



OpenOffice.org 2.0

Understanding and Authoring Online Help

Contents

How OpenOffice.org Help Works.....	5
Help Ingredients.....	5
Installed Help Files.....	9
Application Help Calls.....	12
Structure of the CVS Help Module.....	13
Building the Help Set.....	14
Help File XML format Basics.....	19
Basic Document Structure.....	19
Using Variables.....	19
Paragraph Roles.....	20
Using Bookmarks.....	20
Switching Content.....	22
Embedding Content.....	24
Images and Icons.....	26
Localization Information.....	27
Help File XML Reference.....	29
Common Attributes.....	29
Elements.....	31
Authoring Help With OpenOffice.org	71
Setting Up the Environment	71
Editing Help Files - Basics.....	75
Character Formatting.....	77
Working With the Help Files.....	77
Sections and Paragraphs.....	80
Tables	86
Lists.....	87

Working with Images.....	88
Embedding Content	90
Linking.....	91
Meta Data	91
Bookmarks.....	93
Switching Content.....	96
Miscellaneous.....	100
Troubleshooting.....	101
Appendix.....	103
XML Help DTD.....	103
Glossary.....	107

How OpenOffice.org Help Works

This chapter gives a high level overview of the OpenOffice.org Help system. Understanding the basic principles is helpful when working with help files.

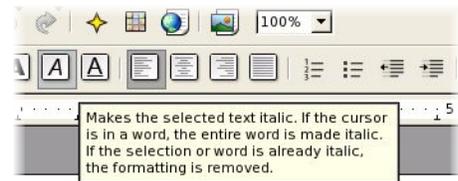
Help Ingredients

The OpenOffice.org help system comprises different help features which are explained in detail in the following sections.

Extended Tips

Extended Tips are "bubbles" on the application UI that give a short reference text for an element.

Display of extended tips is enabled by choosing **Help – Extended Tips** or by pressing `Shift+F1`.



The display of an extended tip for a particular UI element is triggered by resting the mouse over that element for a short amount of time (approx. 1 second). On moving the mouse the extended tip disappears.

When the extended tips were enabled by pressing `Shift+F1` the tips are displayed immediately without any delay. This mode is left when a mouse button is clicked.

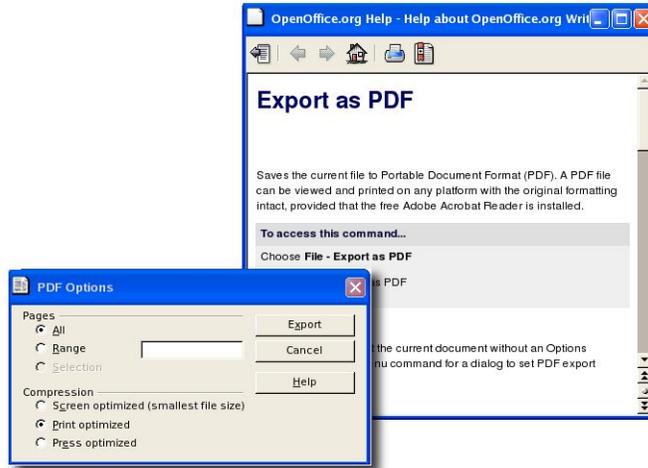
Extended tips use *Help IDs* that are assigned to UI elements to find the correct text for a UI element. The text itself is defined in the help files inside the `ahelp` element. For more information about the structure of the help files, please refer to chapter 2: "*Help File XML format Basics*" on page 19.

Context-Sensitive Help

OpenOffice.org Help is context-sensitive, that is, the help viewer displays reference information or instructions for the current application context when the help is called from within the application.

Context-sensitive help is invoked by pressing F1 or clicking the **Help** button in a dialog.

Help IDs are used to identify the context. A lookup table is used to find the correct anchor inside the help file set (see also *Application Help Calls* on page 12).

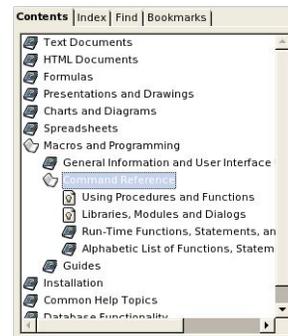


Hierarchical List Of Contents

There is a hierarchical list of help contents available from the **Contents** tab page of the help viewer. This is not a table of contents like in a book but a selection of help topics sorted by different application/document types and task groups.

Help topics can appear more than once if they fit into multiple application/task groups. Currently, these contents trees are manually compiled and saved in *.tree files.

In future, these contents lists can be defined within the help files themselves. The *.tree files will then be created when the help is compiled in the software build cycle.



Note, that although the corresponding elements are included in the help format these are *not yet* evaluated by the help compiler. The tree files must still be generated *manually*.

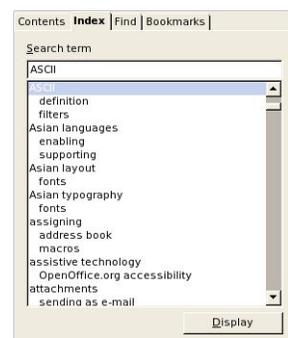
Index Of Keywords

The **Index** tab page of the help viewer contains a two-level keyword index. These two levels allow for a basic grouping of keywords. The index is displayed per help module.

After selecting an OpenOffice.org help module from the dropdown list at the top left of the help viewer the corresponding list of keywords is loaded.

Entering a search term directly jumps to the next suitable first-level entry in the index list. You can directly jump to a second level entry by using ";" (semicolon plus space) as a delimiter.

The keywords are defined inside the help files as `bookmarks`. See also section *Bookmarks* on page 93.

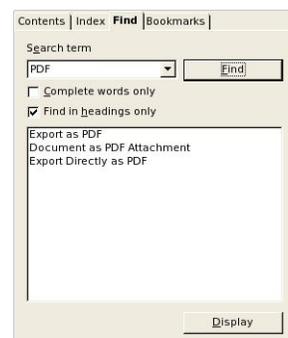


Full-Text Search

The **Find** tab page offers the possibility of searching through the help content. You can only search through one help module at a time.

By default, the search engine searches for case-insensitive substrings that appear anywhere in a help file. You can restrict the search scope by specifying to search complete words only and to only search headings in help files.

The results are displayed sorted by search rank showing the best matches at the top of the list.

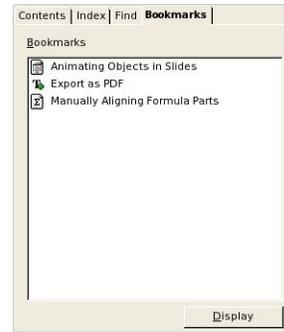


Bookmarks

The **Bookmarks** tab page lists user-defined bookmarks to help pages. User-defined bookmarks from all help modules go to this list. The icon next to a bookmark designates the help module to which the bookmark belongs.

Double-clicking the bookmark takes you back to the corresponding help page.

Bookmarks can be named individually.



Don't confuse these bookmarks with the `bookmark` element in the help XML format.

Installed Help Files

On installation, a `help` directory is created as child of the main OpenOffice.org directory. It contains all global files (currently only `main_transform.xsl`) and one or more subdirectories with language-dependent files. The language directories are designated by the ISO codes for language and country, e.g. `en-us` for US-English. If the ISO code for language and country is the same, only the language ISO code is used, e.g. `de` instead of `de-DE`. The contents of this language directory are as follows:

<code>help/</code>	The main help directory
<code>help/main_transform.xsl</code>	The main transformation style sheet (see <i>The Main Transformation Style Sheet</i> below)
<code>help/{lang}</code>	The language dependent help files
<code>help/{lang}/*.css</code>	The cascading style sheets for displaying the help in the help viewer (see <i>The Cascading Style Sheets</i> below)
<code>help/{lang}/err.html</code>	The error file. This file will be called whenever a help page could not be found.
<code>help/{lang}/shared.jar</code>	The help file archive for shared help files (see <i>Help Section Archives</i> below)
<code>help/{lang}/shared.tree</code>	The contents file for shared help files (see <i>Help Module Contents Files</i> below)
<code>help/{lang}/schart.jar</code>	The help file archive for help files dealing with charts (see <i>Help Section Archives</i> below)[1]
<code>help/{lang}/schart.tree</code>	The contents file for help files dealing with charts (see <i>Help Module Contents Files</i> below)[1]
<code>help/{lang}/{module}.cfg</code>	The configuration files for a help module (see <i>Help Module Configuration Files</i> below)
<code>help/{lang}/{module}.db</code>	The lookup tables for a help module (see <i>Help Module Lookup Tables (Databases)</i> below)
<code>help/{lang}/{module}.ht</code>	The extended tips for a help module (see <i>Help Module Extended Tip Files</i> below)
<code>help/{lang}/{module}.idx/</code>	The fulltext search index for a help module
<code>help/{lang}/{module}.jar</code>	The help file archive for a help module (see <i>Help Section Archives</i> below)
<code>help/{lang}/{module}.key</code>	The index file for a help module (see <i>Help Module Index Files</i> below)
<code>help/{lang}/{module}.tree</code>	The contents file for a help module (see <i>Help Module Contents Files</i> below)

Table 1: Help files that are installed.

[1] This is a legacy remainder of an older StarOffice help structure.

Help Modules And Help Sections

The help is divided into six different help modules that can be selected using the drop down list at the top left of the help viewer. Each help topic file has a scope that consists of one or more *help sections* and includes the corresponding help topic file archives *.jar. In the help file viewer, the index and the full text search work within this scope only.

The distinction between help *module* and help *section* is probably confusing and should be overcome in the future. Basically, a help *section* contains all files that are found inside the `text/{section}` path of the help source. Each help section produces a *.jar archive containing all help files in that path. A help *module* takes one or more sections and combines them to form the scope of a module.

Help module	Help Sections (Scope)
BASIC	sbasic + shared
Calc	scalc + shared + schart
Draw	sdraw + simpress + shared + schart
Impress	sdraw + simpress + shared + schart
Math	smath + shared
Writer	swriter + shared + schart

Table 2: OpenOffice.org help modules and scopes

From the table above it follows that the scope for the Writer help module includes all help files from `swriter.jar`, `shared.jar` and `schart.jar`. Each help module has a set of six files (`cfg`, `db`, `ht`, `jar`, `key`, `tree`) and an *.idx directory associated with it except for Draw which has no `sdraw.tree` file.

Help Module Configuration Files

The configuration files *.cfg are ASCII files with `parameter=value` pairs with configuration information. They are created and maintained manually:

```
Title=%PRODUCTNAME Writer Help
Copyright=Copyright 2004
Language=en-US
Order=2
Start=text/swriter/main0000.xhp
Heading=headingheading
Program=WRITER
08.11.03 01:23:00
```

`title` specifies the help module title as displayed in the drop down list at the top left of the help viewer.

`copyright` is a copyright string.

`Language` specifies the help language for the help module.
`Order` was used in an earlier implementation and is deprecated.
`Start` defines the start page for a help module.
`Heading` defines an internal value that is used by the full text search engine.
`Program` specifies the application name that will be used for switching content (see *Switching Content* on page 96)
The last line contains the creation date. Use of this is deprecated.

Help Module Contents Files

The contents files `*.jar` contain the help topic files for a help section (see *Help Modules and Help Sections* above). It is an archive file with a subdirectory structure that contains all help XML files after compilation.

Help Module Lookup Tables (Databases)

The lookup tables `*.db` are Berkeley databases that contain a lookup table used by the help application to find a help page to display for a given help ID.

The data for that table come from the `bookmark` elements in the help files (see *Bookmarks* on page 93).

Help Module Extended Tip Files

These files `*.ht` are Berkeley databases that contain the extended help tip text for all Help IDs. The application uses these files to fetch the text for an extended tip for a given Help ID.

The data for that table come from the `bookmark` elements in conjunction with the `ahelp` elements in the help files (see *Bookmarks* on page 93).

Help Section Archives

These files `*.jar` are zip archives that contain the help files in the directory structure as defined in the help source tree. There is one archive per help *section* (`sbasic`, `shared`, `sdraw`, `simpres`, `scal`, `schart`, `swriter`, `smath`). Each help *module* comprises more than one help section (see *Help Modules and Help Sections* above).

Help Module Index Files

These files `*.key` are Berkeley databases that contain the index entries for the help modules.

The data for that table come from the `bookmark` elements in the help files (see *Bookmarks* on page 93).

The Main Transformation Style Sheet

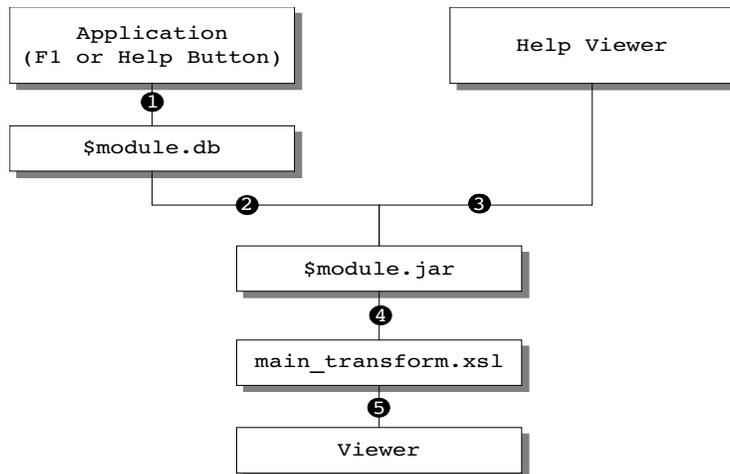
This file `main_transform.xsl` is global for all languages and help files and is used for finally transforming the xml help file to yield a html file that is displayed in the help viewer.

This style sheet is responsible for converting XML help elements and classes into HTML elements and classes. The overall layout of the help file is specified using this style sheet.

The Cascading Style Sheets

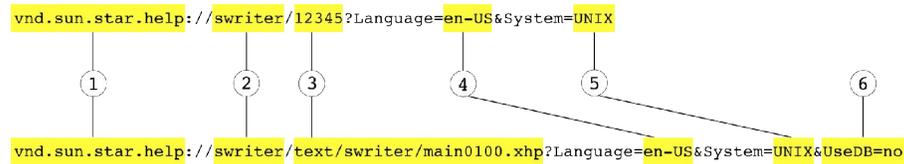
These cascading style sheets are language dependent and describe the formatting style for the help page. The OpenOffice.org help viewer only understands some basic CSS2 commands.

