XMLmind XML Editor - DocBook Support

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

June 30, 2026

Abstract

This document describes the commands which are specific to DocBook v4 and v5.

Table of Contents

1. The DocBook menu 1
    1.1. The "**Convert Document**" sub-menu 8
    1.2. Creating `olink` elements 10
        1.2.1. Specifying the `targetdoc` and `targetptr` attributes of an `olink` element 11
        1.2.2. Specifying the set of olink-ed documents 11
    1.3. Using the `indexterm` editor 13
2. The DocBook tool bar 16
    2.1. Table editor 21
3. Other custom bindings 23
4. Table rendering 24
    4.1. HTML tables 25
5. Adding equations to your DocBook document 25

1. The DocBook menu

Upgrade to DocBook version 5

This menu item is present only if the document being edited is a DocBook *version 4* document. Converts DocBook version 4 document being edited it to DocBook version 5.0, 5.1 or 5.2. A dialog box is displayed to allow choosing a DocBook version. A file chooser is displayed to allow choosing a save file for the DocBook 5 document.

Note that this command does not automatically upgrade the documents referenced in the document being edited (e.g. a `chapter` included in a `book`). This has to be done manually for each referenced document. Once this is done, the `xi:include` elements have to be edited by hand in the master DocBook 5 document after using **Edit** → **Reference** → **Untransclude All** in *XMLmind XML Editor - Online Help*.

Set up olinks

Displays a dialog box allowing to declare the collection of DocBook documents in which `olink` is used for cross-referencing. More information in Specifying the set of olink-ed documents [11].

Paste As

Paste from Word, Browser or Markdown

Import the HTML copied to the clipboard by word processors or web browsers and intelligently paste it into the DocBook document being edited.

- Extensive efforts are made to decently support the “non-filtered HTML” copied by MS-Word to the clipboard.

- When an application other than MS-Word is used, the quality of the result you'll get in **XXE** highly depends on what has been copied to the clipboard. In all cases, **XXE** tries very hard to import something simple, clean and valid at the expense of the fidelity to the original data.

In addition to HTML, this menu entry can also import Markdown whose text source has been copied to the clipboard by text editor or web browsers. Markdown support is documented in the description pane of the "**Install Add-ons**" dialog box, **Uninstall** tab, when the "**Paste from Word Processor or Browser**" add-on is selected. This documentation is also found in "*Paste from Word Processor or Browser*".

The pasted data replaces the text or node selection if any. When there is no selection, XMLmind XML Editor automatically determines a valid insertion location at or following the caret position.

If XMLmind XML Editor fails to find such valid insertion location, the rich text is converted to valid DocBook and then copied to the clipboard, overwriting the original data put there by the third-party application. This allows to use the “normal” **Paste Before**, **Paste** or **Paste After** commands to paste the data elsewhere in the document.

This menu entry allows not only to paste snippets copied from word processors or web browsers, but also to import entire documents or HTML pages. In order to import the entire document into **XXE**:

1. Open the document in the third-party application.
2. Press **Ctrl**+**A** (**Select All**) then press **Ctrl**+**C** (**Copy**) to copy it to the clipboard.
3. Create a new DocBook document in **XXE** by using **File** → **New**.
4. Use **File** → **Save As** to save this new DocBook document to disk.
5. Explicitly select the root element of the DocBook document, for example by clicking on its name in the node path bar.
6. Select menu item "**Paste from Word Processor or Browser**" to paste the content of the clipboard[1].

If, using MS-Word, you want to copy a piece of text rather than a paragraph, do not include the hidden character found at the very end of a paragraph (the *paragraph mark*) in your selection.

Other menu entries

The following entries of this submenu allow to paste the *plain text* copied to the clipboard, typically using a third-party word processor or spreadsheet, as:

- one or more paragraphs,
- OR a `programlisting` element,
- OR one or more list items,
- OR an itemized list,
- OR one or more table rows,
- OR a table.

The last two menu entries assume that each text line specifies a table row and that, within a text line, the contents of the table cells are separated by tab characters.

[1]Note that **Ctrl**+**V**, that is, the plain **Edit** → **Paste** command, would not work here.

If you need to paste the copied text as an ordered list, first paste this text as an itemized list then convert the pasted list to an ordered list using **Edit** → **Convert** (**Ctrl**+**T**).

The following entries of this submenu allow to paste the *image* copied to the clipboard as:

- `inlinemediaobject`,
- `mediaobject`,
- `figure`.

Menu entry "**inlinemediaobject**" replaces the text or node selection if any. When there is no selection, this menu entry pastes its element at caret position (just like **Edit** → **Paste**).

All the other menu entries also replace the text or node selection if any. When there is no selection, these menu entries paste their elements at any valid position in the document following the caret position.

Parse Markdown as XML, Parse Text as XML [2]

Keyboard shortcut **Ctrl**+**Shift**+SPACE. Replace Markdown markup found all the text nodes of explicitly or implicitly selected element by equivalent DocBook elements. Supported Markdown markup: italic, bold, code, highlight (replaced by `emphasis role=underline`), subscript, superscript, link, image, heading (replaced by `bridgehead` with attribute `renderas`), unordered list item, ordered list item.

DocBook 5 example (notice the space character after "`1.`"; this space character is required after all the different “block prefixes” "`#`", "`##`", "`-`", "`*`", "`1.`", etc):

```
<para>* <trademark>Dart</trademark> is a *programming language*.
See [Dart](https://dart.dev/).</para>
```

is replaced by:

```
<itemizedlist>
  <listitem>
    <para><trademark>Dart</trademark> is a
    <emphasis>programming language</emphasis>. See
    <link xlink:href="https://dart.dev/">Dart</link></para>
  </listitem>
</itemizedlist>
```

Convert between informal element and element

Converts an “informal element” to/from a “formal element” having a title.

This command currently works for `informaltable`/`table` (CALS tables only, not HTML tables), `informalfigure`/`figure` and `informalexample`/`example`.

Link callouts

Links a sequence of `callout` elements to the corresponding sequence of `co` or `area` elements (and, of course, also the other way round).

[2] "**Parse Markdown as XML**" may be replaced by a variant called "**Parse Text as XML**" when the "MathML support" add-on is installed.

Useful information about callouts is found in *DocBook XSL: The Complete Guide* by Bob Stayton: Program listings, Annotating program listings, Callouts.

In order to use this command, you need to:

1. Create a `programlisting` containing a number of `co` elements. No need to specify the ID or `linkends` attributes for these `co` elements.

   Note that this command also works for any element containing `area` elements rather than `co` elements (e.g. a `programlistingco`).

2. Add a `calloutlist` element somewhere after the `programlisting`. No need to specify the ID or arearefs attributes for the `callout` elements.

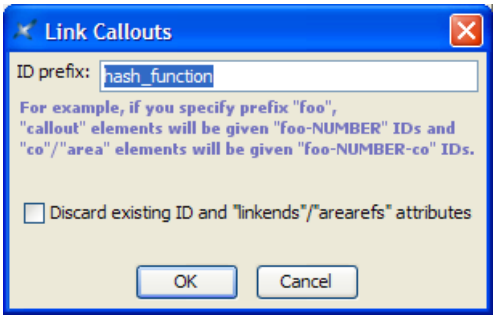
   Make sure to create exactly the same number of `co` and `callout` elements. This is needed because the *N*th `co` element will be linked to the *N*th `callout` element.

