XMLmind XML Editor - DocBook Support

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

September 29, 2011

Most commands described below are also used by the `Simplified DocBook` and `Slides` document types.

1. DocBook menu

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.

Menu	**Item**	**Description**
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.
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	Insert a row before row containing specified cell. **Note** 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.
Cell	Increment Column Span	Increment the number of columns spanned by specified cell.

Menu	Item	Description
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).	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.

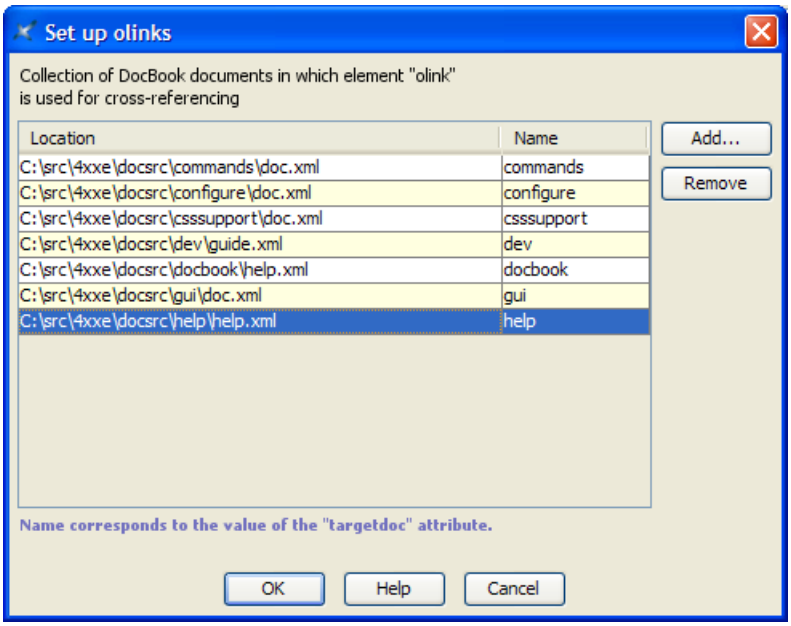
Other commands:

Set up olinks

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.

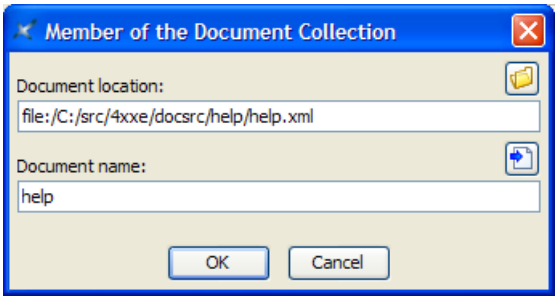
The Attributes tool can help you specify a value for the `targetdoc` attribute by listing[1] all the symbolic names of the target documents. Once the `targetdoc` attribute has been specified, the Attributes tool can help you specify a value for the `targetptr` attribute by listing all the IDs found in the target document. However for this facility to work, you first need to declare the collection of DocBook documents in which `olink` is used for cross-referencing.

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



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

[1] 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.



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/5xxe/docsrc/help/help.xml`".

   ## Tip

   You can mix DocBook v4+ and DocBook v5+ 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>
   ```

   ## Tip

   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 “olinked 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 and DocBook v5+ configurations. However you may add or remove members to/from your document collection at any time.

Paste After As
: The 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.

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`, `informalfigure`/`figure` and `informalexample`/`example`.

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

  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.

   ## Important

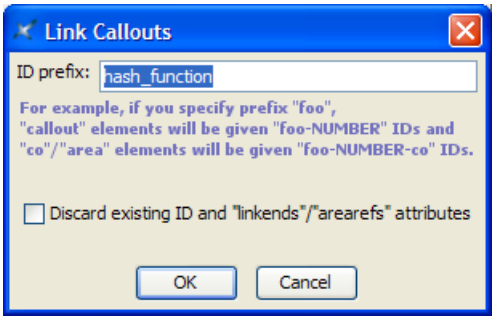
   Make sure to create exactly the same number of `co` and `callout` elements. This is needed because the *N*th `co` element will be linked to the *N*th `callout` element.

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

   In fact, you can select any element containing, at any nesting level, a sequence of `co` elements followed by a sequence of `callout` elements. For example, if your `programlisting` and `calloutlist` elements are contained in a `section` element, you can select just this section.

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[2] `programlisting` plus `calloutlist` elements.

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.

Insert or Edit `indexterm`

If the caret is anywhere inside an `indexterm` element, this menu item displays an `indexterm` editor dialog box [9] allowing to modify this `indexterm` element.

If the caret is not inside an `indexterm` element, this menu item displays an `indexterm` editor dialog box [9] allowing to create a new `indexterm` element and then to insert it at caret position.

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[3] 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">...
```