Application Help Calls



- 1 When F1 or a help button is pressed in an OpenOffice.org application, a help request is sent as an URL to the help content provider.
- 2 The help ID is resolved to a help file using the help module "database" for the application (`$module.db`).
- 3 When another help file is called from within the help the URL sent to the help content provider contains the file path. There is no need for resolving the ID.
- 4 The help file is extracted from the corresponding help file archive (`$module.jar`).
- 5 The extracted help file is transformed into HTML using the `main_transform.xsl` style sheet and sent to the help viewer for display. The stylesheet `main_transform.xsl` controls all conversion from xsl to html and must be adjusted whenever new elements, attributes, or attribute values must be taken account of.

The URLs sent to the help content provider have two forms:



- ① The protocol `vnd.sun.star.help` for the help files.
- ② The application module from which the help was called.
- ③ The numerical help ID or the physical path of the help file within the help file archive.
- ④ The current language.
- ⑤ The current system.
- ⑥ An attribute identifying the URLs that need no resolution using the database.

Structure Of The CVS Help Module

The help source files and all helper files are located in the CVS module `helpcontent2`. The directory layout is as follows:

<code>helpcontent2/</code>	The module's main directory
<code>helpcontent2/helpers</code>	Files that are not used by the help content itself, like the DTD for the XML help format.
<code>helpcontent2/helpers/helpauthoring</code>	The help authoring environment for OpenOffice.org, see Chapter 4: <i>Authoring Help With OpenOffice.org</i> on page 71.
<code>helpcontent2/source</code>	The help source files that are used to build the help.
<code>helpcontent2/source/auxiliary</code>	Auxiliary files that do not contain help content but are still needed for building the help, style sheets, configuration files.
<code>helpcontent2/source/text</code>	The help content source files, the makefiles for the help compiler, and the localized content. Every subdirectory contains its own makefile and a file with all localized content.
<code>helpcontent2/source/text/sbasic</code>	Help files specific to BASIC (and the IDE).[2]
<code>helpcontent2/source/text/scalc</code>	Help files specific to the Calc module.[2]
<code>helpcontent2/source/text/schart</code>	Help files specific to charts.[2]
<code>helpcontent2/source/text/sdraw</code>	Help files specific to the Draw module.[2]
<code>helpcontent2/source/text/shared</code>	Help files common to one or more modules.[2]
<code>helpcontent2/source/text/simpress</code>	Help files specific to the Impress module.[2]

[2] The subdirectory structure of this directory has historical reasons.

helpcontent2/source/text/smath	Help files specific to the Math module.[2]
helpcontent2/source/text/swriter	Help files specific to the Writer module.[2]
helpcontent2/prj	The build lists.
helpcontent2/util	The linker makefiles for the help sections.

Note, that the help images are no longer part of the CVS module. Starting with OpenOffice.org 2.0, application icons are directly taken from the `images.zip` repository in the `share/config` directory.

Images specific to the help need to be added to the CVS `res` module in the `helpimg` subdirectory. These will then also be included in the `images.zip` archive.

Building The Help Set

Setting Up A Build Environment

This is described on tools.openoffice.org.

Makefiles For The Help

The `helpcontent2` module contains three type of makefiles:

- Makefiles for compiling the help source files

These makefiles are found in the `helpcontent2/source/text` directories. Every subdirectory that contains help files to be compiled has a corresponding makefile, for example (*shortened for clarity*):

```
*****
*****
# edit to match directory level
PRJ      = ../..$/../..$/..
# same for all makefiles in "helpcontent2"
PRJNAME = helpcontent2
# edit to match the current package
PACKAGE = text/sbasic/guide
# unique name (module wide);
# using a modified forme of package should do here
TARGET  = text_sbasic_guide
# edit to match the current module
MODULE  = sbasic

# --- Settings -----
.INCLUDE : $(PRJ)/settings.pmk
.INCLUDE : settings.mk
```

```
# this list matches the *.xhp files to process
HZIPFILES = \
  control_properties.hzip \
  create_dialog.hzip \
  insert_control.hzip \
  sample_code.hzip \
  show_dialog.hzip

# --- Targets -----

.INCLUDE : target.mk
.INCLUDE : $(PRJ)/makefile.pmk
```

You find a template for this makefile in `helpcontent2/helpers`. This template is used when the makefiles are created using the `createmakefile.pl` script in `helpcontent2/helpers`.

- Makefiles for linking the compiled files.

These makefiles are found in the subdirectories of `helpcontent2/util` (the directory itself contains the third type of makefile), for example (*shortened for clarity*):

```
*****
*****

# edit to match directory level
PRJ          = ../..
# same for all makefiles in "help2"
PRJNAME      = help2
# unique name (module wide);
# using a modified form of package should do here
TARGET       = sbasic_util

# --- Settings -----

.INCLUDE : $(PRJ)/settings.pmk
.INCLUDE : settings.mk

common_build_zip:=
ziplgeneratedlangs=TRUE
ZIP1TARGET=sbasic_xhp
ZIP1FLAGS= -u -r
ZIP1DIR=$(MISC)/$(LANGDIR)
ZIP1LIST=$(LANGDIR)/text$/sbasic$/* -x "*.dphh*" -x "*.hzip"

LINKNAME=sbasic
LINKADDEDFILES= \
  -add sbasic.cfg $(PRJ)/source$/auxiliary$/LANGUAGE$/sbasic.cfg \
  -add sbasic.tree $(PRJ)/source$/auxiliary$/LANGUAGE$/sbasic.tree \
  -add sbasic.jar $(BIN)/sbasic_xhp_LANGUAGE.zip

LINKADDEDEPS= \
  $(PRJ)/source$/auxiliary$/LANGUAGE$/sbasic.cfg \
  $(PRJ)/source$/auxiliary$/LANGUAGE$/sbasic.tree \
  $(BIN)/sbasic_xhp_LANGUAGE.zip
```

```
LINKLINKFILES= \
    text$/sbasic$/guide$/control_properties.hzip \
    text$/sbasic$/guide$/create_dialog.hzip \
    text$/sbasic$/guide$/insert_control.hzip \
    text$/sbasic$/guide$/sample_code.hzip \
    text$/sbasic$/guide$/show_dialog.hzip \

# --- Targets -----

.INCLUDE : target.mk
.INCLUDE : $(PRJ)$/util$/target.pmk
```

You find a template for this makefile in `helpcontent2/helpers`. This template is used when the makefiles are created using the `createmakefile.pl` script in `helpcontent2/helpers`.

- A makefile for creating the stylesheet archive in `helpcontent2/util` (*shortened for clarity*):

```
*****
*****

# edit to match directory level
PRJ          = ..
# same for all makefiles in "helpcontent2"
PRJNAME     = helpcontent2
# unique name (module wide);
# using a modified forme of package should do here
TARGET      = plain_util

# --- Settings -----

.INCLUDE : $(PRJ)$/settings.pmk
.INCLUDE : settings.mk

ZIPI1TARGET=helpxsl
ZIPI1FLAGS= -u -r
ZIPI1DIR=$(PRJ)$/source$/auxiliary
ZIPI1LIST=main_transform*.xsl

# --- Targets -----

.INCLUDE : target.mk

ALLTAR : $(COMMONBIN)$/helpimg.ilst

$(COMMONBIN)$/helpimg.ilst: helpimg.ilst
    $(COPY) $< $@
```

Help Build Process

The file `helpcontent2/prj/build.lst` defines which directories are to be built using a directory's makefile. Dependencies (which directories need to be built first) are also defined here.

Initiate building of the help by issuing the command `build` while in the `helpcontent2` directory.

1. The help files from `helpcontent2/source/text` are compiled and written to the `misc` subdirectory of the platform directory of the output tree. This step produces a set of `*.hzip` files and dependency files `*.dphh`. These files are the particles that are used to create the help modules in the next – the linking – step.
2. The compiled help files are taken from the `misc` directory and linked into a zip archive. Other files are added from the `helpcontent2/source/auxiliary` directory to that archive as defined in the makefiles of the subdirectories in `helpcontent2/util`. This results in one zip archive per help module and language in the `bin` subdirectory of the platform directory of the output tree.
3. The `helpxsl.zip` archive is built according to the makefile in `helpcontent2/util`.
4. All archive files are delivered according to the `d.lst` file in `helpcontent2/prj`.

Adding A Help File To Or Removing A Help File From The Set Of Help Files

The makefiles need to be adjusted to reflect the changes you made to the set of files. If you added a new file, add this to the makefile of its directory and to the link makefile (in `helpcontent2/util/*`) of any module that will contain the file. If you deleted a help files remove it from the makefile of its directory and from the link makefile (in `helpcontent2/util/*`) of any module that contains the file.

If you rebuild the help after help files have been deleted or dependencies (references) between the have been changed you need to remove all dependency files from the `misc` directory that are no longer valid. To be perfectly safe, you can remove the complete output tree for the platform of the `helpcontent2` module.

Help Images

Images that are used inside the help are stored in different modules and accessed by the help viewer using the `images.zip` archive on runtime. Therefore, you need to add help images, like screenshots to the `helpimg` directory of the `res` module. Including the help images to the `images.zip` repository is controlled by the `help.ilst` file that is found in the `util` directory of `helpcontent2`.

See also *Working with Images* on page 88.

Help File XML Format Basics

Basic Document Structure

The basic skeleton of a valid help file for OpenOffice.org consists of a `helpdocument` root element with one `meta` and one `body` sub-element containing the content (`body`) and meta information (`meta`). The minimum information is a topic title and the filename inside the elements

- `/helpdocument/meta/topic/title` and
- `/helpdocument/meta/topic/filename` .

The following example shows a valid (but empty) help file:

```
<?xml version="1.0" encoding="UTF-8"?>
<helpdocument version="1.0">
  <meta>
    <topic id="textswriter01012345xhp" indexer="include" status="PUBLISH">
      <title xml-lang="en-US" id="tit">Topic Title</title>
      <filename>text/swriter/01/012345.xhp</filename>
    </topic>
  </meta>
  <body>
  </body>
</helpdocument>
```

The help file extension is `xhp`.

Using Variables

In the help files the following variables are used to designate the name and the version of the product. This is to allow for correct branding of the product (for example, OpenOffice.org vs. StarOffice). You must never use the literal name of the product but instead one of the following variables^[3]:

`%PRODUCTNAME` designates the name of the product, for example **OpenOffice.org**.

[3] In addition to these variables, the following two variables are still used in the help files for legacy reasons: `$_officeName` and `$_officeVersion`.

`%PRODUCTVERSION` designates the current version of the product, for example **2.0**.

Both variables are replaced by the main transformation style sheet `main_transform.xml` (see page 12) when the help is displayed. The corresponding information is taken from the application's configuration information.

Paragraph Roles

The main element carrying content is a `paragraph`. There is no heading element, instead all headings are treated as paragraphs with a heading *role*. The `role` attribute defines the role of a paragraph with the `paragraph` role being the standard. The values for the role attribute are not defined in the DTD.

During the conversion process (XML→HTML) the `role` attribute is mapped to a `class` attribute of the corresponding HTML element allowing to influence the layout of the corresponding paragraph using style sheets.

The following roles are currently suggested and defined in the help authoring template. More roles can be defined as required (see also *Paragraph Formatting* on page 83):

Role	Description	Converts to...
<code>paragraph</code>	A standard paragraph	<code><p class="paragraph">...</p></code>
<code>heading</code>	A heading If this role is assigned to a paragraph, the heading level has to be specified using the <code>level</code> attribute of the <code>paragraph</code> element.	<code><h1>...</h1></code> to <code><h6>...</h6></code>
<code>note</code>	A note	<code><p class="note">...</p></code>
<code>warning</code>	A warning	<code><p class="warning">...</p></code>
<code>tip</code>	A tip	<code><p class="tip">...</p></code>
<code>code</code>	A code fragment	<code><p class="code">...</p></code>
<code>example</code>	An example	<code><p class="example">...</p></code>
<code>tablehead</code>	A table head (first rows)	<code><p class="tablehead">...</p></code>
<code>tablecontent</code>	Table contents	<code><p class="tablecontent">...</p></code>

If you use other roles, you must ensure that they are taken account of by the CSS files that define the help file display format.

Using Bookmarks

The help uses one unified bookmarking system to set anchors inside the help files which are used by the **Index** tab, the **Contents** tab and for context-sensitive help.

A bookmark has a `branch` attribute representing the purpose of the bookmark. Currently there are three branches defined: `contents`, `index`, and `hid`.

To define an anchor for a bookmark inside a help document, the element `<bookmark>` has to be declared at the place the bookmark will point to. The `branch` attribute specifies the type of bookmark to be defined (a content entry, an index entry, or a help id), while the sub-element `bookmark_value` contains the visible bookmark text.

The only child element that is allowed inside the `bookmark_value` is `embedvar` to allow embedding of commonly used titles for content nodes or index entries (for examples of the usage of embedded fragments inside bookmark values, refer to the next sections).

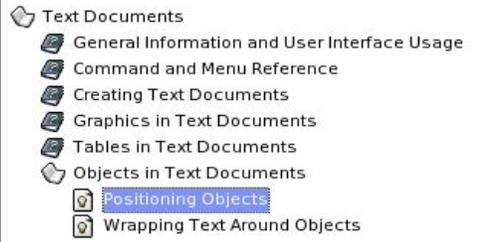
"Contents" Branch

Content entries are displayed on the Content tab page of the help viewer. The `branch` attribute takes the value `"contents"`.

The bookmark value can contain any number of levels separated by slashes with the last part of the bookmark value serving as the entry and the other parts serving as nodes.

Note, that currently the contents branch *is not* implemented in the help build process.