3. Explicitly select the node range comprising both the `programlisting` and the `calloutlist` elements.

   In fact, you can explicitly or implicitly select any element containing, at any nesting level, a sequence of `co` or `area` elements followed by a sequence of `callout` elements. For example, if your `programlisting` and `calloutlist` elements are contained in a `programlistingco` element, simply click anywhere inside the `programlistingco` element.

4. Select **DocBook** → **Link callouts**.

5. The following dialog box is displayed:

   Specify a prefix for the IDs which will be automatically generated for the `co` and the `callout` elements. The links (`linkends` and `arearefs` attributes) between the `co` and the `callout` elements of course need to refer to these IDs.

6. Click **OK**.

Notice that the above dialog box has a "**Discard existing ID and linkends/arearefs attributes**" checkbox. This checkbox is needed because the "**Links callouts**" command has been designed to be used, not only on newly created `programlisting` plus `calloutlist` elements, but also on existing, possibly hand-written, possibly complex[3] `programlisting` plus `calloutlist` elements.

[3]For example, containing a `callout` element linked to *several* `co` elements. In such case, the numbering of `co` and `callout` elements done on screen by XMLmind XML Editor will not reflect what you'll get when you'll convert your document to HTML or PDF. However this limitation should not prevent you from specifying such multi-`co` `callout` elements if needed to.

When the `co` and `callout` elements found inside the node selection are found to already have ID attributes, this checkbox is enabled and, by default, unchecked. When this is the case, running this command will affect only the newly created `co` and `callout` elements. All the existing IDs and links will be left unchanged.

This command also works with image maps

DocBook → **Link callouts** is also designed to work with the DocBook equivalent of *HTML image maps*.

An easy way to create an image map pointing to a `callout` list describing areas of interest in the image is to proceed as follows (DocBook 5 example):

1. Use tool bar button "**Add image**" [20] and select "**mediaobject(calloutlist)**" to add a `mediaobject` element containing an `imageobjectco` element to your document.
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 `imageobjectco` element and select "**Edit Image Map**" from the contextual popup menu.
4. Use the image map editor in *XMLmind XML Editor - Online Help* 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.
5. 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.
6. Use **DocBook** → **Link callouts** to link the hot areas to the corresponding `callout` elements (and the other way round of course).

Insert or Edit indexterm

If the caret is anywhere inside an `indexterm` element or if a single element or node is explicitly selected anywhere inside an `indexterm` element, this menu item displays an `indexterm` editor dialog box [13] allowing to modify this `indexterm` element.

Otherwise, this menu item displays an `indexterm` editor dialog box [13] allowing to create a new `indexterm` element and then to insert it at caret position.

If some text has been selected, field **Term** of the dialog box is automatically initialized with the text selection. Therefore the simplest way to create an `indexterm` element is first to select the term in the body of the document, then invoke **Insert or Edit indexterm** and finally click **OK**.

Move Up

Move selected element up, that is, swap it with its preceding sibling node. Requires the element to be explicitly selected.

Move Down

Move selected element down, that is, swap it with its following sibling node. Requires the element to be explicitly selected.

Promote

To make it simple, increase the level of selected subsection (e.g. a `sect2` element is converted to a `sect1` element).

Requires a ``subsection'' (`section`, `sect1`, `sect2`, `sect3`, `sect4` or `sect5`) or an element which is contained in the body[4] of the section to be explicitly selected.

- If a subsection is selected, this subsection becomes a sibling of its parent section. Example: `sect2` element having `id="C"` is ``promoted'':

```
<sect1 id="A">...
  <sect2 id="B">...
  <sect2 id="C">...
  <sect2 id="D">...
```

  This results in:

```
<sect1 id="A">...
  <sect2 id="B">...
<sect1 id="C">...
  <sect2 id="D">...
```

- If another type of child element is selected, this element is wrapped in a newly created section which becomes a sibling of its parent section. Example: `para` element having `id="C"` is ``promoted'':

```
<sect1 id="A">...
  <para id="B">...
  <para id="C">...
  <sect2 id="D">...
```

  This results in:

```
<sect1 id="A">...
  <para id="B">...
<sect1>...
  <para id="C">...
  <sect2 id="D">...
```

Demote

To make it simple, decrease the level of selected section (e.g. a `sect1` element is converted to a `sect2` element).

Requires a ``section'' (`chapter`, `appendix`, `section`, `sect1`, `sect2`, `sect3` or `sect4`) or an element which is contained in the body[5] of the section to be explicitly selected.

[4] That is, it is not possible to ``promote'' the *title* of a section.
[5] That is, it is not possible to ``demote'' the *title* of a section.

- If a section is selected and if this section is preceded by a section of the same type, this section becomes a subsection of its preceding sibling. Example: `sect1` element having `id="C"` is ``demoted'':

```
<sect1 id="A">...
  <para id="B">...
<sect1 id="C">...
  <para id="D">...
```

  This results in:

```
<sect1 id="A">...
  <para id="B">...
  <sect2 id="C">...
    <para id="D">...
```

- If a section is selected and if this section is *not* preceded by a section of the same type, a new section is created and selected section becomes a subsection of this new section. Example: `sect2` element having `id="C"` is ``demoted'':

```
<sect1 id="A">...
  <para id="B">...
  <sect2 id="C">...
    <para id="D">...
```

  This results in:to declare the collection of DocBook documents in which olink is used for cross-referencing. How to do this is explained in next section.

```
<sect1 id="A">...
  <para id="B">...
  <sect2>...
    <sect3 id="C">...
      <para id="D">...
```

- If another type of child element is selected, this element and all the other ``body elements'' which follow it are wrapped in a newly created subsection. Example: `para` element having `id="C"` is ``demoted'':

```
<sect1 id="A">...
  <para id="B">...
  <para id="C">...
  <para id="D">...
  <sect2 id="E">...
```

  This results in:

```
<sect1 id="A">...
  <para id="B">...
  <sect2>...
    <para id="C">...
    <para id="D">...
  <sect2 id="E">...
```

Check External References

Keyboard shortcut **F8**. Checks all the links to external resources found in the document being edited. All kinds of external resources are checked for existence: images, audio, video, PDF documents, HTML pages, etc. When the resource is an HTML page and the link ends with a fragment (e.g. "`#bar`" in "`../doc/foo.html#bar`") then this fragment is also checked for existence.

1.1. The "Convert Document" sub-menu

The "**Convert to RTF**", **WML**, **DOCX**, **ODT**, entries documented below are absent in XMLmind DocBook Editor. They are found only in XMLmind XML Editor.

Using the profiling stylesheets

Conditional processing, also called *profiling* or conditional text, means that you can create a single XML document with some elements marked as conditional. When you process such a document, you can specify which conditions apply for that version of the output, and the XSLT stylesheet will include or exclude the marked text to satisfy the conditions. More information in DocBook XSL: The Complete Guide.

If you need to use the profiling XSLT stylesheets rather than the regular ones, use **Options** → **Customize Configuration** → **Customize Document Conversion Stylesheets** in *XMLmind XML Editor - Online Help* and select the corresponding stylesheet.