[2]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.
[3]That is, it is not possible to ``promote'' the *title* of a section.

```
  <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[4] of the section to be explicitly selected.

- 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:

```
<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>...
```

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

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

1.1. Convert Document sub-menu

This menu is present only in XMLmind XML Editor Professional Edition.

Using the profiling stylesheets

Profiling, or conditional text, means that you can create a single XML document with some elements marked as conditional. More information in DocBook XSL: The Complete Guide.

If you need to use the profiling XSL stylesheets rather than the plain XSL stylesheets, use Options → Customize Configuration → Document Conversion Preferences and check option "Use the profiling stylesheets".

Convert to HTML, Convert to HTML [one page], Convert to HTML [one page, no TOC]
Converts the document being edited to multi page or single page HTML.

Generating XHTML rather than HTML

If you prefer to generate XHTML rather than HTML, use Options → Customize Configuration → Document Conversion Preferences and check option "Generate XHTML rather than HTML".

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 Java Help
Converts the document being edited to a `.jar` file for use by the Java™ Help system.

For this command to work, the Java™ Help indexer, `jhindexer`, must have been declared as the helper application associated to files having a "`application/x-java-help-index`" MIME type. 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.

The `eclipse.plugin.name`, `eclipse.plugin.id`, `eclipse.plugin.provider` XSL style sheet parameters must have been specified using the Options → Customize Configuration → Change Document Conversion Parameters facility.

All HTML files as well as Eclipse's `plugin.xml` and `toc.xml` are generated in the same directory:

1. This directory must be a subdirectory of the Eclipse `plugins/` directory.

2. The name of this directory must be identical to the value of the `eclipse.plugin.id` XSL style sheet parameter.

Convert to EPUB
Converts the document being edited to EPUB.

Convert to RTF (Word 2000+), Convert to RTF [no TOC]
: 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+)., Convert to WordprocessingML [no TOC]
: 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+), Convert to Office Open XML [no TOC]
: 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+), Convert to OpenDocument [no TOC]
: 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.

Print PostScript, Print PostScript [no TOC]
: Converts the document being edited to PostScript® 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/), and then, sends the generated file to the chosen printer.

Convert to PDF, Convert to PDF [no TOC]
: 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.

Customizing the XSL style sheets used by the above Convert commands