Example

<pre><bookmark branch="contents"> <bookmark_value xml-lang="en-US"> Text Documents/ Objects in Text Documents/ Positioning Objects </bookmark_value> </bookmark></pre>	
--	--

A bookmark value may also contain embedded fragments for node titles. This reduces redundancy, maintenance effort, and the risk of introducing errors through typos. This can be avoided if the top level entries for the content tree are defined separately:

```
<variable id="textdocs">Text Documents</variable>
<variable id="objtextdocs">Objects in Text Documents</variable>
```

and embedded as text fragments:

```
<bookmark_value>
  <embedvar href="/text/shared/00/variables.xhp#textdocs"/>/
  <embedvar href="/text/shared/00/variables.xhp#objtextdocs"/>/
  ...
</bookmark>
```

"Index" Branch

Index Entries are displayed on the Index tab page of the help viewer. The `branch` attribute takes the value "index" Currently, index entries may contain two levels separated by a **semicolon**.

Example

```
<bookmark branch="index" id="bm_1234" xml-
lang="en-US">

  <bookmark_value>
    editor;contour editor
  </bookmark_value>

</bookmark>
```



As with content entries, the bookmark values for index entries can contain embedded text fragments by using the `embedvar` element which is particularly useful if names of UI elements are used which may be subject to change.

"hid" Branch

Help IDs are never displayed but instead trigger context-sensitive help inside OpenOffice.org. The `branch` attribute takes the value "hid" and in addition contains the help id associated with the bookmark.

```
<bookmark id="bm_9876" xml-lang="en-US" branch="hid/HID_SOME_HELP_ID"/>
```

A bookmark for a given help id may only be used *once* inside the help files since the bookmark defines the entry point for the help viewer when context-sensitive help is triggered from the UI either through the use of the **F1** key or the **Help** button.

There are two types of help ids currently used in the help files:

- **Symbolic names**, like `SID_FM_CONVERTTO_IMAGECONTROL`
- **UNO command names**, like `.uno:InsertCtrl`

For details on determining a help ID for a UI element, see *Determining A Help ID* on page 95.

Switching Content

In some cases it is necessary to distinguish between different platforms or applications when displaying the help. For example, on one platform a key stroke to achieve a certain action may differ from the key stroke used on other platforms. To avoid duplicating large amounts of text and to reduce redundancy, switching elements are available which are used to select the correct portion of the content at runtime.

The help content provider sends additional information along with a help request stating the current platform, language and application context. This information can be evaluated using the switch constructs to display the corresponding information.

There are two types of content switching:

- Switching complete paragraphs or sections
- Switching text fragments inside paragraphs

Currently, the following values are used for the select attribute of a `switch` and `switchinline` element to specify the switching context:

Value	Switching context	Example/Comment
<code>sys</code>	Operating System	Switching content for Unix, Windows, or Mac platforms.
<code>appl</code>	Application	Switching content for different OpenOffice.org applications (Writer, Calc,...) in files that are common to multiple applications.
<code>distrib</code>	Distribution ^[4]	Switching content for different distributions, like OpenOffice.org and StarOffice, which contains extra commercial features.

The following values are used for the select attribute of a `case` and `caseinline` element within a given switching context:

Switching Context	Values
Operating System (<code>sys</code>)	WIN UNIX MAC
Application (<code>appl</code>)	WRITER CALC DRAW IMPRESS MATH BASIC CHART

Switching Complete Paragraphs Or Sections

This type is used, for example, if contents of a paragraph differ considerably on different platforms or for different applications, or if a certain paragraph or section is only applicable to a certain platform or application.

While for example, mounting a CD-ROM drive may be a necessary step on a Unix system it is usually not applicable to Windows computers. The `switch` element can be used to accomplish this distinction:

```
<switch select="sys">
  <case select="UNIX">
    <paragraph>Mount the cd rom drive.</paragraph>
  </case>
</switch>
```

[4] Note, that this switch is currently *not evaluated* in the main transformation step since the help content provider does not provide the necessary information. Currently, the main transformation style sheet uses the value of the product name to distinguish between open source and commercial distributions but this is only implemented for StarOffice and OpenOffice.org.

Switching Text Fragments Inside Paragraphs

This type is used if only small text fragments differ on different platforms or applications. A typical case is the use of shortcuts on different systems or the notation of file paths on different platforms.

While, for example, on Windows the standard installation path for OpenOffice.org may be something like "C:\Program Files\OpenOffice.org-2.0", it may be "~/OpenOffice.org-2.0" on a Unix system making it necessary to distinguish between the operating environments when talking about these paths. The `switchinline` element can be used to accomplish the distinction:

```
<paragraph>The software will be installed in the
<switchinline select="sys">
  <caseinline select="UNIX">
    ~/OpenOffice.org-2.0
  </caseinline>
  <caseinline select="WIN">
    C:\Program Files\OpenOffice.org-2.0
  </caseinline>
  <defaultinline>
    home
  </defaultinline>
</switchinline>
directory.</paragraph>
```

In the code example above there is also a default value defined (by using the optional `defaultinline` element) which will be shown if neither `UNIX` nor `WIN` is set as the platform value when calling the help.

Embedding Content

Another way of reducing redundant content is to define reusable text fragments and blocks that can be referenced from other places. The references are resolved at runtime when the help is displayed.

There are two ways of reusing contents by means of embedding:

- Embedding complete sections
- Embedding text fragments

Embedding Complete Sections

Single or multiple paragraphs may apply to more than one help file. For example, standard steps inside procedures can be written once and embedded in multiple places reducing maintenance and translation effort.

The URL for the reference takes the form `file#id`. If, for instance, the section with the id 12345 from the file `text/writer/01/012345.xhp` is to be embedded, the URL

would be `text/swriter/01/012345.xhp#12345`. The file name refers to the path and name that is stored in the jar files.

Complete sections can be embedded using the `embed` element. The section to be embedded is referenced using the attribute `id`, which must be unique within the file.

If, for example, multiple processes described in the help involve logging on to a computer, this particular step may be written once and embedded wherever required:

Example

Original location (filename: `original.xhp`):

```
<section id="logon">
  <paragraph id="par_id12345" role="paragraph" xml-lang="en-US">
    Log on to your computer using your user name and password.
  </paragraph>
</section>
```

Referenced location:

```
<paragraph id="par_id9876" role="heading" level="1" xml-lang="en-US">
  Starting %PRODUCTNAME
</paragraph>
<list>
  <listitem><embed href="original.xhp#logon"/></listitem>
  <listitem>
    <paragraph id="par_id9877" role="paragraph" xml-lang="en-US">
      Start %PRODUCTNAME</paragraph>
    </listitem>
  </list>
```

This results in something like

Starting OpenOffice.org

1. Log on to your computer using your user name and your password.
2. Start OpenOffice.org

Embedding Text Fragments

Text fragments may, for example, represent commonly used phrases or names of UI elements. These can be specified once and used in multiple places reducing maintenance and localization effort.

The URL for the reference takes the form `file#id`. If, for instance, the variable with the id 12345 from the file `text/swriter/01/012345.xhp` is to be embedded, the URL would be `text/writer/01/012345.xhp#12345`. The file name refers to the path + name that is stored in the jar files.

These fragments can be embedded using the `embedvar` element if they are previously defined as being variables so that they can be referenced. The text fragment to be

embedded is placed inside a `variable` element and assigned a unique id using the element's `id` attribute:

Original location (filename: `original.xhp`):

```
<paragraph id="par_id1234">Press the <variable id="btn_prnprev"><item type="button">Print Preview</item></variable> button.</paragraph>
```

The fragment can then be referenced in other locations using the `embedvar` element:

Referenced location:

```
<paragraph id="par_id9876">A preview can be shown using the <embedvar href="original.xhp#btn_prnprev"/> button.</paragraph>
```

Result:

```
A preview can be shown using the Print Preview button.
```

If, for example, the name of the button changes from "Print Preview" to "Show Preview" you only need to update one location to make the change available in all referenced locations.

You can also embed the content of paragraphs by referring to the **paragraph ID**. Note that only the contents of the paragraph are embedded. The paragraph formatting information is disregarded:

Referenced location

```
<paragraph id="par_id433122"><embedvar id="referenced.xhp#par_id9876"/>
```

Result:

```
A preview can be shown using the Print Preview button.
```

Images And Icons

All images must be placed inside `paragraphs`. The `image` element contains information about the image source in the `src` element and must be assigned a unique `id`. Every `image` element must also contain a child element `alt` that contains a short description of the image used if the visual content is not displayed or cannot be accessed by visually impaired users.

In addition to the `alt` element there is also an optional `caption` element that can take a long description as an image caption.

Starting with OpenOffice.org 2.0, the help retrieves all images from the central image repository `images.zip` which is available in the `share/config` directory of the OpenOffice.org installation. This archive contains all images that OpenOffice.org uses separated by modules. The OpenOffice.org Help fetches any icons displayed in the help files from here. Since this also is the place where the application fetched the icons to

display in the user interface, the icons in the help will always be in sync with the application, even if the `images.zip` archive contains a customized set of images.

The help itself also has a subdirectory inside the `images.zip` archive that contains all images that are specific to the help and only used by it, for instance screen captures. These images are stored under `res/helpimg` in the archive.

Localization Information

Content which is to be localized is found inside elements with the `xml-lang` attribute that contains the elements language code. Elements may be excluded from localization by specifying the `localize` attribute and setting it to `false`. Any such element *and all of its child elements* will be excluded from the localization process.

The special language code `x-comment` designates a comment that can be used to communicate with the localizers, e.g.

```
<paragraph id="par_id1234" xml-lang="x-comment">  
Comment to translators  
</paragraph>
```

All paragraphs contain an `l10n` attribute used to specify the localization status of the paragraph. This attribute was only used in the migration phase and is not evaluated. It can be used to store a paragraph authoring status to implement basic content management functionality.

Help File XML Reference

This chapter lists all elements of the XML help file DTD in alphabetical order as presented in the Document Type Definition in the Appendix.

The element sections presented here all share a common structure. The name of the element serves as heading and is followed by element details:

- Element Description and Purpose
- Attributes
- Parent Elements
- Child Elements
- Element Definition
- Element Example

Examples for elements may show an element within its parent or child context.

Common Attributes

The following attributes are common to several elements.

Xml-Lang

The `xml-lang` attribute designates elements that need localization. The localization process identifies elements to be localized by this attribute. It contains the language of the element it belongs as a combination of language ISO code (*ISO 639-1*) and country ISO code (*ISO 3166*) separated by a dash.

```
xml-lang="en-US"
```

All elements containing text to be translated have an `xml-lang` attribute: `alt`, `bookmark`, `caption`, `paragraph`, and `title`.

An element with the pseudo language code `x-comment` designates a comment. In this way, comments can be bound to existing elements that are localized by giving them the same `id` as the original element, e. g.

```
<paragraph xml-lang="en-US" id="someid">Pay US$ 23.99</paragraph>
```

```
<paragraph xml-lang="x-comment" id="someid">Be sure to convert the currency  
when localizing</paragraph>
```

You can also use the `comment` element to insert comments into the help file. But if they are outside an element that will be localized they will not be recognized by the localizers.

Localize

The `localize` attribute can only take the value `false` and designates elements which are excluded from the localization process. If an element contains this attribute set to `false` its contents and the contents of all child elements should not be translated. If the attribute contains any other value than `false` it will be ignored. The attribute is optional.

```
localize="false"
```

All elements containing text to be translated or can contain descending elements with text to be translated have an optional `localize` attribute: `alt`, `body`, `bookmark`, `bookmark_value`, `caption`, `list`, `listitem`, `paragraph`, `section`, `switch`, `table`, `tablecell`, `tablerow`, and `title`.

Id

The `id` element contains a unique string used to identify the element for localization and referencing purposes. The `id` must be unique within a help file so that referencing across files and relocating sections and paragraphs across files is possible.

```
id="some_unique_value_123"
```

All elements that can be embedded or have to be translated contain a mandatory `id` attribute: `image`, `bookmark`, `paragraph`, `section`, `table`, `title`, `topic`, and `variable`.

Valid characters for the `id` value are capital or small letters from a-z, numbers from 0-9, and the underscore, in any combination. Other characters are not allowed. [5]

[5] For historical reasons, the help files contain plenty of IDs that are not valid XML "id" types. Therefore, the `id` attribute is defined in the DTD to be of the type `CDATA`.

Elements

Ahelp

This element designates text which is to be used as extended tips (aka tool tips or active help). It may contain text (PCDATA) and child elements. It may only be used as a child of a paragraph.

Attributes

Attribute	Required	Contents	Values allowed	Description
hid	yes	CDATA		The symbolic help id for which the content is to be displayed.
visibility	no	fixed value	"hidden", "visible"	The visibility of the aHelp content inside the help viewer. If set to "hidden" the content is only visible in the extended tips popup.

Parent Elements

caseinline, defaultinline, paragraph, variable

Child Elements

comment, embedvar, br, emph, item, link, variable

Element Definition

```
<!ELEMENT ahelp (#PCDATA | embedvar | br | comment | emph | item |
                link | switchinline | variable)*>
<!ATTLIST ahelp
  hid CDATA #REQUIRED
  visibility (hidden | visible) #IMPLIED
  >
```

Example

```
<ahelp hid="HID_SOME_HID" visibility="hidden">
You will only see this text in the extended tips (bubble help, tool tips) for
the ui control with the help id HID_SOME_HID.
</ahelp>
```

Alt

This element is used to specify an alternative text for an image. It corresponds to the HTML attribute of the same name and may only contain PCDATA that is localized (no markup).

Attributes

Attribute	Required	Contents	Values allowed	Description
xml-lang	yes	CDATA		See <i>Common Attributes</i> on page 29.
id	yes	CDATA		A unique id to identify the element, see <i>Common Attributes</i> on page 29.
localize	no	fixed value	"false"	See <i>Common Attributes</i> on page 29.

Parent Elements

image

Child Elements

none

Element Definition

```
<!ELEMENT alt (#PCDATA)>
<!ATTLIST alt
  xml-lang CDATA #REQUIRED
  id CDATA #REQUIRED
  localize CDATA #IMPLIED
>
```

Example

```
<image src="img/imagefile.png" id="img_id1235">
<alt xml-lang="en-US" id="alt_id1235">Dialog File Open</alt>
</image>
```

Body

This element contains all help content information. It may itself not contain any PCDATA but only child elements.