Convert to HTML, **Convert to HTML [one page]**

Converts the document being edited to multi page or single page HTML.

Generating XHTML rather than HTML

If you prefer to generate XHTML 1.0 or 5 rather than plain HTML, use **Options** → **Customize Configuration** → **Customize Document Conversion Stylesheets** and select the corresponding stylesheet.

Convert to Web Help

Converts the document being edited to *Web Help* containing XHTML 5 pages.

The Web Help is generated by XMLmind Web Help Compiler and not by the XSLT stylesheets generating Web.Help which are part of the stock DocBook XSL Stylesheets.

Therefore, you *cannot* use any of XSLT stylesheet parameters having a name starting with "`webhelp.`" and which are documented in "*DocBook XSL Stylesheets: Reference Documentation, HTML Parameter Reference, WebHelp*". For example, using *`webhelp.autolabel`* would have *no effect* on the Web Help generated by this menu entry.

Instead please use any of the parameter documented in "*XMLmind Web Help Compiler Manual, Parameters*" after prefixing their names with "`wh-`". Example: `wh-favicon=https://docbook.org/graphics/banner.png`

Convert to HTML Help

Converts the document being edited to a `.chm` file. This command is disabled on platforms other than Windows.

For this command to work, the HTML Help compiler, `hhc.exe`, must have been declared as the helper application associated to files having a "`hhp`" extension. This can be specified by using the **Preferences** dialog box, **Helper Applications** section.

Convert to Eclipse Help

Converts the document being edited to Eclipse Help.

If you want Eclipse to display your Eclipse Help document in its help viewer, you must

1. specify the following XSLT stylesheet parameters: *`eclipse.plugin.name`*, *`eclipse.plugin.id`*, *`eclipse.plugin.provider`*, prior to selecting **DocBook** → **Convert Document** → **Convert to Eclipse Help**;
2. give to the output folder the name specified in *`eclipse.plugin.id`*;
3. copy the output folder containing the generated Eclipse Help document to *`eclipse_install_dir`*`/dropins/` and not *`eclipse_install_dir`*`/plugins/`.

Convert to EPUB

Converts the document being edited to EPUB.

Convert to RTF (Word 2000+)

Converts the document being edited to RTF (Rich Text Format) using XMLmind FO Converter (see http://www.xmlmind.com/foconverter/). The document generated by this command can be edited and printed using Microsoft® Word 2000 and above.

Convert to WordprocessingML (Word 2003+).

Converts the document being edited to WordprocessingML using XMLmind FO Converter. The document generated by this command can be edited and printed using Microsoft® Word 2003 and above.

Convert to Office Open XML (Word 2007+)

Converts the document being edited to Office Open XML (`.docx` file) using XMLmind FO Converter. The document generated by this command can be edited and printed using Microsoft® Word 2007 and above.

Convert to OpenDocument (OpenOffice.org 2+)

Converts the document being edited to OpenDocument (`.odt` file) using XMLmind FO Converter. The document generated by this command can be edited and printed using OpenOffice.org 2.

Convert to PDF

Converts the document being edited to PDF (Adobe® Portable Document Format, also known as Acrobat®) using RenderX XEP (see http://www.renderx.com/), if its plug-in has been installed, and Apache FOP otherwise (see http://xmlgraphics.apache.org/fop/).

All the above **Convert** commands display the URL chooser dialog box rather than the standard file chooser dialog box.

For all **Convert** commands except for the "**Convert to HTML**" command, you must specify the URL (Uniform Resource Locator) of a save file. The "**Convert to HTML**" command creates multiple HTML pages with a first page called `index.html`, therefore you need to specify the URL of a save directory.

Note that these commands can create directories on the fly, if needed to. For example, if you specify `http://www.acme.com/docs/report43/mydoc.html` as the URL of the save file and if directory `report43/` does not exist, this directory will be created during command execution.

Syntax highlighting

You can automatically colorize the source code contained in `programlisting` elements. This feature, commonly called *syntax highlighting*, has been implemented using an open source software component called "XSLT syntax highlighting".

If you want to turn on syntax highlighting in a DocBook document:

1. Add attribute `language` to element `programlisting`. The value of attribute `language` must be any of: `bourne`, `c`, `cmake`, `cpp`, `csharp`, `css21`, `delphi`, `ini`, `java`, `javascript`, `lua`, `m2` (Modula 2), `perl`, `php`, `python`, `ruby`, `sql1999`, `sql2003`, `sql92`, `tcl`, `upc` (Unified Parallel C), `html`, `xml`.
2. Specify XSLT stylesheet parameter highlight.source=1 using **Options** → **Customize Configuration** → **Change Document Conversion Parameters**. Do this for each output format you want to generate.

If you want to customize syntax highlighting for an HTML-based output format (XHTML, EPUB, etc), redefine any of the following CSS styles: `.hl-keyword`, `.hl-string`, `.hl-number`, `.hl-comment`, `.hl-doccomment`, `.hl-directive`, `.hl-annotation`, `.hl-tag`, `.hl-attribute`, `.hl-value`, `.hl-doctype`. Example:

```
.hl-keyword {
    font-weight: bold;
    color: #602060;
}
```

This can be done from within **XXE** using **Options** → **Customize Configuration** → **Customize Document Conversion Stylesheets**.

If you want to customize syntax highlighting for an XSL-FO-based output format (PDF, RTF, etc), redefine any of the following `attribute-sets`: `hl-keyword`, `hl-string`, `hl-number`, `hl-comment`, `hl-doccomment`, `hl-directive`, `hl-annotation`, `hl-tag`, `hl-attribute`, `hl-value`, `hl-doctype`. Example:

```
<xsl:attribute-set name="hl-keyword" use-attribute-sets="hl-style">
  <xsl:attribute name="font-weight">bold</xsl:attribute>
  <xsl:attribute name="color">#602060</xsl:attribute>
</xsl:attribute-set>
```

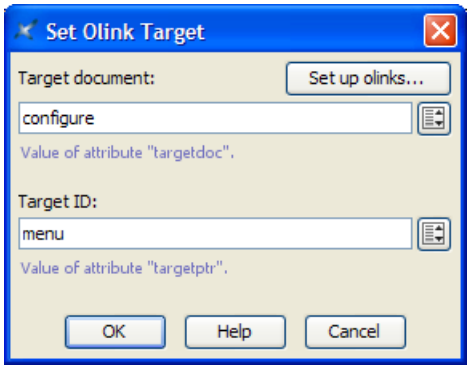
This can be done from within **XXE** using **Options** → **Customize Configuration** → **Customize Document Conversion Stylesheets**.

1.2. Creating `olink` elements

The `olink` element allows to create links between different documents. Once the `olink` element has been inserted in a document, you have to specify a value for its `targetdoc` attribute and optionally, a value for its `targetptr` attribute. The `targetdoc` attribute contains the symbolic name of the document which is the target of the `olink`. The `targetptr` attribute is the ID of an element found in the target document. More information about the `olink` element and how this element is processed by the DocBook XSL stylesheets in *DocBook XSL: The Complete Guide*, by Bob Stayton.

