
XXE_DAV_URI_ENCODING_USES_8859_1	Any (example: `1`; suffice to *set* this property).	Used by the WebDAV client. Specifies that the WebDAV server expects and returns URIs where accented characters (to make it simple) are escaped using the customary %*HH* form but where *HH* is the hexadecimal code of an ISO-8859-1 character. The standard mandates *HH* to be UTF-8 codes, not ISO-8859-1 codes. For example, this flag is needed for Apache/mod_dav running on an “old” Linux server where the file system encodes filenames using ISO-8859-1.

XXE_FOP_CONFIG	URL or absolute file path of a user-defined FOP configuration file	Specifies the location of a FOP configuration file. More information in Using in **XXE** an existing FOP configuration file [132].
		
		Note that, because this URL is resolved using the XML catalogs of **XXE**, this allows to bundle a custom FOP configuration file with the FOP plug-in, prior to deploying **XXE** using Java™ Web Start.
		
		Procedure:
		
		1. Create the custom FOP configuration file in *FOP_plugin_dir*`/conf/fop.conf`.
		2. Create XML catalog `fop_catalog.xml` in *FOP_plugin_dir*`/`. This file should contain:
		
		`<catalog`
		`  xmlns="urn:oasis:names:tc:entity:¬`
		`xmlns:xml:catalog">`
		`  <rewriteURI uriStartString="`**`fop-plugin:`**`"`
		`  rewritePrefix="."/>`
		`</catalog>`

XXE_JAVA_IMAGE_TOOLKIT_EXCLUDE	List of file extensions separated by semi-colons (';').	Example: `-DXXE_JAVA_IMAGE_TOOLKIT_EXCLUDE="bmp;wbmp"` This would prevent the built-in Java™ 1.5 image toolkit from handing BMP images. This is useful in case you find an external image toolkit which handles specified image extensions better than the built-in Java™ image toolkit.

Property name	Value	Description
XXE_NETWORK_TIMEOUT	A number of milliseconds.	Specifies the default timeout used for creating network connections and reading responses. A negative or null number means: wait indefinitely. Used at various places in XMLmind XML Editor, for example by WebDAV and FTP virtual drives. Built-in default value is 120,000ms (2mn).
XXE_SHOW_CLIPBOARD_ERRORS	Any value. Suffice for this system property to be set.	Causes **XXE** to report system clipboard errors on the console.
XXE_SHOW_SEMANTIC_ERRORS	Any value. Suffice for this system property to be set.	Allows to make cross-reference errors less important than semantic warnings and errors (which are typically reported by Schematron validation). This has an influence of the **Validity icon** found at the bottom left of the main window and on the contents of the **Validity tool**. This system property is examined once and for all. Therefore it is best to specify it as a command-line argument (e.g. `-DXXE_SHOW_SEMANTIC_ERRORS=1`) or in a `customize.xxe` file (e.g. `<property name="XXE_SHOW_SEMANTIC_ERRORS">1</property>`). It may also be specified in a `.xxe` configuration file. However, in this case, you'll have to make sure that this configuration file is loaded before any other one.
XXE_STRICT_CONREF	Any value. Suffice for this system property to be set.	When transcluding DITA conrefs, check the compatibility of the domains of the referencing document with the domains of the referenced document.
XXE_XEP_CONFIG	URL or absolute file path of a user-defined XEP configuration file	Same as XXE_FOP_CONFIG [207], but applies to RenderX XEP rather than to Apache FOP. More information in Using in **XXE** an existing XEP configuration file [133].