Attributes

Attribute	Required	Contents	Values allowed	Description
localize	no	fixed value	"false"	See <i>Common Attributes</i> on page 29

Parent Elements

helpdocument

Child Elements

section, paragraph, table, comment, bookmark, switch, embed, list, sort

Element Definition

```
<!ELEMENT body (section | paragraph | table | comment | bookmark |
                switch | embed | list | sort)*>
<!ATTLIST body
  localize CDATA #IMPLIED
>
```

Example

```
<body>
<paragraph>This is the content of a help file</paragraph>
</body>
```

Bookmark

This element contains information about a bookmark used in the help files. The bookmark type is specified inside the `branch` attribute of the `bookmark` element while the bookmark value is defined in the child element `bookmark_value`. For more information about the bookmarking system in the help please refer to *Using Bookmarks* on page 20.

Attributes

Attribute	Required	Contents	Values allowed	Description
branch	yes	CDATA	"contents", "index", "hid"	The bookmark type specified by the branch inside the unified bookmarks tree. See <i>Using Bookmarks</i> on page 20 for details.
id	yes	CDATA		A unique id to identify the element, see <i>Common Attributes</i> on page 29.
xml-lang	yes	CDATA		See <i>Common Attributes</i> on page 29.
localize	no	fixed value	"false"	See <i>Common Attributes</i> on page 29.

Parent Elements

`body`, `case`, `default`, `section`, `topic`, `tablecell`, `listitem`

Child Elements

`bookmark_value`

Element Definition

```
<!ELEMENT bookmark (bookmark_value)*>
<!ATTLIST bookmark
  branch CDATA #REQUIRED
  xml-lang CDATA #REQUIRED
  id CDATA #REQUIRED
  localize CDATA #IMPLIED
>
```

Example

```
<bookmark branch="contents" xml-lang="en-US" id="bm_id1234">
  <bookmark_value>
    StarOffice Writer Help/Working with Fields/Editing Field Contents
  </bookmark_value>
</bookmark>
<bookmark branch="index" xml-lang="en-US" id="bm_id9876">
  <bookmark_value>
    Formulas/Exporting
  </bookmark_value>
</bookmark>
<bookmark branch="hid/12345"/>
```

Bookmark_value

This element contains the value of a bookmark (see *Using Bookmarks* on page 20 for details).

Parent Elements

bookmark

Child Elements

embedvar

Element Definition

```
<!ELEMENT bookmark_value (#PCDATA | embedvar)*>
```

Example

```
<bookmark branch="contents" xml-lang="en-US" id="bm_123">
  <bookmark_value>
    StarOffice Writer Help/Working with Fields/Editing Field Contents
  </bookmark_value>
</bookmark>

<bookmark branch="index/scalc" xml-lang="en-US" id="bm_543">
  <bookmark_value>
    Formulas/Exporting
  </bookmark_value>
</bookmark>

<bookmark branch="index/scalc" xml-lang="de-DE" id="bm_543">
  <bookmark_value>
    Formeln/Exportieren
  </bookmark_value>
</bookmark>
```

Br

This element can be used to place a manual line break. It works like the corresponding HTML element. The element itself is empty.

Attributes

none

Parent Elements

ahelp, caption, caseinline, defaultinline, paragraph, variable

Child Elements

none

Element Definition

```
<!ELEMENT br EMPTY>
```

Example

```
<paragraph>This line must have a<br/>manual<br/>line break.</paragraph>
```

Caption

This element specifies the (optional) caption of an image or a table.

Attributes

Attribute	Required	Contents	Values allowed	Description
xml-lang	yes	CDATA		See <i>Common Attributes</i> on page 29.
id	yes	CDATA		A unique id to identify the element, see <i>Common Attributes</i> on page 29.
localize	no	fixed value	"false"	See <i>Common Attributes</i> on page 29.

Parent Elements

image, table

Child Elements

embedvar, br, emph, item, link, switchinline, variable

Element Definition

```
<!ELEMENT caption (#PCDATA | embedvar | br | emph | item | link |
                  switchinline | variable)*>
<!ATTLIST caption
  xml-lang CDATA #REQUIRED
  id CDATA #REQUIRED
  localize CDATA #IMPLIED
>
```

Example

```
<table>
<caption xml-lang="en-US" id="cp_1234">
  List of all <item type="productname">StarOffice</item> slots.
</caption>
</table>
```

Case

This elements holds the cases of a `switch` statement.

Attributes

Attribute	Required	Contents	Values allowed	Description
select	yes	CDATA		Contains the value that is to be evaluated. See <i>Switching Content</i> on page 22 for more information.

Parent Elements

`switch`

Child Elements

`paragraph`, `table`, `comment`, `bookmark`, `embed`, `list`, `switch`, `section`

Element Definition

```
<!ELEMENT case (paragraph | table | comment | bookmark | embed |
                link | list | switch | section)*>
<!ATTLIST case
  select CDATA #REQUIRED
>
```

Example

```
<switch select="sys">
  <case select="WIN">
    <paragraph>This appears in Windows.</paragraph>
  </case>
  <case select="UNIX">
    <paragraph>This appears in Unix.</paragraph>
  </case>
  <default>
    <paragraph>This appears in all other cases</paragraph>
  </default>
</switch>
```

Caseinline

This element holds the cases for an `switchinline` statement.

Attributes

Attribute	Required	Contents	Values allowed	Description
select	yes	CDATA		Contains the value that is to be evaluated. See <i>Switching Content</i> on page 22 for more information.

Parent Elements

`switchinline`

Child Elements

`image`, `embedvar`, `br`, `emph`, `item`, `link`, `switchinline`, `variable`, `ahelp`, `object`

Element Definition

```
<!ELEMENT caseinline (#PCDATA | image | embedvar | br | emph |
                    item | link | switchinline | variable |
                    ahelp | object)*>
<!ATTLIST caseinline
  select CDATA #REQUIRED
>
```

Example

```
<paragraph>Press the
  <switchinline select="sys">
    <caseinline select="WIN">Ctrl</caseinline>
    <caseinline select="MAC">Apple</caseinline>
    <defaultinline>any</defaultinline>
  </switchinline>
  key to start.
</paragraph>
```

Comment

This element is used for inserting comments into the help files used by the author/editor/translator. They are to be removed when the help files are compiled.

Attributes

none

Parent Elements

body, case, default, list, listitem, section, switch, tablecell

Child Elements

none

Element Definition

```
<!ELEMENT comment (#PCDATA)>
```

Example

```
<section>  
  <comment>FPE: This section is in a draft state!</comment>  
</section>
```

Created

This element holds the date of document creation and additional information (author or comment).

Attributes

Attribute	Required	Contents	Values allowed	Description
date	yes	CDATA		Contains the date of document creation in the format YYYY-MM-DDThh:mm:ss where: YYYY = four-digit year MM = two-digit month DD = two-digit day of month hh = two digits of 24 hour mm = two digits of minute ss = two digits of second

Parent Elements

history

Child Elements

none

Element Definition

```
<!ELEMENT created (#PCDATA)>
<!ATTLIST created
  date CDATA #REQUIRED
>
```

Example

```
<meta>
  <history>
    <created date="2002-05-20T15:15:00">FPE: New topic created</created>
    <lastedited date="2002-06-20T15:15:00">UFI: Made minor
changes</lastedited>
  </history>
</meta>
```

Default

This element holds the default values for a `switch`. It is evaluated if all `case` elements of a `switch` element evaluate to false.

Attributes

none

Parent Elements

`switch`

Child Elements

`paragraph`, `table`, `comment`, `bookmark`, `embed`, `list`, `section`

Element Definition

```
<!ELEMENT default (paragraph | table | comment | bookmark |  
                  embed | link | list | switch | section)*>
```

Example

```
<switch select="sys">  
  <case select="WIN">  
    <paragraph>This appears in Windows.</paragraph>  
  </case>  
  <case select="UNIX">  
    <paragraph>This appears in Unix.</paragraph>  
  </case>  
  <default>  
    <paragraph>This appears in all other cases</paragraph>  
  </default>  
</switch>
```

Defaultinline

This element holds the default values for an inline switch. It is evaluated if all `caseinline` elements of a `switchinline` element evaluate to false.

Attributes

none

Parent Elements

`switchinline`

Child Elements

`image`, `embedvar`, `br`, `emph`, `item`, `link`, `switchinline`, `variable`, `ahelp`, `object`

Element Definition

```
<!ELEMENT defaultinline (#PCDATA | image | embedvar | br | emph |
                           item | link | switchinline | variable |
                           ahelp | object)*>
```

Example

```
<paragraph>Press the
  <switchinline select="sys">
    <caseinline select="WIN">Ctrl</caseinline>
    <caseinline select="MAC">Apple</caseinline>
    <defaultinline>any</defaultinline>
  </switchinline>
  key to start.
</paragraph>
```

Embed

This element is used to embed content from a different source at the current position. The only elements that can be embedded from somewhere else are `sections` or `paragraphs` which are identified by their URL. For smaller text fragments, `embedvar` may be used. See also *Embedding Content* on page 24.

The optional `role` attribute may override the role of a paragraph. For embedded sections, the `role` attribute has no effect.

Attributes

Attribute	Required	Contents	Values allowed	Description
<code>href</code>	yes	URL		A URL pointing to the content to be embedded. The URL has the form <code>filepath#id</code> . Filepath is the path of the file as contained in the <code>jar</code> archive.
<code>role</code>	no		see <i>Paragraph Roles</i> on page 20.	The role in which the embedded paragraph will appear. If this attribute is specified the paragraph is displayed with this role overwriting its original role (not applicable for sections).
<code>level</code>	no	fixed values	numerical value	The heading level if the <code>role</code> attribute is set to "heading"

Parent Elements

`body`, `case`, `default`, `listitem`, `section`, `tablecell`

Child Elements

none

Element Definition

```
<!ELEMENT embed EMPTY>
<!ATTLIST embed
  href CDATA #REQUIRED
  role CDATA #IMPLIED
  level CDATA #IMPLIED
>
```

Example

```
<embed href="text/swriter/guide/editing#4711"/>
<embed href="text/scalc/01/0123456#9876" role="warning"/>
```

Embedvar

This element is used to embed smaller text fragments with and without markup which were previously declared as being `variables`. See also *Embedding Content* on page 24.

Attributes

Attribute	Required	Contents	Values allowed	Description
href	yes	URL		A URL pointing to the content to be embedded. The URL has the form <code>filepath#id</code> . Filepath is the path of the file as contained in the <code>jar</code> archive.
markup	no	fixed values	"keep" "ignore"	Specifies whether markup contained in the variable to be embedded is ignored or kept in the target position. The default is to keep markup within the text fragment.

Parent Elements

`ahelp`, `caption`, `caseinline`, `defaultinline`, `link`, `paragraph`, `variable`

Child Elements

none

Element Definition

```
<!ELEMENT embedvar EMPTY>
<!ATTLIST embedvar
  href CDATA #REQUIRED
  markup (keep | ignore) #IMPLIED
>
```

Example

```
<paragraph>This element can be found on the <embedvar
href="text/swriter/01/dialogs#fileopen" markup="ignore"/> dialog.</paragraph>
```

Emph

This element is used to mark emphasized content. It may only contain PCDATA.

Attributes

none

Parent Elements

ahelp, caption, caseinline, defaultinline, link, paragraph, variable

Child Elements

item, comment

Element Definition

```
<!ELEMENT emph (#PCDATA | item | comment)*>
```

Example

```
<paragraph><emph>Never</emph> delete the paragraph</paragraph>
```

Filename

This element contains the path and name of the help topic file as included in the jar file, e.g. `text/swriter/01/1234567.xhp`.

Attributes

none

Parent Elements

`topic`

Child Elements

none

Element Definition

```
<!ELEMENT filename (#PCDATA)>
```

Example

```
<filename>text/swriter/01/08154711.xhp</filename>
```

Helpdocument

This is the root element of a help document and contains the `meta` and `body` part of the help topic.

Attributes

Attribute	Required	Contents	Values allowed	Description
version	yes	CDATA		Contains the Help XML format version number currently 1.0) for compatibility to future versions.

Parent Elements

none

Child Elements

`meta`, `body`

Element Definition

```
<!ELEMENT helpdocument (meta, body)>
<!ATTLIST helpdocument
  version CDATA #REQUIRED
>
```

Example

```
<helpdocument version="1.0">
<meta></meta>
<body></body>
</helpdocument>
```

History

This element contains information about the author and the date of creation as well as the same information about the last editing cycle.

Attributes

none

Parent Elements

meta

Child Elements

created, lastedited

Element Definition

```
<!ELEMENT history (created, lastedited)>
```

Example

```
<meta>
  <history>
    <created date="2002-05-20T15:15:00">FPE: New topic created</created>
    <lastedited date="2002-06-20T15:15:00">UFI: Made minor changes</created>
  </history>
</meta>
```

Image

This element carries information for images in the document.

Attributes

Attribute	Required	Contents	Values allowed	Description
src	yes	URL		A URL pointing to the image as included in the picture archive <code>picture.jar</code> .
width	no	CDATA		The image width. If missing it is set to 100%.
height	no	CDATA		The image height. If missing it is set to 100%.
id	yes	CDATA		A unique id to identify the image, see <i>Common Attributes</i> on page 29.

Parent Elements

caseinline, defaultinline, paragraph, variable, tablecell

Child Elements

caption, alt

Element Definition

```
<!ELEMENT image (caption* | alt+)?>
<!ATTLIST image
  src CDATA #REQUIRED
  width CDATA #IMPLIED
  height CDATA #IMPLIED
  id CDATA #REQUIRED
>
```

Example

```
<image src="picture/win/common/writermainwin.xhp" id="img4711" width="75"
height="75">
  <caption xml-lang="en-US" id="cp4711">
    The <emph>main writer windows</emph> showing all
    writer toolbars.
  </caption>
  <alt xml-lang="en-US" id="alt4711">Main program window</alt>
</image>
```

Item

This generic element is used to markup special objects which are to be formatted in a special way. The attribute `type` is used to specify the item type (a keystroke, a menu item, a dialog title etc). This element resembles the `` element in HTML.

Attributes

Attribute	Required	Contents	Values allowed	Description
<code>type</code>	yes			The item type that will be used to format the data, for example "menuitem".