1.2.1. Specifying the `targetdoc` and `targetptr` attributes of an `olink` element

The easiest way to specify the `targetdoc` and `targetptr` attributes of an `olink` element is to right-click anywhere inside the `olink` element. Doing this displays a contextual menu containing "**Set Link Target**" in addition to "**Follow Link**". Menu item "**Set Link Target**" displays the following specialized dialog box:



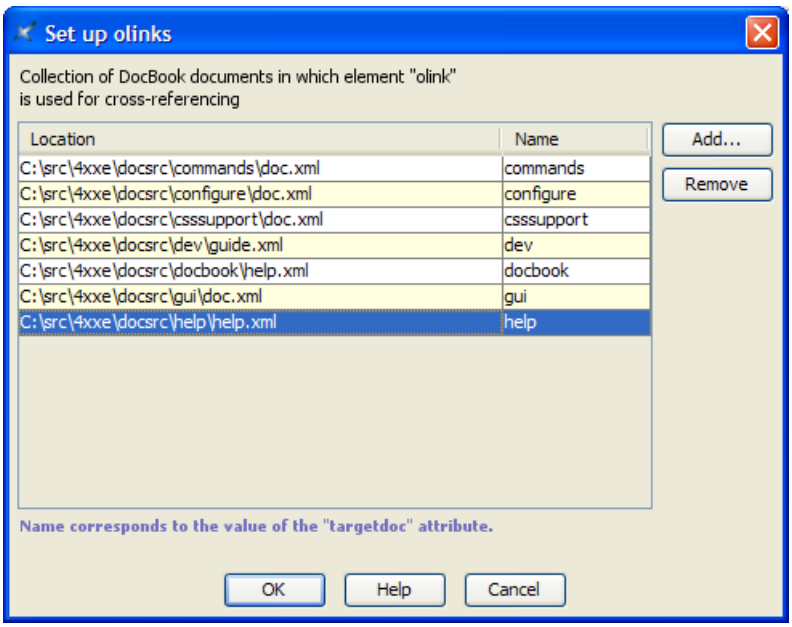
The **Attributes** tool also can help you specify a value for the `targetdoc` attribute by listing[6] all the symbolic names of the target documents. Once the `targetdoc` attribute has been specified, the **Attributes** tool can help you specify a value for the `targetptr` attribute by listing all the IDs found in the target document.

However for the two above facilities to work, you first need to declare the collection of DocBook documents in which `olink` is used for cross-referencing. How to do this is explained in next section.

1.2.2. Specifying the set of olink-ed documents

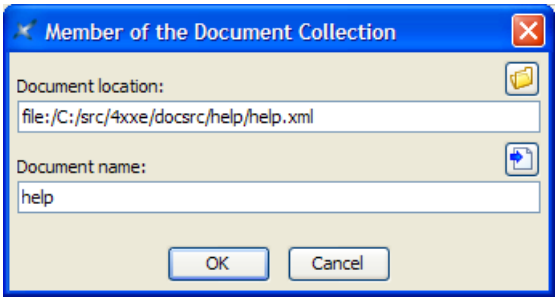
Procedure:

1. Select **DocBook** → **Set up olinks**. This will display the following dialog box:



[6] Type a value in the **Value** field and use auto-completion or use the **Edit** button which is found at the right of this text field to display a specialized dialog box.

2. Click **Add**. This will display this other dialog box.



3. Use the **Browse** button to specify the URL of a document which is a member of the collection. In the above screenshot, this URL is "`file:/C:/src/4xxe/docsrc/help/help.xml`".

You can mix DocBook 4 and DocBook 5 documents within the same collection.

4. Type the symbolic name of the document in the **Document name** text field. In the above screenshot, this name is "`help`".

   This name, which cannot contain space characters, corresponds to a possible value for the `targetdoc` attribute. The same symbolic name must also be used in the *target database document*. Example:

```
<!DOCTYPE targetset
  SYSTEM "../../addon/config/docbook/xsl/common/targetdatabase.dtd" [
  ...
  <!ENTITY help SYSTEM "help_html.targets">
  ...
]>
<targetset>
  <sitemap>
    <dir name="doc">
      ...
      <dir name="help">
        <document targetdoc="help">
          &help;
        </document>
      </dir>
      ...
    </dir>
  </sitemap>
</targetset>
```


Instead of typing the symbolic name of the document referenced in the **Document location** text field, it's also possible to click the button. This button allows to use the ID of the root element (if any) of the document referenced in the **Document location** text field as a symbolic name.

Using the ID of the root element as the symbolic name of an “olink-ed document” is a common practice. However, before using this button, make sure that this practice is actually used in your organization.

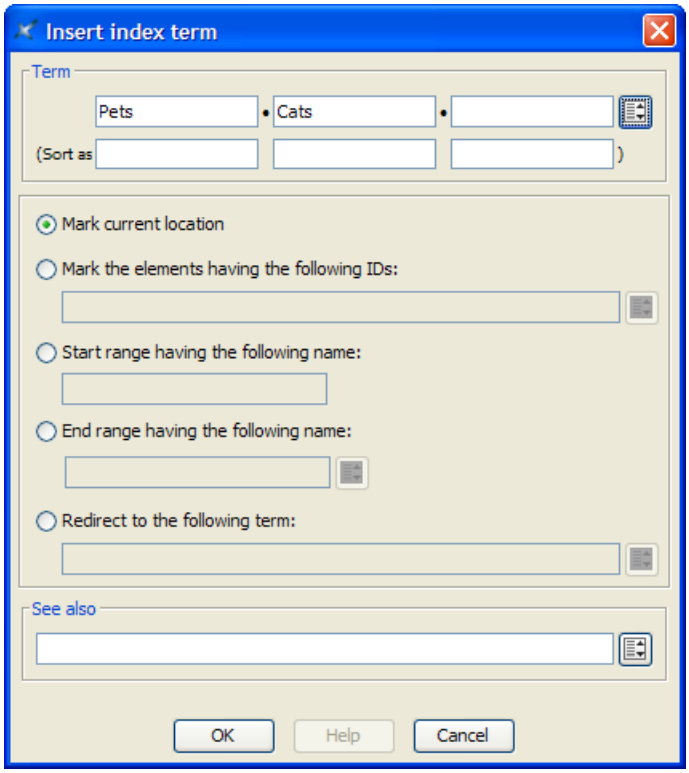
5. Repeat steps 1 to 4 until you have declared all the members of your document collection.

This setup is done once for all for both the DocBook 4 and 5 configurations. However you may add or remove members to/from your document collection at any time.

XXE can help you create `olink` elements. However it is important to understand that **XXE** cannot help you in putting these `olink` elements into use when converting your DocBook document to HTML, PDF, etc. For example, **XXE** cannot assist you in creating the sitemap file, in populating it with link targets, etc. All these tasks must be performed “by hand”, outside **XXE**. More information in *DocBook XSL: The Complete Guide*, by Bob Stayton.

1.3. Using the `indexterm` editor

This dialog box, displayed by menu item **DocBook** → **Insert or Edit indexterm** [5], allows to edit the selected `indexterm` element if any, or to create a new `indexterm` element and then insert it at caret position otherwise.



We'll explain with examples how to use the `indexterm` editor.

- If you want to get this kind of entry in your back of the book index:

```
P
```

```
Pet 12
```

specify **Term**=`Pet`.

- Back of the book index:

```
P
Pet
    Cat 26
```

specify **Term**=`Pet`, **Term** #2=`Cat`.