The XSL style sheets used to convert DocBook, Simplified DocBook and Slides document to HTML, RTF, PostScript or PDF can be customized by specifying parameters, which are name/value pairs.

The parameters of the XSL style sheets used for DocBook and Simplified DocBook are documented in "DocBook XSL Stylesheet Documentation". The parameters of the XSL style sheets used for Slides are documented in "The Slides Document Type".

The easiest way to specify your own parameters is to use menu item Options → Customize Configuration → Change Document Conversion Parameters.

An alternate method, which should be reserved to experts, consists in defining named `parameterGroup` elements in your `customize.xxe` configuration file (for example, *`%APPDATA%`*`\XMLmind\XMLEditor5\addon\customize.xxe` on Windows and *`$HOME/`*`.xxe5/addon/customize.xxe` on Linux). How to do this is explained in Section 19, “parameterGroup” in *XMLmind XML Editor - Configuration and Deployment*.

The names of the `parameterGroups` supported by the Convert commands is found in the following tables:

DocBook Convert command	Name of associated parameterGroup
Convert to HTML	docb.toHTML.transformParameters
Convert to HTML (one page)	docb.toHTML1.transformParameters
Convert to HTML Help	docb.toHTMLHelp.transformParameters
Convert to Java Help	docb.toJavaHelp.transformParameters
Convert to Eclipse Help	docb.toEclipseHelp.transformParameters
Convert to RTF, WordprocessingML, OpenDocument, Office Open XML	docb.toRTF.transformParameters
Print PostScript, Convert to PDF	docb.toPS.transformParameters

Slides Convert command	Name of associated parameterGroup
Convert to HTML	slides.toHTML.transformParameters
Convert to RTF, WordprocessingML, OpenDocument, Office Open XML	slides.toRTF.transformParameters
Print PostScript, Convert to PDF	slides.toPS.transformParameters

Specifying an alternate XSLT style sheet

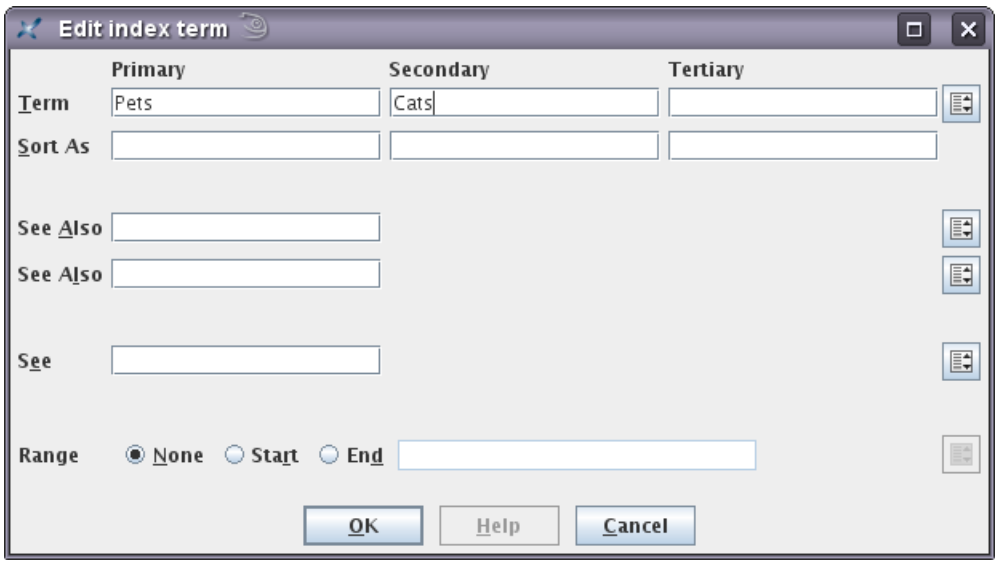
It is also possible to *extensively* customize the Convert commands by specifying alternate XSLT style sheets for them. This is explained in the documentation of the `process`/`transform` configuration element found in Section 1.5.1, “Using a custom XSLT style sheet” in *XMLmind XML Editor - Commands*.

About DocBook v5+

Everything explained in this section applies to DocBook v5+ as well as DocBook 4, simply replace the "`docb.`" prefix of the above `parameterGroups` by a "`db5.`" one.

1.2. 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.



Note that this editor does not allow to modify `indexterm` elements which are invalid, inconsistent or simply too complex. In such case, you'll have to revert using the normal editing tool (Edit tool, Attributes tool, etc) in order to modify such `indexterm` elements. However, this `indexterm` editor is sufficiently powerful to fulfill the vast majority of needs.

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\Primary=`Pet`.

- Back of the book index:

  ```
  P
  Pet
      Cat 26
  ```

  specify Term\Primary=`Pet`, Term\Secondary=`Cat`.

- Back of the book index:

  ```
  P
  "+" 54
  ```

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

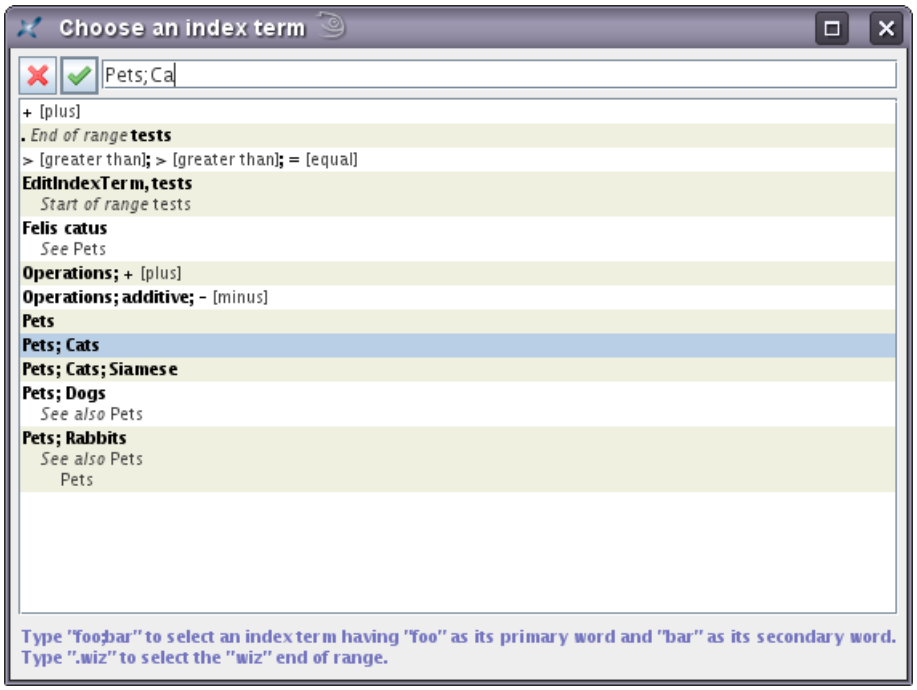
  ```
  Symbols
  "*" 53, 78
  "+" 54
  "-" 55, 91-95
  ...
  ```

- Back of the book index:

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

  specify Term\Primary=`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.



It is possible to specify up to two See Also entries for a Term.

- Back of the book index:

```
I
IT See Information Technology
```

specify Term\Primary=`IT`, See=`Information Technology`. (In the above example, notice that `IT` has no associated page number.)

Like See Also, the content of the See field must refer to an existing index entry. Unlike See Also, a See 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\Primary=`Operation`, Term\Secondary=`Additive`, Term\Tertiary="`+`", Sort As\Tertiary=`plus`.

   Then check Range/Start and give your `indexterm` element an ID by specifying "`plus_reference`" in the 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 Range/End and specify the same ID, "`plus_reference`", in the Range field. All the other fields must be left blank.

   Note that instead of typing "`plus_reference`" in the 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 Range row.

2. Custom bindings

Keystroke	Action
F2 Up	Same as menu item Move Up [5].
F2 Down	Same as menu item Move Down [5].
F2 Left	Same as menu item Promote [5].
F2 Right	Same as menu item Demote [6].
Enter	Inside a `para`/`simpara`, splits the `para`/`simpara` in two parts. Elsewhere, normal behaviour.
Del	At the end of a `para`/`simpara`, if there is no node or text selection, joins this `para`/`simpara` with following sibling `para`/`simpara` if any. Otherwise, normal behaviour.
BackSpace	At the beginning of a `para`/`simpara`, if there is no node or text selection, joins this `para`/`simpara` with preceding sibling `para`/`simpara` if any. Otherwise, normal behaviour.
Ctrl+**Enter**	Inside a `para`, `simpara`, `listitem`, `callout`, `step`, inserts same element after this one. Elsewhere, no effect. **Note** If the `para`/`simpara` is the first child of a `listitem`, `callout` or `step`, it is a `listitem`, `callout` or `step` which is inserted, not a `para`/`simpara`. If, in such case, you want to insert a new `para`/`simpara`, simply press the **Enter** key at the end of the paragraph.
Shift+**Ctrl**+**Enter**	Inside a `para`, `simpara`, `listitem`, `callout`, `step`, inserts same element before this one. Elsewhere, no effect.
Application Event	**Action**
`drop`	On an `ulink` element, changes the value of attribute `url` to the dropped string. Elsewhere, considers that the dropped string is a filename or an URL and therefore, attempts to open the corresponding document in the editor.

3. 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`.

3.1. 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 Section 4, “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.