Parent Elements

`ahelp`, `caption`, `caseinline`, `defaultinline`, `link`, `paragraph`, `variable`, `emph`

Child Elements

none

Element Definition

```
<!ELEMENT item (#PCDATA)>
<!ATTLIST item
  type CDATA #REQUIRED
>
```

Example

```
<paragraph>
You see the <item type="dialog">File Open</item> dialog if you press the <item
type="button">Open</item> button or select <item type="menuitem">File-
Open</item> from the menu.
</paragraph>
```

Lastedited

This element contains the date when the document was last edited inside the `date` attribute. Additional information can be specified as PCDATA.

Attributes

Attribute	Required	Contents	Values allowed	Description
date	yes	CDATA		<p>Contains the date when the document was last edited, in the format</p> <p>YYYY-MM-DDThh:mm:ss</p> <p>where:</p> <p>YYYY = four-digit year MM = two-digit month DD = two-digit day of month hh = two digits of 24 hour mm = two digits of minute ss = two digits of second</p>

Parent Elements

history

Child Elements

none

Element Definition

```
<!ELEMENT lastedited (#PCDATA)>
<!ATTLIST lastedited
  date CDATA #REQUIRED
>
```

Example

```
<meta>
  <history>
    <created date="2002-05-20T15:15:00">FPE: New topic created</created>
    <lastedited date="2002-06-20T15:15:00">UFI: Made minor
changes</lastedited>
  </history>
</meta>
```

Link

This element contains a link to another document inside or outside the help system. For links to other help files, the URL syntax is

```
filename#anchor_target
```

with

filename Path and name of the help file as contained in the help jar archive, e. g. `text/swriter/01/123345.xhp`.

anchor_target Anchor target to jump to (optional). These can be the ids of bookmarks, sections, or paragraphs.

Attributes

Attribute	Required	Contents	Values allowed	Description
href	yes	URL		This contains the link address as URL.
name	yes			This is the link name, needed to fulfill accessibility requirements.
type	no			This specifies the link type, for example, a popup. Currently not evaluated.
target	no			Can be used to specify a target frame.

Parent Elements

ahelp, caption, caseinline, defaultinline, paragraph, variable

Child Elements

emph, item, variable, embedvar, switchinline

Element Definition

```
<!ELEMENT link (#PCDATA | embedvar | emph | item | variable |
                switchinline)*>
<!ATTLIST link
  href CDATA #REQUIRED
  name CDATA #REQUIRED
  type CDATA #IMPLIED
  target CDATA #IMPLIED
>
```

Example

Please refer to `<link href="http://www.openoffice.org" name="Link to the OpenOffice.org Website">the <emph>OpenOffice.org</emph> website</link>` for further details.

More details can be found in `<link href="text/common/guide/overview.xhp" name="Link to the overview">the overview</link>`.

List

This element represents ordered (numbered) and unordered (bulleted) lists. The element itself does not contain any PCDATA but only child elements which carry the content or comments.

Attributes

Attribute	Required	Contents	Values allowed	Description
type	yes	fixed values	"ordered" or "unordered"	Describes the list type as either being ordered (numbered) or unordered (bulleted).
startwith	no	CDATA		The starting number of an ordered list; if omitted, the list starts with 1.
format	no	fixed values	"1", "i", "I", "a", "A"	The number format used in numbered (ordered) lists: "1": arabic numerals "i": small roman numerals "I": capital roman numerals "a": small letters "A": capital letters If omitted the list uses "1".
bullet	no	fixed values	"disc", "circle", "square"	The bullet to be used in bulleted (unordered) lists.
localize	no	fixed value	"false"	See <i>Common Attributes</i> on page 29
sorted	no	fixed values	"asc", "desc"	Specifies whether the list should be sorted, either ascending or descending. If this attribute is not given, the list is not sorted. If it is given, the listitem child elements are sorted according to the current locale.

Parent Elements

body, case, default, section, tablecell

Child Elements

listitem, comment

Element Definition

```
<!ELEMENT list (listitem | (comment)*)+>
<!ATTLIST list
  type CDATA #REQUIRED
  startwith CDATA #IMPLIED
  format (1 | i | I | a | A) #IMPLIED
  bullet (disc | circle | square) #IMPLIED
  localize CDATA #IMPLIED
  sorted (asc | desc) #IMPLIED
>
```

Example

```
<list type="ordered" startwith="5" format="a">
<comment>This list lists the first five letters of the alphabet</comment>
...
</list>
```

Listitem

This element holds the content of a list in child elements.

Attributes

Attribute	Required	Contents	Values allowed	Description
format	no	fixed values	"1", "i", "I", "a", "A"	The number format used in numbered (ordered) list items: "1": arabic numerals "i": small roman numerals "I": capital roman numerals "a": small letters "A": capital letters If omitted the list item uses the value set in the list.
bullet	no	fixed values	"disc", "circle", "square"	The bullet to be used in bulleted (unordered) list items.
localize	no	fixed value	"false"	See <i>Common Attributes</i> on page 29
class	no			

Parent Elements

list

Child Elements

comment, section, paragraph, table, switch, embed, bookmark

Element Definition

```
<!ELEMENT listitem (comment | section | paragraph | table |
                    switch | embed | bookmark)*>
<!ATTLIST listitem
  format (1 | i | I | a | A) #IMPLIED
  bullet (disc | circle | square) #IMPLIED
  localize CDATA #IMPLIED
  class CDATA #IMPLIED
>
```

Example

```
<listitem bullet="disc">
  <paragraph>Insert the CD.</paragraph>
</listitem>
```

Meta

This element contains sub-elements with data used to organize the help.

Attributes

none

Parent Elements

helpdocument

Child Elements

topic, history

Element Definition

```
<!ELEMENT meta (topic, history?)>
```

Example

```
<helpdocument>  
  <meta>  
    <history>...</history>  
  </meta>  
  <body>  
  </body>  
</helpdocument>
```

Object

This generic element contains information about objects to be embedded into the help page like audio or video files. It is reserved for future use.

Attributes

Attribute	Required	Contents	Values allowed	Description
type	yes	CDATA		Specifies the mime type of the embedded object data.
id	yes	CDATA		A unique id to identify the image, see <i>Common Attributes</i> on page 29.
data	yes	CDATA		Specifies the object file.
height	no	CDATA		Specifies the width of the object.
width	no	CDATA		Specifies the height of the object.

Parent Elements

paragraph, caseinline, defaultinline, variable

Child Elements

none

Element Definition

```
<!ELEMENT object EMPTY>
<!ATTLIST object
  type CDATA #REQUIRED
  id CDATA #REQUIRED
  data CDATA #REQUIRED
  height CDATA #IMPLIED
  width CDATA #IMPLIED
>
```

Example

```
<object data="clock.svg" id="objClock" type="image/svg+xml" width="200"
height="200">
```

Paragraph

This element is the standard element holding content. The `role` attribute defines its context in greater detail. See also *Paragraph Roles* on page 20.

Attributes

Attribute	Required	Contents	Values allowed	Description
<code>role</code>	yes	CDATA		Describes the current role of the paragraph, for example a simple paragraph or a heading or an example or a note. See also <i>Paragraph Roles</i> on page 20.
<code>level</code>	no			Defines the heading level if the paragraph role is set to "heading".
<code>id</code>	yes			See <i>Common Attributes</i> on page 29.
<code>l10n</code>	yes			Contains the localization status of the old help files and is only used for migration purposes.
<code>xml-lang</code>	yes	CDATA		See <i>Common Attributes</i> on page 29.
<code>oldref</code>	no	CDATA		This contains the reference number used by the old help files and is only used for migration purposes.
<code>localize</code>	no	fixed value	"false"	See <i>Common Attributes</i> on page 29

Parent Elements

`body`, `case`, `default`, `listitem`, `section`, `tablecell`

Child Elements

`image`, `embedvar`, `br`, `emph`, `item`, `link`, `switchinline`, `variable`, `ahelp`, `object`, `bookmark`

Element Definition

```
<!ELEMENT paragraph (#PCDATA | image | comment | embedvar | br |
                    emph | item | link | switchinline | variable |
                    ahelp | object | bookmark)*>
<!ATTLIST paragraph
  role CDATA #REQUIRED
  level CDATA #IMPLIED
  id CDATA #REQUIRED
  l10n CDATA #REQUIRED
  xml-lang CDATA #REQUIRED
  oldref CDATA #IMPLIED
  localize CDATA #IMPLIED
>
```

Example

```
<paragraph
  role="heading" level="1" id="par4711_001" xml_lang="en-US">
  Italic characters
</paragraph>

<paragraph
  role="paragraph" id="par4711_002" xml_lang="en-US">
  Proceed as follows to write an italic character
</paragraph>
```

Section

This element serves as a generic container for multiple elements to make them embeddable in other documents. Each `section` takes a unique `id` which is used to identify it when embedded in other documents. Subsections are allowed. A `section` may only contain sub-elements.

Attributes

Attribute	Required	Contents	Values allowed	Description
<code>id</code>	yes			See <i>Common Attributes</i> on page 29.
<code>localize</code>	no	fixed value	"false"	See <i>Common Attributes</i> on page 29

Parent Elements

`body`, `listitem`, `section`, `tablecell`, `case`, `default`

Child Elements

`section`, `paragraph`, `table`, `list`, `comment`, `bookmark`, `embed`, `switch`, `sort`

Element Definition

```
<!ELEMENT section (section | paragraph | table | list | comment |
                  bookmark | embed | switch | sort )*>
<!ATTLIST section
  id CDATA #REQUIRED
  localize CDATA #IMPLIED
>
```

Example

```
<section id="4711"><paragraph>This applies to multiple
applications</paragraph></section>
```

Sort

This element is used to mark up a set of sections that are to be sorted. The sequence of the sections inside the sort element plays no role for the display sequence.

Attributes

Attribute	Required	Contents	Values allowed	Description
order	no	fixed values	"asc", "desc"	Defines the sorting order as being ascending or descending .

Parent Elements

body, section

Child Elements

section

Element Definition

```
<!ELEMENT sort (section+)>
<!ATTLIST sort
  order (asc | desc) #IMPLIED
>
```

Example

```
<sort order="asc">
  <section id="123243">...</section>
  <section id="24345">...</section>
</sort>
```

Switch

This element is used to switch sections or paragraphs for different platform, application, distribution, target medium or language context. For switching content inside paragraphs, `switchinline` must be used.

Attributes

Attribute	Required	Contents	Values allowed	Description
<code>select</code>	yes	fixed values	"sys" "appl" "distrib" "target" "lang" "ver"	Defines the context that is to be evaluated, see also <i>Switching Content</i> on page 22.
<code>localize</code>	no	fixed value	"false"	See <i>Common Attributes</i> on page 29

Parent Elements

`body`, `listitem`, `section`

Child Elements

`case`, `comment`, `default`

Element Definition

```
<!ELEMENT switch ((case | comment)* | default?)*>
<!ATTLIST switch
  select (sys | appl | distrib | target | lang | ver) #REQUIRED
  localize CDATA #IMPLIED
>
```

Example

```
<switch select="sys">
  <case select="WIN">
    <paragraph>This appears in Windows.</paragraph>
  </case>
  <case select="UNIX">
    <paragraph>This appears in Unix.</paragraph>
  </case>
  <default>
    <paragraph>This appears in all other cases</paragraph>
  </default>
</switch>
```

Switchinline

This element is used to switch parts of paragraphs for different platform, application, distribution, target medium or language context. For switching complete paragraphs or sections `switch` must be used.

Attributes

Attribute	Required	Contents	Values allowed	Description
select	yes	fixed values	"sys", "appl", "distrib", "target", "lang", "ver"	Defines the context that is to be evaluated, see also <i>Switching Content</i> on page 22.

Parent Elements

caption, caseinline, defaultinline, paragraph, variable, link

Child Elements

[caseinline](#), [defaultinline](#)

Element Definition

```
<!ELEMENT switchinline ((caseinline)+, (defaultinline?))>
<!ATTLIST switchinline
  select (sys | appl | distrib | target | ver | lang) #REQUIRED
>
```

Example

```
<paragraph>Press the
  <switchinline select="sys">
    <caseinline select="WIN">Ctrl</caseinline>
    <caseinline select="MAC">Apple</caseinline>
    <defaultinline>any</defaultinline>
  </switchinline>
  key to start.
</paragraph>
```

Table

This element defines a table containing one or more `tablerows`. The `table` element itself only contains child elements.

Attributes

Attribute	Required	Contents	Values allowed	Description
<code>name</code>	no	CDATA		Contains a table name.
<code>width</code>	no	CDATA		Contains the width of the table in units as given in the <code>units</code> attribute. If omitted, the table width is set by the help viewer.
<code>height</code>	no	CDATA		Contains the height of the table in units as given in the <code>units</code> attribute. If omitted, the table height is set by the help viewer.
<code>unit</code>	no	fixed values	"px", "pt", "cm", "in", "pct"	Contains the unit to be used for table <code>width</code> and <code>height</code> : px = pixels pt = points cm = centimeters in = inches pct = percent (%) If omitted, pixels (px) are used as units.
<code>class</code>	no	CDATA		Contains a class descriptor for the table which may be used to assign special formats.
<code>id</code>	yes	CDATA		See <i>Common Attributes</i> on page 29.
<code>localize</code>	no	fixed value	"false"	See <i>Common Attributes</i> on page 29

Parent Elements

`body`, `case`, `default`, `listitem`, `section`

Child Elements

`caption`, `tablerow`

Element Definition

```
<!ELEMENT table (caption*, tablerow+)>
<!ATTLIST table
  name CDATA #IMPLIED
  width CDATA #IMPLIED
  height CDATA #IMPLIED
  unit CDATA #IMPLIED
  class CDATA #IMPLIED
  id CDATA #REQUIRED
  localize CDATA #IMPLIED
>
```

Example

```
<table
  id="tab4711" name="List of Shortcuts"
  width="90" unit="pct" class="shortcutlist">
  <caption>
    <paragraph>List of shortcuts</paragraph>
  </caption>
  <tablerow>...</tablerow>
</table>
```

Tablecell

This element contains child elements taking the cell content. Complex tables may be created using the `rowspan` and `colspan` attributes of `tablecell`.