- Back of the book index:

```
P
"+" 54
```

specify **Term**=`"+"`, **Sort as**=`plus`. Without this **Sort as** specification, the index entry corresponding to `"+"` would have been found in the **Symbols** category:

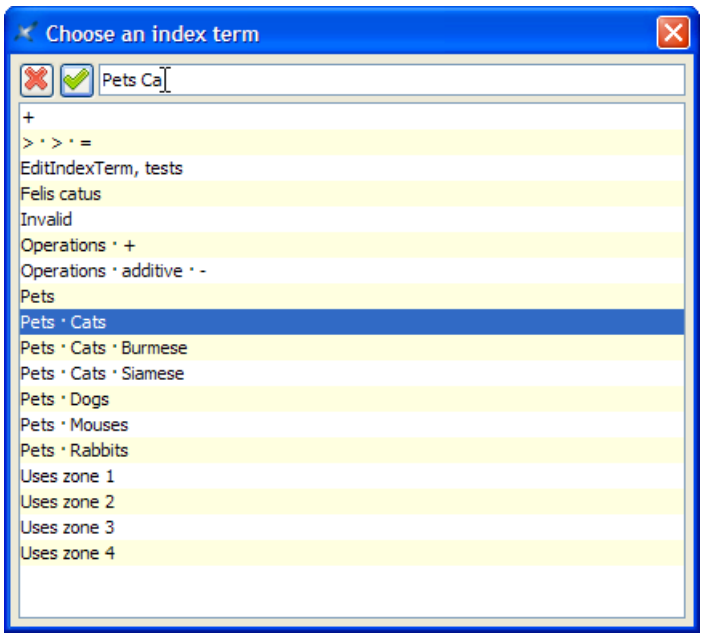
```
Symbols
"*" 53, 78
"+" 54
"-" 55, 91-95
...
```

- Back of the book index:

```
D
Domesticated animals 34 See also Pet
```

specify **Term**=`Domesticated animals`, **See also**=`Pet`.

Note that the content of the **See also** field must refer to an existing index entry. That's why instead of typing "`Pet`", you can select this index entry by using the dialog box displayed by the **Pick from list** button found at the right of the **See also** row.



The above dialog box supports autocompletion. Note that if, for example, you want specify compound term "`Pet Cat Siamese`", you must type a space character between each simple term.

- Back of the book index:

```
I
IT See Information Technology
```

specify **Term**=`IT`, select "**Redirect to the following term**" then specify **Redirect**=`Information Technology`. (In the above example, notice that `IT` has no associated page number.)

Like **See Also**, the content of the **Redirect** field must refer to an existing index entry. Unlike **See Also**, a **Redirect** entry is merely a redirection to an actual index entry.

- Back of the book index:

```
O
Operation
    Additive
        "+" 87-90
```

1. Insert a first `indexterm` element at the beginning the range (this will give us page number 87).

   In order to do that, use **DocBook** → **Insert or Edit indexterm** and specify **Term**=`Operation`, **Term** #2=`Additive`, **Term** #3=`"+"`, **Sort as** #3=`plus`.

   Then check "**Start range having the following name**" and give your `indexterm` element an ID by specifying "`plus_reference`" in the **Start range** field.

2. Insert another `indexterm` element at the end the range (this will give us page number 90).

   In order to do that, use **DocBook** → **Insert or Edit indexterm**, check "**End range having the following name**" and specify the same ID, "`plus_reference`", in the **End range** field. All the other fields must be left blank.

Note that instead of typing "`plus_reference`" in the **End range** field, you can select this ID by using the dialog box displayed by the **Pick from list** button found at the right of the **End range** field.

- Normally, that is, when "**Mark current location**" is selected, an `indexterm` element contributes to the back of the book index with its own page number. However, in some cases, it may be convenient to insert an `indexterm` at some place (typically in `chapterinfo`, `sectioninfo`, etc, elements) and specify that this `indexterm` corresponds to the page numbers of one or more other elements.

  Example: let's suppose that the `indexterm` element is contained in a `sectioninfo` element and that the chapter about dogs has "`ch.dogs`" its ID. Back of the book index:

```
P
Pet
    Dog 22
```

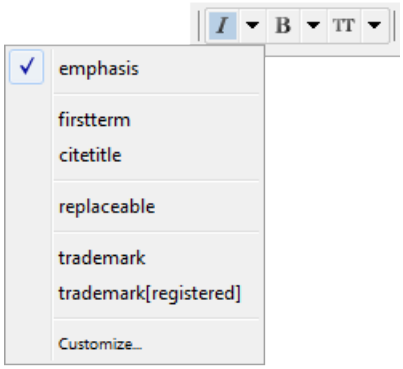
  specify **Term**=`Pet`, **Term** #2=`Dog`, select "**Mark the elements having the following IDs**" then specify **Element IDs**=`ch.dogs`.

  Note that instead of typing "`ch.dogs`" in the **Element IDs** field, you can select one or more IDs by using the dialog box displayed by the **Pick from list** button found at the right of the **Element IDs** field.

2. The DocBook tool bar

The DocBook tool bar starts 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. More information about text style toggles in About “text style” toggles in *XMLmind XML Editor - Online Help*.

Figure 1. 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.

Toggle emphasis

“Toggle” element `emphasis`. Next to this toggle is found an arrow button displaying a menu containing additional checkboxes for the following elements: `firstterm`, `citetitle`, `replaceable`, `trademark`, `trademark[registered]`.

Toggle emphasis[bold]

"Toggle" element `emphasis[bold]`. Next to this toggle is found an arrow button displaying a menu containing additional checkboxes for the following elements: `abbrev`, `guilabel`, `guibutton`, `guimenuitem`, `guisubmenu`, `guimenu`, `keycap`, `keysym`.

Toggle literal

"Toggle" element `literal`. Next to this toggle is found an arrow button displaying a menu containing additional checkboxes for the following elements: `filename`, `sgmltag[element]`, `sgmltag[attribute]`, `sgmltag[attvalue]`.

Change case

Displays a menu letting the user change the character case of selected text. If a single node is selected, this converts the character case of all the text contained in this node. If there is no selection of any kind, this converts the character case from caret position to end of word, then it moves the caret to the next word.

lowercase

All characters are converted to lowercase characters. Keyboard shortcut: **Esc**+**l**.

UPPERCASE

All characters are converted to uppercase characters. Keyboard shortcut: **Esc**+**u**.

Capitalize Each Word

First character of a word is converted to an uppercase character. The other characters are converted to lowercase characters. Keyboard shortcut: **Esc**+**c**.

Convert to plain text

Convert implicit or explicit selection to plain text.

Add link

Displays menu letting the user add a link to your document.

link[linkend] (DocBook 4: **link**)

Converts the selection, if any, to an "internal link"; simply inserts an empty "internal link" at caret position otherwise. An internal link is a `link` element having a `linkend` attribute.

link[href] (DocBook 4: **ulink**)

Converts the selection, if any, to an "external link"; simply inserts an empty "external link" at caret position otherwise. An external link is a `link` element having an `xlink:href` attribute (DocBook 4: an `ulink` element having an `url` attribute).