Attributes

Attribute	Required	Contents	Values allowed	Description
<code>colspan</code>	no	CDATA		Contains the number of columns spanned by this cell. If omitted, the cell spans 1 column.
<code>rowspan</code>	no	CDATA		Contains the number of rows spanned by this cell. If omitted, the cell spans 1 row.
<code>width</code>	no	CDATA		Contains the width of the table cell in units as given in the <code>units</code> attribute. If omitted, the table cell width is set by the help viewer.
<code>unit</code>	no	fixed values	"px", "pt", "cm", "in", "pct"	Contains the unit to be used for width: px = pixels pt = points cm = centimeters in = inches pct = percent (%) If omitted, pixels (px) are used as units.
<code>class</code>	no	CDATA		Contains a class descriptor for the table cell which may be used to assign special formats.
<code>localize</code>	no	fixed value	"false"	See <i>Common Attributes</i> on page 29

Parent Elements

`tablerow`

Child Elements

`section`, `paragraph`, `comment`, `embed`, `bookmark`, `image`, `list`

Element Definition

```
<!ELEMENT tablecell (section | paragraph | comment | embed |  
                    bookmark | image | list)*>  
<!ATTLIST tablecell  
  colspan CDATA #IMPLIED  
  rowspan CDATA #IMPLIED  
  width CDATA #IMPLIED  
  class CDATA #IMPLIED  
  unit CDATA #IMPLIED  
  localize CDATA #IMPLIED  
>
```

Example

```
<table  
  id="tab_4711" name="List of Shortcuts"  
  width="90" unit="pct" class="shortcutlist">  
  <tablerow>  
    <tablecell>Column 1</tablecell>  
    <tablecell>Column 2</tablecell>  
    <tablecell>Column 3</tablecell>  
  </tablerow>  
  <tablerow>  
    <tablecell colspan="2">  
      This cell spans 2 columns, namely column 1 and 2  
    </tablecell>  
    <tablecell>  
      This cell spans 1 column, namely column 3  
    </tablecell>  
  </tablerow>  
</table>
```

Tablerow

This element contains table rows which themselves only contain table cells.

Attributes

Attribute	Required	Contents	Values allowed	Description
height	no	CDATA		Contains the height of the table row in units as given in the <code>units</code> attribute. If omitted, the table row height is set by the help viewer.
unit	no	fixed values	"px", "pt", "cm", "in", "pct"	Contains the unit to be used for <code>height</code> : px = pixels pt = points cm = centimeters in = inches pct = percent (%) If omitted, pixels (px) are used as units.
class	no	CDATA		Contains a class descriptor for the table row which may be used to assign special formats.
localize	no	fixed value	"false"	See <i>Common Attributes</i> on page 29

Parent Elements

`table`

Child Elements

`tablecell`

Element Definition

```
<!ELEMENT tablerow (tablecell+)>
<!ATTLIST tablerow
  height CDATA #IMPLIED
  class CDATA #IMPLIED
  unit CDATA #IMPLIED
  localize CDATA #IMPLIED
>
```

Title

This is the title of the help page as displayed in the list on the **Contents** tab page, the **Index** list and the search results. The title content may **not** contain HTML entities like `'` or `&`.

Attributes

Attribute	Required	Contents	Values allowed	Description
xml-lang	yes	CDATA		See <i>Common Attributes</i> on page 29.
id	yes	CDATA		See <i>Common Attributes</i> on page 29.
localize	no	fixed value	"false"	See <i>Common Attributes</i> on page 29.

Parent Elements

topic

Child Elements

none

Element Definition

```
<!ELEMENT title (#PCDATA)>
<!ATTLIST title
  xml-lang CDATA #REQUIRED
  id CDATA #REQUIRED
  localize CDATA #IMPLIED
>
```

Example

```
<topic>
  <title xml_lang="en-US" id="tit1233">How to open a text document</title>
</topic>
```

Topic

This element contains child elements with information about the current help topic.

Attributes

Attribute	Required	Contents	Values allowed	Description
id	yes	CDATA		See <i>Common Attributes</i> on page 29.
indexer	no	fixed value	"exclude", "include"	Specifies whether the current file is to be excluded from the indexing process. An excluded file cannot be found using the help search facility.

Parent Elements

meta

Child Elements

title, bookmark, filename

Element Definition

```
<!ELEMENT topic (title+, filename, bookmark*)>
<!ATTLIST topic
  id CDATA #REQUIRED
  indexer (exclude | include) #IMPLIED
>
```

Example

```
<topic id="4711" indexer="exclude">
  <title xml_lang="en-US">Invisible help file</title>
  <filename>text/swriter/01/08154711</filename>
</topic>
```

Variable

This element is used to define reusable text fragments. The fragments can be embedded in other contexts by means of the `embedvar` element.

Attributes

Attribute	Required	Contents	Values allowed	Description
id	yes	CDATA		See <i>Common Attributes</i> on page 29.
visibility	no	fixed values	"hidden", "visible"	Specifies whether the element content will be shown in the help viewer. If this attribute is omitted, the variable will be visible .

Parent Elements

ahelp, caption, caseinline, defaultinline, link, paragraph, variable

Child Elements

ahelp, embedvar, br, emph, item, link, variable, image, object, switchinline

Element Definition

```
<!ELEMENT variable (#PCDATA | ahelp | embedvar | br | emph |
                    item | link | variable | image | object |
                    switchinline)*>
<!ATTLIST variable
  id CDATA #REQUIRED
  visibility (hidden | visible) #IMPLIED
>
```

Example

```
You may use the <variable id="dlg_FileOpen"><item type="dialog">Open a
file</item></variable> to open a file.

<comment>List of menu names to be embedded</comment>
<paragraph xml_lang="en-US">
<variable id="menu_File" visibility="hidden">File Menu</variable>
<variable id="menu_Edit" visibility="hidden">Edit Menu</variable>
<variable id="menu_View" visibility="hidden">View Menu</variable>
</paragraph>
```


Authoring Help With OpenOffice.org

You need OpenOffice.org 1.1.x to use the help authoring environment. The authoring environment also is not compatible with OpenOffice.org 2.0.

Setting Up The Environment

There is an import/export filter available that allows for direct editing of OpenOffice.org help files without the need of extra conversion steps. This is how you set it up.

The help files use the extension **xhp**.

Directory Hierarchy

The correct function of the help authoring environment with OpenOffice.org relies on a special directory layout. Since all help file and image references are expressed in a relative way the environment needs to know the absolute paths to be able to assemble and disassemble the references correctly for display in OpenOffice.org.

The following directory structure is recommended (the marked entries have to be entered into the import/export filters, see below). The directory structure below `helpcontent2` is mandatory and determined by the structure of the CVS module:

```
$root
|__ $imagedir
|__ helpcontent2
|   |__ helpers
|   |__ util
|   |__ prj
|   |__ source
|       |__ text
|       |__ auxiliary
```

- Select and create a root directory for working on the help files (`$root`), for example: `/opt/ooohelp` or `D:\ooohelp`

Creating The Image Directory

\$`imagedir` contains all images of the central image repository `images.zip`.

1. Create an image directory as child of the help root directory (as described above), for example: `/opt/oooohelp/helpimages`.
2. Get the file `images.zip` from an installation or the output tree (solver) [6] and unpack it into the image directory.

Creating The Help Files Directory

- Check out the module `helpcontent2` from OpenOffice.org CVS into the help root directory (as described above) [6]. This will create the correct directory hierarchy for the help files.

Installing The Import/Export Filters

Ensure that the XML Filter option is installed in OpenOffice.org. If this option is not installed install it using the option from the the OpenOffice.org setup.

1. Get the filter package from the CVS module
`http://documentation.openoffice.org/source/browse/documentation/helpcontent2/helpers/helpauthoring/filter/helpauthoring.jar`

There is one common package for Windows and Unix.

The packages contain the **XSLT Import/Export filters** and the **template**. The Import filter has the root path to the help files hard coded in it. You can open the import filter `xmlhelp2soffice.xsl` from your OpenOffice.org `/user/xslt` directory and adjust the root path manually.

2. Open a text document in OpenOffice.org
The menu item for the XML Settings will only be visible if a document is loaded.
3. Choose **Tools - XML Filter Settings**
Remove any filters that are listed there. You need to remove the **Docbook** filter at least since this is using the xml extension
4. Click **Open Package**
5. Browse to the package you downloaded in the first step and click **Open**.
6. Click **Close**

You can now open and save files in OpenOffice.org help format.

[6] http://www.openoffice.org/dev_docs/source/get_source.html

Customizing The Import/Export Filters

In order to allow for correct image display and linking you need to adjust the import filter manually to match your directory hierarchy as described in the previous section.

Incorrect or missing path names in the filters can lead to corrupt file or image links in the help files.

1. Change to the `user/xslt/help` directory of your OpenOffice.org installation, for example, `~/openoffice.org/user/xslt/help`. The two `xsl` files you see there are the import and export filters for the help files.

2. Open `soffice2xmlhelp.xsl` using an XML or a standard text editor.

3. Change line 34 to reflect the root path of your help images (`$imagedir` on page 71) in URL notation, for example,

```
<xsl:param name="imgroot" select="'file:///helpimages/'"/>
```

Be sure to add a trailing slash.

4. Save `soffice2xmlhelp.xsl`.

5. Open `xmlhelp2soffice.xsl` using an XML or a standard text editor.

6. Change lines 42 and 43 to reflect the root path of your help files and the root path of your images (`$root/source` and `$imagedir` on page 71) in URL notation:

```
<xsl:param name="root" select="'file:///helpcontent2/source/'"/>
```

```
<xsl:param name="imgroot" select="'file:///helpimages/'"/>
```

Installing The Supporting Macros

The macro set is used to perform tasks inside the help files.

The macros are only tested with OpenOffice.org versions 1.1.x

1. Download the macro archive from <http://documentation.openoffice.org/source/browse/documentation/helpcontent2/helpers/helpauthoring/macros/HelpAuthoring.tar.gz>
2. Unpack it to a temporary directory.
3. Choose **Tools - Macros - Macro**
4. Click **Organizer**
5. Click the **Libraries** tab
6. Click **Append** and browse to the temporary directory with the unpacked macros

7. Select the file `script.xlb` inside the macro directory and click **Open**
8. Select **Replace existing libraries** and click **Ok**
9. Close the macro dialogs

You can now use the macros for help authoring.

For some tasks the macro set needs to know the paths to your help files and images as described on page 71. You will be asked to provide these paths once a macro needs it. They will then be stored inside your `user/config` directory in the file `helpauthoring.cfg`. You need to delete this file when the paths to your help files or images change.

Installing The Help Authoring Menu

The Help Authoring menu allows easy access to the macros to perform standard tasks with the help files.

This procedure will overwrite any menu customization made by you.

1. Download the zip archive that contains the menu set from
http://documentation.openoffice.org/source/browse/documentation/helpcontent2/helpers/helpauthoring/helpauthoring_menu.zip
2. Choose **Tools - Configure** and select the **Menu** tab
3. Click **Load**
4. Locate the `helpauthoring_menu.zip` file
5. Select the file and then click **Ok**



You can now use the menu. If all steps were performed successfully you should be ready to go.

Editing Help Files - Basics

Since OpenOffice.org cannot simply be used as an XML editor we need to make some effort to map elements and attributes in the Help file to elements that are known to OpenOffice.org.

For now, not all attributes for the elements are supported for editing in OpenOffice.org. The most important ones, however, are available.

Paragraphs And Paragraph Formatting

Paragraphs are the central content carrying element in a help file. A `paragraph` in the help file maps to a paragraph in OpenOffice.org. The `role` attribute of a paragraph maps to a paragraph **style** in OpenOffice.org.

For every paragraph role in the help file there is a paragraph style beginning with `h1p_` and ending with the role name, e.g. the role `paragraph` maps to a style named `h1p_paragraph`.

There are also special paragraph styles that start with `h1p_aux_`. These are used for identifying special elements and should *never* be used for paragraphs.

Any paragraph that does *not* have a paragraph style starting with `h1p_` will be disregarded on export by the export filter. Its content will be *lost* on the next reload of that file.

The default paragraph styles for a help file (that is, the default roles of a paragraph) are already pre-defined in the help authoring template (`xmlhelptemplate.stw`) that is used for loading the help files in OpenOffice.org. This template is part of the import/export package and automatically installed in your `user/template` directory.

You can define new paragraph styles that are transformed to roles in the help files on export by creating a custom style beginning with `h1p_` and ending in the role name. This new style will be recreated on loading that file in OpenOffice.org next time. However, there will be no formatting information for OpenOffice.org associated with it. For that, the style needs to be added to the template.

Note, that these styles don't have any effect on the help as it is displayed *per se*. In order to adjust the help appearance the roles that are created from the paragraph styles must be transformed by the main transformation style sheet and/or assigned to formats using the style sheets of the help.

Sections

A `section` in the help file maps to a section in OpenOffice.org. You can use the navigator to get an overview of existing sections or use the **Insert - Section** and **Format - Sections** menus to modify existing sections. The `section ID` maps to the section name in OpenOffice.org. Nested sections are supported.

Tables

A table in a OpenOffice.org help file transforms to a visible table in the OpenOffice.org file. The table name holds all attributes for that table. If the table has a caption defined in the help file, a paragraph is created directly after the table that contains the caption. It is important that this sequence is not modified since the export filter relies on that.

Tables should no longer be used for formatting purpose, e.g. to place images or to mimic numbered lists. Nevertheless, there still is a considerable amount of legacy help files that does that.

Images

Images are mapped to image objects in OpenOffice.org that are *linked* (not embedded) to the OpenOffice.org file and anchored *as character*. The alternative text is defined in the **Alternative** property of the image object that can be accessed through the **Graphics** properties dialog (by double-clicking the image) on the **Options** tab page. The `ID` of an image is stored in the name of the image object and should not be altered manually.

The images will only be displayed in OpenOffice.org correctly if the path to the image files was correctly specified in the import and export filters (see *Customizing the Import/Export Filters* on page 73).

Lists

There are two types of lists in the help (unordered and ordered) that match to the corresponding list type in OpenOffice.org.

Embedding

The embedding technique is unique to the help. Therefore, we use some workarounds to implement embedding when editing the Help files in OpenOffice.org:

Sections to be embedded are represented as sections.

Paragraph parts to be embedded are surrounded by a `variable` tag pair.

Places where sections are embedded are designated by an `embed` tag.

Places where parts of paragraphs are embedded are designated by an `embedvar` tag.

Character Formatting

Direct (hard) formatting is *not* supported. Any character with a direct format will lose its format definition on export. Instead, character formatting is achieved using **character styles**. The importing template already contains a list of pre-defined character styles. All styles that begin with `h1p_` may be used for character formatting except for the styles beginning with `h1p_aux_` because those are used for internal purpose.

Similar to the paragraph styles you can define new character styles that are transformed to `type` attributes of `item` elements in the help files on export. To do this you create a custom character style beginning with `h1p_` and ending in the type name (e.g., `h1p_dialogname`). This new style will be recreated on loading that file in OpenOffice.org next time. However, there will be no formatting information for OpenOffice.org associated with it. For that, the style needs to be added to the help authoring template.

Note, that these styles don't have any effect on the help as it is displayed *per se*. In order to adjust the help appearance the item types that are created from the character styles must be transformed by the main transformation style sheet and/or assigned to formats using the style sheets of the help.

Working With The Help Files

Ensure that you have set up your work environment correctly as described in *Setting Up the Environment* on page 71.

Creating A Help File

1. Start OpenOffice.org and open a new empty Writer window

The help authoring menu is only available in the **Writer context**. So you need to have a Writer window open.

2. Choose **Help Authoring - Create New Help File**.

You should always use this way for creating a new file to ensure that all settings are made correctly.

3. Select a file name inside the help directory structure to save to

The directory structure is described on page 71. You will be automatically prompted to save the file. You need to save it before you can actually work on it.

4. Insert an initial comment (optional)

You will be prompted to insert a comment on file creation. This comment will be stored in the file metadata and cannot be changed using OpenOffice.org.

Now you have created a fresh empty help document. The file title is set to the generic term `<Set Topic Title>`. To adjust the topic title see *Meta Data* on page 91.

Switch the Stylist to show **Custom** styles to view a list of all styles that are allowed in the help file. None but these (and the ones created by you following the guidelines above) may be used.

After the file is finished it needs to be added to the list of files to be processed by the help compiler. This is done by adding the file to the makefile of its directory. You can either do that manually or run the `createmakefiles.pl` script when you have finished working on the help files. For details, see *Building the Help Set* on page 14.

Opening A Help File

1. Choose **File - Open**

Browse to the file you want to open and select **Help (*.xhp)** as the file type

2. Click **Open**

Removing A Help File

Since a help file is referenced from multiple locations, simply deleting a file from disk is *not* sufficient for removing a help file from the set of help files.

To remove a help file from the set of help files that are compiled you need to remove it from the **makefile** of its directory. In this way, it will not be included in the index, or the full text search. However, it will still be included in the help files archive `*.jar`.

To delete a help file completely, you need to **remove it from your local disk** and remove its entry in the **makefile** of its directory. If you work on the CVS you also need to **remove it from the CVS repository**.

You also need to remove all **dependency files** in the output tree that are created during a help build that refer to the deleted file, see also *Building the Help Set* on page 14. If you haven't built the help set before you don't need to worry about this. If you have changed multiple files it is safer to remove the output tree completely and rebuild the help from scratch.

Finally, you need to ensure that the deleted file is not referenced by other help files or by the content files `*.tree` in the auxiliary directory.

To Remove A Help File

1. Delete the help file on your local disk.
2. If this change will also be committed back to the CVS, remove the help file from the CVS repository. [7]
3. In the `makefile.mk` of its directory, locate the help file's entry (it has the extension `.hzip` instead of `.xhp`) and delete the corresponding line.
4. Check if the help file is referenced in on of the `*.tree` files in `helpcontent2/source/auxiliary` and delete its reference there, if required.
5. If you have built the help before and you have an existing output tree with dependency files `*.dphh`, delete any dependency files that reference the deleted help file. A dependency file lists all files that it depends on.

Moving A Help File

From the build environment's view, moving a help file from one directory to another is equivalent to removing a file from one directory and creating a file in another directory.

1. Move the help files to the new directory.
2. Follow the procedure described above for removing a help file.
3. Ensure that all links inside the moved help file to itself have been adjusted.
4. Add the file to the `makefile` of the new directory.

Note that moving a help file may create new localization effort since it the moved help file looks like a brand new file to the localization process. However, translation memory systems should be able to automatically translate it because the content did not change - except for internal links.

[7] See <http://tools.openoffice.org/> for information about OpenOffice.org CVS

Sections And Paragraphs

Sections are used to specify parts of help files that are used for referencing purpose in other files. Sections can be embedded and linked to.

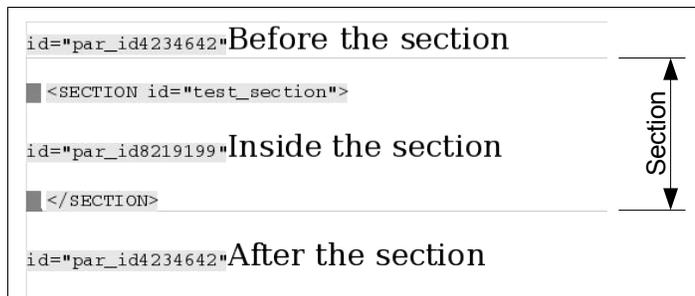
Where Are The Sections?

Since OpenOffice.org natively supports sections we make use of them to represent sections in help files. The ID attribute of a section in the help is represented by the name property of the section in OpenOffice.org.

All other properties of sections inside OpenOffice.org have no influence on the help sections. Any layout settings made to sections (background, visibility...) are lost on next reload.

You can use the navigator to view and navigate sections. Nested sections are also supported (both by the help and by OpenOffice.org).

Sections start and end with a `section` tag that is placed in the first paragraph directly after the section starts and in the last paragraph before the section ends:



The diagram shows a code editor window with a light gray background. It contains three paragraphs of text. The first paragraph is "id=\"par_id4234642\"Before the section". The second paragraph is "<SECTION id=\"test_section\">". The third paragraph is "id=\"par_id8219199\"Inside the section". The fourth paragraph is "</SECTION>". The fifth paragraph is "id=\"par_id4234642\"After the section". A vertical double-headed arrow on the right side of the code editor spans the height of the second and third paragraphs, with the word "Section" written vertically next to it.

If you want to insert something before or after a section ensure that you place it before or after the section delimiter line, not just before or after the `section` tag.

If the section **starts at the top** of the document and you want to insert something before that section, go to the top of the section and press `Alt+Return` to create a paragraph in front of the section.

If the section **ends at the bottom** of the document and you want to insert something after that section, go to the bottom of the section and press `Alt+Return` to create a paragraph after the section.

Adding A Section

1. Depending on whether you want to insert a new empty section or build a section around existing text, do one of the following:
 - Place the cursor where you want to insert the new empty section.
 - Mark the piece of the document that you want to include in a section.
2. Choose **Help Authoring - Insert Section**
3. Insert an identifier for the section in the text box.

This section identifier will be used as ID attribute for the section in the help file. It must be *unique within the file*. It is advisable to use some kind of descriptive name. Use only letters, numbers and the underscore.

Adding A Subsection

A subsection is a section that is the child of another section. OpenOffice.org supports nested sections. The procedure to insert a subsection is the same as inserting a section except for that if you insert a section with the cursor inside an existing section you will create a subsection.

You cannot create overlapping sections. Neither the help format nor OpenOffice.org support this.

Removing A Section

1. If you want to remove a section including its content, delete the section content first.
2. Choose **Format - Sections**.
3. Select the section you want to remove from the list of sections and click **Remove**.

If you remove a section that has subsections only the selected section will be removed while the subsections will be preserved.

If you remove a section ensure that no other file references it to avoid broken links.

Linking To A Section

You can create a link to a section by specifying the section ID as the target in the hyper-link URL when creating a link, for example

```
<link href="text/shared/guide/file_name.xhp#section_id">
```

For details, see *Linking* on page 91.

Embedding A Section

You can embed a section using the `embed` element. You need the file name and the `id` of the section that you want to embed. The embedded section preserves its structure. For details on embedding, see *Embedding Content* on page 90.

Adding A Paragraph

Paragraphs in the Help have some attribute values that cannot be represented in OpenOffice.org without using certain workarounds. Therefore, you need to follow the following instructions to create valid paragraphs.

You can write the paragraph content in the usual way. You only have to ensure that

- the paragraph has **meta information** associated with it
- the paragraph has a valid **paragraph format** assigned

The paragraph **meta information** consists of the paragraph `ID`, the `language` (which in the source files is always `en-US`), the `localize` attribute, and some other attributes that are not relevant in this context.

All of these values are stored in a **variable field** at the beginning of the paragraph. The paragraph `ID` identifies the paragraph contents in the localization database. If the `ID` of a paragraph gets lost or is changed it is regarded as new for the database and needs to be localized again. So messing with `IDs` must be strictly avoided.

Before saving the final file each paragraph must have a valid and unique `ID`. The easiest way to do this is to place the cursor somewhere in the paragraph and to choose **Help Authoring - Paragraph - Set Paragraph ID**. If the paragraph doesn't have the correct style associated (see below) ID creation will be denied.

Not all paragraphs get IDs. Some paragraphs only contain structural information, like opening and closing tags, or bookmarks, and don't need an ID because they don't transform back to content paragraphs in the help file. All these paragraphs have a paragraph style assigned that starts with `h1p_aux_`.

If you happen to forget to assign IDs to all corresponding paragraphs the export filter will do that for you. All paragraphs that lack an ID will be assigned one by the export filter when the file is saved. However, these will only be reflected on next reload of the file.

If you save a file with ID-less paragraphs the file on the disk has IDs assigned (because the export filter did that) but OpenOffice.org doesn't know. If you then assign paragraph IDs in OpenOffice.org and save these will override the IDs that were created using the export filter.

Editing A Paragraph

Editing the contents of a paragraph doesn't need any further action. The localization process finds out for itself when something changed in a paragraph.[8]

This has two consequences for the writers:

1. You need not worry whether a change has an influence on the localization or not.
2. You cannot force retranslation of a paragraph by just setting its editing flag.

For basic content management purposes the `l10n` attribute of a paragraph can be used since this was only relevant for the migration phase. For instance, you can set the paragraph status to `NEW` or `CHG` (changed) to allow reviewers to easily spot these paragraphs for content review. Any other values come from the migration phase and are no longer relevant. Paragraphs that have been reviewed don't carry an `l10n` attribute (or carry an empty one).

Note that this attribute is evaluated *nowhere*. It only influences the display of the paragraph in OpenOffice.org (the meta data appear with yellow background). If you want to use it you will have to take care for its evaluation in the context of content management yourself.

Paragraph Formatting

All paragraphs in a help file are formatted using paragraph styles. Direct formatting (borders, indentation, etc.) is *strictly forbidden*. In fact, all direct formatting will simply be *lost* on export.

Use the predefined styles to format the paragraphs. The following styles are available in the help authoring template (switch to the **Custom** view in the Stylist):

- `hlp_default`
This is the parent style for all `hlp_*` styles and only used as a fallback. It translates to a `paragraph@role="paragraph"` in the help file.
- `hlp_paragraph`
This is the standard style to be used for paragraphs. It translates to a `paragraph@role="paragraph"` in the help file.
- `hlp_head`, and `hlp_head1...hlp_head5`
The `hlp_head` style is the parent style for the other `hlp_head*` styles and should never be used as such. The `hlp_head*` styles designate heading elements of different level. They translate to `paragraph@role="heading"@level="x"` in the help file with `x` corresponding to the heading level.
- `hlp_listitem`
This is the style to be used for list items. Its use is optional, paragraphs inside list

[8] For details on the localization process see `l10n.openoffice.org`.

items may also have the paragraph style. It translates to a `paragraph@role="listitem"` in the help file.

- `hlp_tablehead`
This is the style to be used for table header cells. It translates to a `paragraph@role="tablehead"` in the help file when the `main_transform.xsl` stylesheet is used.
- `hlp_tablecontent`
This is the style to be used for table content cells. It translates to a `paragraph@role="tablecontent"` in the help file.
- A couple of `hlp_aux_*` styles
These are not meant to be used by the writers. These styles designate paragraphs that contain structural information rather than content.

Creating New Styles

If the styles in the pre-defined set are not sufficient for your purpose you can create new styles as following these rules:

- A new paragraph style **must** be based on the `hlp_default` style
- The style name **must** begin with `hlp_`
- The style name **must not** begin with `hlp_aux_`

You can use these styles in the help document. They will be transformed to values of the `role` attribute for a paragraph in the help file, e.g. `hlp_mystyle` will result in a paragraph with the `role` set to `mystyle`.

This style will be reconstructed when the help file is loaded. But any formatting information for OpenOffice.org will be lost. Also, the style will only be available in that file. If you want the style to be available for all documents and have a defined appearance it must be added to the help authoring template.

In order to have a special appearance in the final help the role must also be addressed in the stylesheets that are delivered with the help and define its appearance.

Changing A Paragraph Style

Changing the style of a paragraph has no impact on the localization process. Only the contents of a paragraph (including inline elements) are subject to localization.

Changing A Character Style

Changing the style of a character inside a paragraph does have an impact on the localization of that paragraph since the character style transforms to an `<item type="">` inline element.

Moving A Paragraph Inside A Help File

You can safely move a paragraph in a help file without the need of further actions. The paragraph styles may need adjustment if the paragraph is moved to a different context in a help file.

Ensure that you also move the paragraph meta data that are stored in the variable field at the start of the paragraph. If you **copy** a paragraph, however, **never** copy the meta data. The ID of a paragraph **must** be unique within a help file.

Moving A Paragraph To A Different Help File

Moving a paragraph from one file to another is a sequence of deleting and creating that can be accomplished by cutting and pasting the paragraph **without** its meta data.

For localization this will look as a new paragraph.