In the context of a modular document (e.g. a book comprising chapters, each chapter being contained in its own file), *do not create external links between modules* (e.g. chapters). Instead create internal links between modules.

xref

Inserts an `xref` element at caret position.

olink

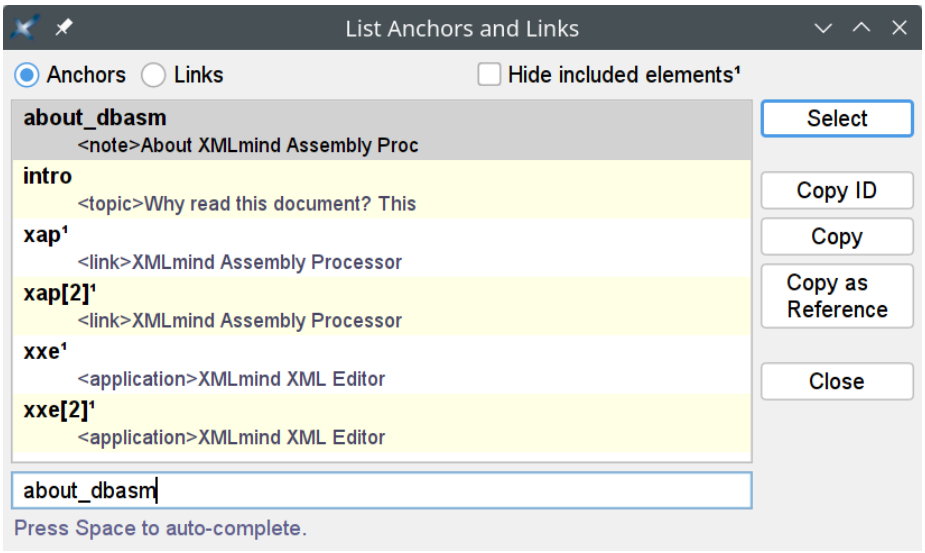
Converts the selection, if any, to an `olink` element (that is, a link between different DocBook documents); simply inserts an empty `olink` element at caret position otherwise.

Set ID

This button displays a menu having 2 entries: **Set ID** and **List Anchors and Links**.

The **List Anchors and Links** menu entry displays a dialog box letting the user search and select anchors (that is, any element having an ID) and links.

*Figure 2. The "**List Anchors and Links**" dialog box*



Button "**Copy ID**" (respectively **Copy Link**") copies selected ID (respectively selected link target) to the clipboard.
Button **Copy** may be used to copy selected anchor or link element to the clipboard just like **Edit** → **Copy** in *XMLmind XML Editor - Online Help*.
Button "**Copy As Reference**" may be used to copy selected *anchor* element as a reference to the clipboard just like **Edit** → **Reference** → **Copy As Reference** in *XMLmind XML Editor - Online Help*.

Decrease nesting level

Convert a paragraph to a list item and a list item to a paragraph, the new element having a lesser nesting level than the original one. This button automatically splits lists when needed to.

This is the inverse command of "**Increase nesting level**". More Information below [18].

Increase nesting level

Convert a paragraph to a list item and a list item to a paragraph, the new element having a greater nesting level than the original one. This button automatically creates lists or merges adjacent lists when needed to.

Note that the two above buttons strictly alternate between paragraphs and list items. This means that you'll often have to click a button *twice in a row*. For example, in order to create a nested list, first click anywhere inside a list item and then click "**Increase nesting level**" twice. First click converts the list item to a plain paragraph contained in the preceding list item. Second click converts this paragraph to the first item of a new nested list.

For the two above buttons to function, any of the following conditions should be met:

- A sequence of list items must be explicitly selected.
- A list must be explicitly selected. This is equivalent to selecting all its items.
- A sequence of blocks *starting with a paragraph* must be explicitly selected.
- A paragraph must be implicitly selected. In order to implicitly select a paragraph, suffice to click anywhere inside it. However if this paragraph is the first child of a list item, then it's the list item which is implicitly selected.
- A list item must be implicitly selected. In order to implicitly select a list item, suffice to click anywhere inside it.

When a paragraph is to be converted to a new list using this "**Increase nesting level**" button, there is a quick and simple way to specify the kind of list to be created:

- If the paragraph is empty, simply type "`*`", "`-`", "`1.`", "`a.`", "`A.`", "`i.`", "`I.`", optionally followed by space characters, in it.
- If the paragraph is not empty, type "`*`", "`-`", "`1.`", "`a.`", "`A.`", "`i.`", "`I.`", followed by at least one space character, at the very beginning of it.

These “prefixes” are used to create respectively: **itemizedlist**, **itemizedlist**, **orderedlist**, **orderedlist[loweralpha]**, **orderedlist[upperalpha]**, **orderedlist[lowerroman]**, **orderedlist[upperroman]**, as if the "**Change list type**" menu below was automatically used.

Change list type

Displays a menu allowing the user to change the type of the current list.

Additional menu entry "**Continue Numbering**", which is rendered as a checkbox, sets the `continuation` attribute of an ordered list to `continues` when this attribute is absent and removes this attribute otherwise. Additional menu entry "**Inherit Numbering**", which is rendered as a checkbox, sets the `inheritnum` attribute of an ordered list to `inherit` when this attribute is absent and removes this attribute otherwise.

The list must be explicitly or implicitly selected. In order to implicitly select a list, suffice to click anywhere inside it.

Known problems

Except for the simplest cases, "**Continue Numbering**" is *not correctly rendered* in the styled view of the document. For example, "**Continue Numbering**" has no visual effect on `orderedlist` elements having an `orderedlist` ancestor.

The reason of this limitation is that the implementation of "**Continue Numbering**" leverages standard CSS counters. However, when you'll convert your document to other formats such as HTML, PDF, DOCX, etc, there are no such limitations and "**Continue Numbering**" should give you the expected results.

Add list

Displays a menu allowing the user to select a type of list (`itemizedlist`, `orderedlist`, `variablelist`). The chosen list is added after node selection or after caret at a location where it is valid to do so.

Add para

Add a `para` element after node selection or after caret at a location where it is valid to do so.

Add list item

Add a `listitem` or `varlistitem` element after current list item. For this command to work, suffice to click anywhere inside an `itemizedlist`, `orderedlist` or `variablelist` element.

Add footnote

Displays a menu allowing the user to insert a footnote or a reference to a footnote (`footnoteref`) at caret position.

If a reference to a footnote is already selected, the **footnoteref** menu entry lets the user choose the ID of the footnote to be referenced.

Add note

Displays a menu allowing the user to add different kinds of admonitions after node selection or after caret at a location where it is valid to do so.

Add programlisting

Displays a menu allowing the user to add different kinds of elements containing preformatted text after node selection or after caret at a location where it is valid to do so.

Menu entry "**Normalize Whitespace**" normalizes whitespace in implicitly or explicitly selected program listing. Normalizing whitespace means: expanding tab characters to a number of space characters and removing the space characters which are common to the beginning of all text lines (that is, removing the superfluous “indentation” in the program listing, if any).

Add image

Displays a menu letting the user add different kinds of images and figures after node selection or after caret at a location where it is valid to do so.

Adding an image map to your document

The following menu items allow to add the equivalent of an *HTML image map* to your DocBook documents.

mediaobject(callout)

Add an image map containing internal links typically pointing to the `calloutlist` element found at the end of the inserted `imageobjectco` element. See also command "**Link callouts**" [5].

mediaobject(imagemap) *(DocBook 5 only)*

Add an image map containing external links typically pointing to Web pages.

Once any of the above menu items has been used, right-click anywhere inside the newly inserted `imageobjectco` element and select "**Edit Image Map**" from the contextual popup menu to display an *image map editor*. This image map editor allows to add “hot areas” to your image. More information in Section 17, “The "**Edit Image Map**" dialog box” in *XMLmind XML Editor - Online Help*.

Add MathML equation *(DocBook 5 only)*

Displays a menu letting the user add various kinds of MathML equations after node selection or after caret at a location where it is valid to do so.

Add media object *(DocBook 5.1+ only)*

Displays a menu containing the following items:

inlinemediaobject(audio)

Inserts an `inlinemediaobject` containing an `audioobject` at caret position.

inlinemediaobject(video)

Inserts an `inlinemediaobject` containing an `videoobject` at caret position.

mediaobject(audio)

Adds a `mediaobject` containing an `audioobject` after node selection or after caret at a location where it is valid to do so.

mediaobject(video)
Adds a `mediaobject` containing an `videoobject` after node selection or after caret at a location where it is valid to do so.

More information in Section 16, “The media player dialog box” in *XMLmind XML Editor - Online Help*.

Do not forget to select the DocBook XSL stylesheets generating XHTML 5 rather than plain HTML

If your document contains audio and video elements, do not forget to select the DocBook XSL stylesheets generating XHTML 5 prior to using **DocBook** → **Convert Document** → **Convert to HTML**. This setting is done once for all using **Options** → **Customize Configuration** → **Customize Document Conversion Stylesheets** in *XMLmind XML Editor - Online Help*.

Add section
Displays a menu allowing the user to add chapter or section elements after node selection or after caret at a location where it is valid to do so.

Add table
Displays a menu allowing the user to add different kinds of tables after node selection or after caret at a location where it is valid to do so.

Table editor
See Section 2.1, “Table editor” [21].

2.1. Table editor

The following table editing commands fully support CALS tables as well as HTML tables. Most table editing commands can be repeated by using **Edit** → **Repeat** (**Ctrl**+**A**).

Note that using this table editor, or simply saving a document, or checking a document for validity, guarantees that the `cols` attribute of a `tgroup` is up to date. That is, you may forget about the `cols` attribute, XMLmind XML Editor will always compute it for you.

Button	Menu item	Description
Table column For a command in this menu to work, click anywhere inside a cell (or explicitly select a cell or an element having a cell ancestor).	**Insert Before**	Insert a column before column containing specified cell.
	Insert After	Insert a column after column containing specified cell.
	Cut	Cut to the clipboard the column containing specified cell.
	Copy	Copy to the clipboard the column containing specified cell.
	Paste Before	Paste copied or cut column before column containing specified cell.
	Paste After	Paste copied or cut column after column containing specified cell.
	Delete	Delete the column containing specified cell.
	Sort Rows	Sort all the rows of the table according to the string values of the cells of the “selected column”. (The “selected column” is the column containing specified cell.) A dialog box is displayed allowing to specify the following sort options:

Button	Menu item	Description
		Order **Dictionary** is the language-specific alphabetical order. Example: (`Charles`, `best`, `Albert`) is sorted as (`Albert`, `best`, `Charles`). **Numeric**. The string value of a cell is expected to start with a number. Example: (`+15.0%`, `1.50%`, `-20%`) is sorted as (`-20%`, `1.50%`, `+15.0%`). **Lexicographic** is the order of Unicode characters. Example: (`Charles`, `best`, `Albert`) is sorted as (`Albert`, `Charles`, `best`). **Dictionary** and **Numeric** orders will cause this menu item to fail, unless the language of the table can be determined (i.e. lookup for the `lang` attribute). Direction **Ascending** means: A to Z, low to high. **Descending** means: Z to A, high to low. Note that: • Header/footer rows (i.e. `thead`) are never sorted. • The contents of row groups (i.e. `tbody`) are sorted separately.
Table row For a command in this menu to work, click anywhere inside a cell (or explicitly select a cell or an element having a cell ancestor) or explicitly select a row.	**Insert Before-For**	Insert a row before row containing specified cell. Note that row editing commands are enabled, not only by implicitly or explicitly selecting a table cell or any of its descendants, but also by explicitly selecting a table row.
	Insert After	Insert a row before row containing specified cell.
	Cut	Cut to the clipboard the row containing specified cell.
	Copy	Copy to the clipboard the row containing specified cell.
	Paste Before	Paste copied or cut row before row containing specified cell.
	Paste After	Paste copied or cut row after row containing specified cell.
	Delete	Delete the row containing specified cell.
Table cell For a command in this menu to work, click anywhere inside a cell	**Increment Column Span**	Increment the number of columns spanned by specified cell.
	Decrement Column Span	Decrement the number of columns spanned by specified cell.
	Increment Row Span	Increment the number of rows spanned by specified cell.
	Decrement Row Span	Decrement the number of rows spanned by specified cell.

Button	Menu item	Description
(or explicitly select a cell or an element having a cell ancestor).	**Set Color**	Displays a dialog box allowing to give a background color to specified cell. Unlike the other entries of this menu, this entry allows to give a background color, not only to specified cell, but also to one or more of any of the following explicitly selected elements: `row`, `entry`, `tr`, `td`, `th`.

3. Other custom bindings

Keystroke	Action
Alt+Shift+Up	Same as menu item Move Up [5].
Alt+Shift+Down	Same as menu item Move Down [6].
Alt+Shift+Left	Same as toolbar button Decrease nesting level [18].
Alt+Shift+Right	Same as toolbar button Increase nesting level [18].
Ctrl+Alt+click (**Cmd**+Alt+click on the Mac)	Follow the link clicked upon.
Enter	Insert a newline character if possible. Otherwise, if caret is at the beginning of a paragraph, list item or a few other kinds of block, insert same block before. Otherwise, if caret is at the end of a block, insert same block after. Otherwise, split block.
Del	Delete selection if any. Otherwise, if caret is at the end of a paragraph, list item or a few other kinds of block, join with following block. Otherwise, delete character following caret.
BackSpace	Delete selection if any. Otherwise, if caret is at the beginning of a paragraph, list item or a few other kinds of block, join with preceding block. Otherwise, delete character preceding caret.
Ctrl+Enter	Add same block after the paragraph, list item or a few other kinds of block which is the ancestor of selected node.
Ctrl+Shift+Enter	Add same block before the paragraph, list item or a few other kinds of block which is the ancestor of selected node.
Application Event	**Action**
Drop an object.	• On a `ulink` element, change the value of the `url` attribute to the dropped string.

Application Event	Action

	• On an `image` element, change the value of the `fileref` attribute to the dropped string.
	• Elsewhere
	  • If the object being dropped represents an URL or an absolute filename, open the corresponding document in XMLmind XML Editor.
	  • Otherwise, displays a popup menu allowing to paste the dropped text or XML before, into or after the drop location.

Drag one of the “handles” displayed around an image. (The “handles” are displayed after clicking on the image.)	Resize the image, but always preserve its aspect ratio. Pressing Ctrl (Cmd on the Mac) while dragging the handle allows to distort the image.
Drag a separator found between two table columns.	Resize the table column. More precisely this gives an appropriate proportional width (e.g. `<colspec colwidth="3*">`) to *all* table columns.

4. Table rendering

The following attributes are either completely ignored or partially supported. All other attributes are supported.

Attribute	Support
`table` (or `informaltable`) `orient`	Ignored.
`table` (or `informaltable`) `pgwide`	Ignored.
`colspec colwidth`	All forms including "`2*`" or "`3*+1pc`" are supported. Coefficients of "`*`" are always converted to integers. Examples: "`2.5*`" is equivalent to "`2*`". "`3.95*+0.5in`" is equivalent to "`4*+0.5in`". A column must contain at least one cell with a column span equal to 1 for the `colwidth` attribute to have an effect.
`entry rotate`	Ignored.
`align`	Values `justify` and `char` are rendered like `left`.
`char`	Ignored. See `align`.
`charoff`	Ignored. See `align`.

4.1. HTML tables

DocBook supports HTML tables as well as CALS tables (that is, ``traditional'' DocBook tables) starting from version 4.3. Therefore XMLmind XML Editor also supports both table models. See Appendix A, *Table rendering* in *XMLmind XML Editor - XHTML Support* for details.

The only limitation is that mixing both HTML and CALS content models in the same `table` or `informaltable` is *absolutely not supported* by table rendering code and by table editing commands, even if this is allowed according to the DTD V4.3.

Example 1: an `informaltable` contains `tr` child elements. In such case, the `informaltable` is an HTML table. Setting attribute `frame` to `topbot` on this `informaltable` will have absolutely no visual effect.

Example 2: a `table` has a child `tgroup` element which itself contains a `tbody` with `row`/`entry` descendants. In such case, the `table` is a CALS table. Adding a `thead` having `tr`/`td` descendants before the `tbody` of the `tgroup` would lead to catastrophic results. Fortunately, the DocBook configuration of XMLmind XML Editor makes it hard to do this unintentionally.

5. Adding equations to your DocBook document

MathML

MathML support requires the "**MathML support**" and the "**JEuclid image toolkit plug-in**" add-ons to have been installed using **Options** → **Install Add-ons** in *XMLmind XML Editor - Online Help*.

- Use the "**Add image**" [20] toolbar button to add an image to your DocBook document. The image file pointed to must have a "`.mml`" file extension and must contain valid MathML presentation markup. Example:

```
<figure>
  <title>Proving ...</title>
  <mediaobject>
    <imageobject>
      <imagedata fileref="equations/eq1.mml"/>
    </imageobject>
  </mediaobject>
</figure>
```

Figure 3. Proving the Pythagorean theorem

$$\begin{aligned} (a+b)^2 &= c^2 + 4 \cdot \left(\tfrac{1}{2} ab\right) \\ a^2 + 2ab + b^2 &= c^2 + 2ab \\ a^2 + b^2 &= c^2 \end{aligned}$$

- Or, use the "**Add MathML equation**" [20] tool bar button to embed MathML presentation markup in your DocBook 5 document. See also XMLmind XML Editor - Support of MathML 2.0. Example: $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

```
<inlineequation><math display="inline" xmlns="http://www.w3.org/1998/Math/MathML">
```

```
  <mrow>...</mrow>
</math></inlineequation>
```

TeX/LaTeX math

TeX/LaTeX math support requires the "**JLaTeXMath image toolkit plug-in**" add-on to have been installed using **Options** → **Install Add-ons** in *XMLmind XML Editor - Online Help*.

- Use the "**Add image**" [20] toolbar button to add an image to your DocBook document. The image file pointed to must have a "`.tex`" file extension and must contain mathematical formulas written in LaTeX, which are supported by JLaTeXMath. Example:

```
<figure>
  <title>Excerpts ...</title>
  <mediaobject>
    <imageobject>
      <imagedata fileref="equations/eqA.tex"/>
    </imageobject>
  </mediaobject>
</figure>
```

Figure 4. Excerpts from "User's Guide for the amsmath Package (Version 2.1)"

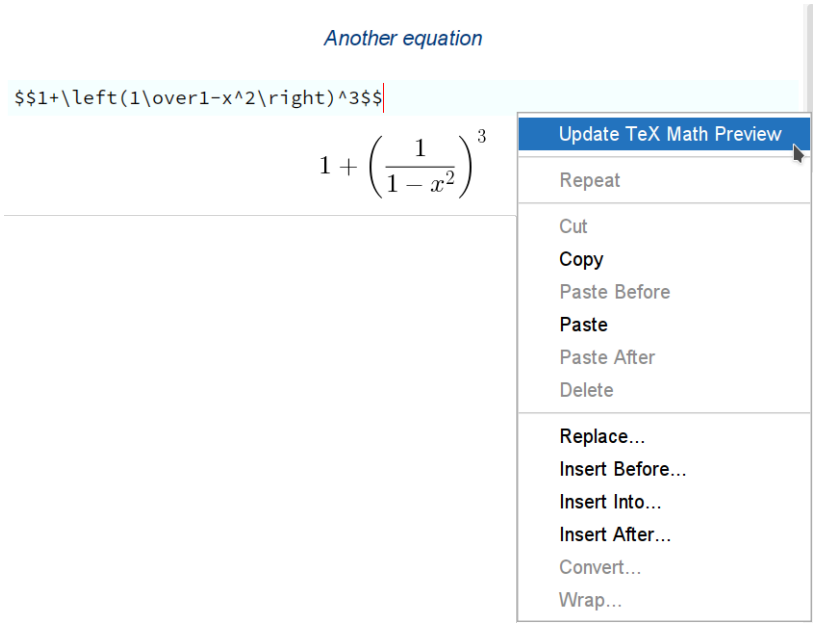
$$\partial_s f(x) = \frac{\partial}{\partial x_0} f(x) \quad \text{for } x = x_0 + Ix_1.$$

- Use the **Edit** tool in *XMLmind XML Editor - Online Help* to insert an `inlineequation(tex)`, `informalequation(tex)` or `equation(tex)` element template in your DocBook 4.5 or 5+ document. The newly inserted equation has a ready to use `<mathphrase role="tex">` child element. Example:
$\lim_{h \rightarrow 0 } \frac{f(x+h)-f(x)}{h}$

```
<inlineequation><mathphrase role="tex">
\(\lim_{h \rightarrow 0 } \frac{f(x+h)-f(x)}{h}\)
</mathphrase></inlineequation>
```


TeX/LaTeX math delimiters are optional. However specifying them may be useful as `\[`*math*`\]`, `$$`*math*`$$` or `\begin{`*env_name*`}`*math*`\end{`*env_name*`}` specify displayed math, while `\(`*math*`\)` or `$`*math*`$` specify in-line math.

If you need to refresh the preview after modifying the TeX/LaTeX math code found in the `<mathphrase role="tex">`, simply right-click this element and select item "**Update TeX Math Preview**" from the contextual menu.


Another equation
$$1+\left(1\over1-x^2\right)^3$$
$$1+\left(\frac{1}{1-x^2}\right)^3$$
Update TeX Math Preview
Repeat
Cut
Copy
Paste Before
Paste
Paste After
Delete
Replace...
Insert Before...
Insert Into...
Insert After...
Convert...
Wrap...