Excluding A Paragraph From Localization

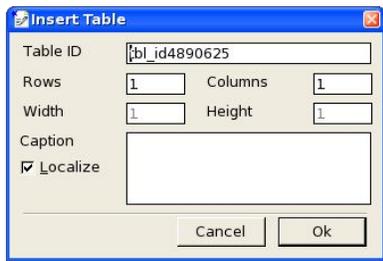
Paragraphs can be excluded from localization. In this case, the localized help files contain the English source for the corresponding paragraph. This is controlled by the `localize` attribute of a paragraph. If it is set to `false` the paragraph will not be localized, in **all other cases** it will.

To exclude a paragraph from localization choose **Help Authoring - Paragraph - Exclude from L10N**.

Tables

Adding A Table

1. Choose **Help Authoring - Table - Insert Table**



2. Insert the initial number of rows and columns in the corresponding text boxes.
You can change the table layout after creation, if required.
The `width` and `height` values are currently unsupported.
3. If required, insert a table caption in the **Caption** text field.
You can exclude the caption from localization by clearing the **Localize** check box.
4. Click **Ok**

Nested tables are unsupported. You cannot insert a table in another table.

The meta data of the table are stored "encoded" in the **Name** property of the table in OpenOffice.org. This must be left untouched.

The created table is followed by a paragraph containing the caption, if a caption was defined.

Modifying The Table Layout

After creation of the table you can change the table layout to suit your needs. You can add or remove rows or columns.

Initially, the column widths will be distributed equally. You can manually resize the column widths but for now this will *be lost* on next reload.

Never ever merge cells. Complex layouts are untested and may lead to unexpected results.

Deleting A Table

Delete a table as usual in OpenOffice.org. Make sure that the trailing paragraph with the caption is also removed.

Using A Table For Formatting Purposes

Don't do that.

There are still quite some places in the help files that use tables for formatting. We will try to get rid of these occurrences over time.

Adding A Caption To An Existing Table

When you have created a table and want to add a caption to it proceed as follows:

1. Place the cursor **after** the paragraph containing the table attributes.

In any other place the script will reject adding a caption.

2. Choose **Help Authoring - Table - Insert Table Caption**
3. Specify the caption text and click **Ok**.

Lists

Inserting, Removing, Modifying Lists

You can work with lists as you would usually do in OpenOffice.org if you note the following:

Interrupting A List

A list that is interrupted by a paragraph that is not part of the list and then continued with the next number as displayed below on the left is unsupported. If you create such a list in OpenOffice.org it will transform to the list below on the right after the reload. You will end up with two separate lists both starting with 1.

1. My
2. list
3. contains
important
4. stuff,
5. you
6. know.

after reloading



1. My
2. list
3. contains
important
1. stuff,
2. you
3. know.

A list where the paragraph is unnumbered but is still part of the list item is supported and will work fine. In OpenOffice.org you get this by pressing the **Backspace** key once to get rid of the list number.

1. My
2. list
3. contains
important
4. stuff,
5. you
6. know.

Images in lists can be placed in such paragraphs. There is no need to mimic lists using weird table constructions.

Working With Images

Help Image Repository

Help images are stored inside the `helpimg` subdirectory of the `res` CVS module. Images that are used by the help need to be added to this repository module. See tools.openoffice.org for details on working with the OpenOffice.org CVS repository.

The `helpimg` directory contains all help images in English. Subdirectories for each language (except for the source language which is `en-us`) contain the localized images. If an image doesn't need localization it only needs to be present in the `helpimg` directory. The subdirectories are named using the ISO codes for language and country as described on page 9.

To Add An Image To The Repository

1. Place the English image inside the `helpimg` directory of the `res` module.
2. Place the localized images inside the corresponding language subdirectories of `helpimg`, for example `zh-cn` for simplified Chinese.

If there is no localized image available the help will display the English image which resides in `helpimg`.

3. Open the file `helpimg.lst` in the `util` directory of the `helpcontent2` module and add the English and all localized variants to the file. Keep the file entries sorted.

To Remove An Image From The Repository

1. Remove the English and localized files from the CVS
2. Open the file `helpimg.lst` in the `util` directory of the `helpcontent2` module and remove the corresponding file entries.

Inserting A Block Image

A block image is an image that is located in a paragraph of its own. It can contain a caption.

1. Choose **HelpAuthoring - Image - Insert Image**.
2. Select an image file to insert and click **Open**.

The image **must** be located inside the help file hierarchy as described in *Setting Up the Environment* on page 71.

3. Specify an alternative text for the image (mandatory).

This text is needed to comply with accessibility regulations.

4. Specify a caption text for the image (optional).

The image will be added on a paragraph of its own surrounded by `img` tags. If you have specified a caption this caption text will appear inside `imgcaption` tags.

Inserting An Inline Image

An inline image is an image that is displayed inline in between paragraph text. It may not contain a caption.

1. Choose **HelpAuthoring - Image - Insert Inline Image**.

Select an image file to insert and click **Open**.

The image **must** be located inside the help file hierarchy as described in *Setting Up the Environment* on page 71.

2. Specify an alternative text for the image (mandatory).

This text is needed to comply with accessibility regulations. The image will be added to the paragraph surrounded by `img` tags.

Adding An Image Caption

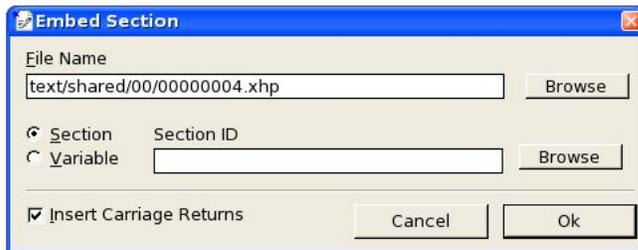
You can add a caption to an existing block image.

1. Choose **HelpAuthoring - Image - Insert Caption**.
2. Specify a caption and click **Ok**.

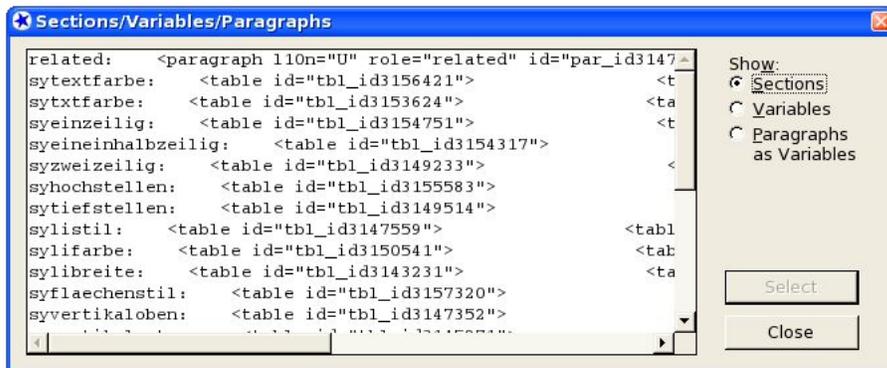
Embedding Content

Embedding A Section Or Variable

1. Choose **Help Authoring - Embed Sections or Variables**



2. Enter the name of the file that contains the section or variable to be embedded in the **File Name** text box or click **Browse** to browse for a file. The path starts with the `text` directory in the help directory hierarchy (see page 71).
3. Select whether a variable (text block in a paragraph or the complete contents of a paragraph) or a section is to be embedded.
4. Insert the section or variable ID or click **Browse** to browse all sections, variables, and paragraphs in the selected file.



Linking

Linking To Another Help File

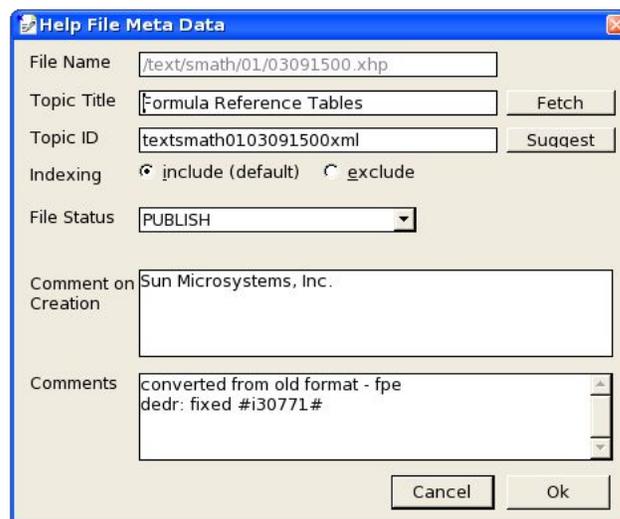
1. Mark the text that you want to appear as hyperlink.
2. Choose **Help Authoring - Insert Link**
3. Enter the name of the file to link to in the **Link target** box. The path starts with the **text** directory in the help file hierarchy. The path may contain a target anchor, for example, `text/swriter/01/01020304.xhp#anchor`
4. Click **Ok**

Linking To The WWW

Proceed as for links to help files but instead specify a WWW URL as link target.

Meta Data

The meta data are available through the **Help Authoring - Meta Data** menu that calls the **Meta Data** dialog:



Setting The Topic Title

On help file creation the topic title is set to a generic string. This must be changed before finally saving the file.

1. Choose **Help Authoring - Meta Data**

2. Insert a topic title in the corresponding text box or click **Fetch** to fetch the topic title from the first heading in the document.

The topic title may not be empty.

Setting The Topic ID

On document creation the topic id will be set from the file name. There is usually no need for setting the topic id manually but you can do so by entering the ID in the corresponding text box. Characters that are not allowed are automatically stripped from the ID. Clicking **Suggest** creates an ID based on the filename (like when the file is created).

Excluding A File From The Search Index

By default all files are included in the full text search index creation. You can exclude files from this search index by selecting the **exclude** option in the **Indexing** section.

Changing The Initial File Creation Comment

If you are ashamed of your initial comment you need to patch the `xhp` file.

Changing The Last Edited Comment

You can insert a comment when you edit and save a help file. This comment can be used to describe why a change was made and what changes were performed. A new comment overrides existing comments.

Bookmarks

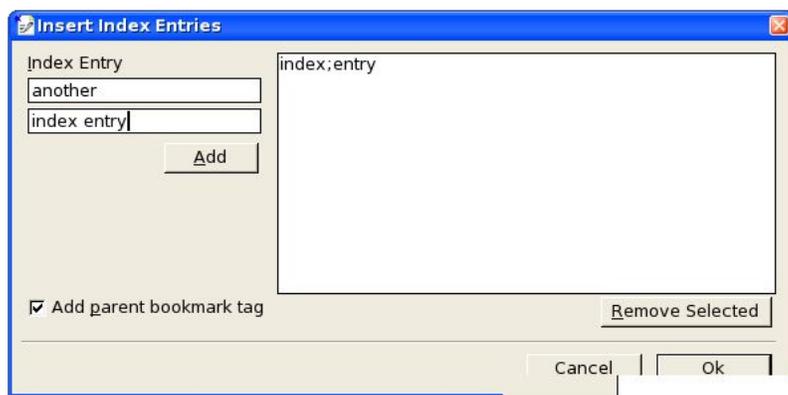
Bookmarks host index entries, help IDs, and entries for the table of contents (TOC)[9].

Adding A New Bookmark Set With Index Entries

1. Place the cursor where you want the index entry to appear.

Remember that an index entry transforms to an anchor target in the help file. Therefore, an index entry should always be placed directly **above** the text it refers to. Index entries that refer to the complete help topic should be placed at the top of the file.

2. Choose **Help Authoring - Bookmarks - Insert Index Entries**.



3. Enter the first and second level of the index entry in the **Index Entry** text boxes and click **Add** or press the **Ins** key to add it to the list of index entries.

You can remove index entries from the list by selecting them and clicking **Remove Selected**.

4. Select **Add parent bookmark tag** to create a new set of index entries. If you want to add index entries to an existing set you need to clear this box (see next procedure).
5. Click **Ok**.

Adding Index Entries To An Existing Bookmark Set

1. Place the cursor inside the set of index entries where you want to add index entries.

You cannot mix different types of bookmarks (index entries, help ids, and TOC entries).

[9] TOC entries are currently unused.

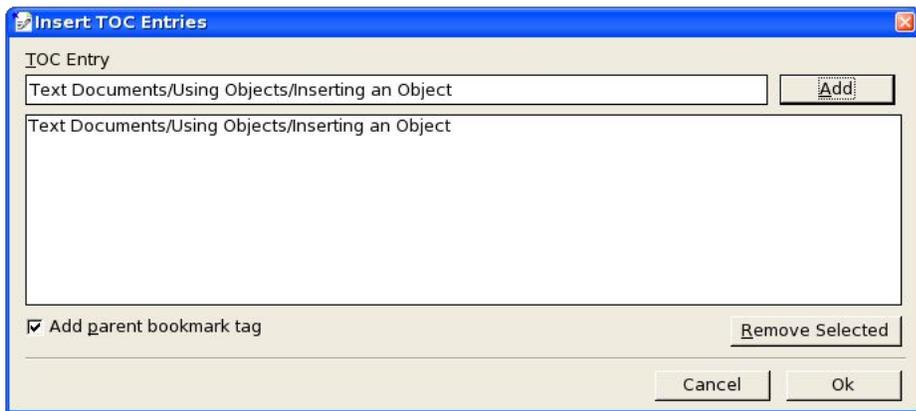
2. Choose **Help Authoring - Bookmarks - Insert Index Entries**.
3. Compile the list of index entries that you want to add to the bookmark set like described in the previous procedure.
4. Clear the **Add parent bookmark tag** box.
If the box is checked a *new* bookmark set with the specified index entries will be created after the set at the cursor position.
5. Click **Ok**.

Modifying Index Entries In An Existing Bookmark Set

If you need to modify an existing index entry (e.g., correct a typo) delete and recreate this index entry as described above.

Adding A New Bookmark Set With TOC Entries[10]

1. Place the cursor where you want the TOC entry to appear.
2. Remember that a TOC entry transforms to an anchor target in the help file. Therefore, a TOC entry should always be placed directly **above** the text it refers to.
TOC entries that refer to the complete help topic must be placed at the start of the help topic.
3. Choose **Help Authoring - Bookmarks - Insert TOC Entries**.



4. Enter the TOC entry string in the **TOC Entry** text box and click **Add** to add it to the list of TOC entries.

The TOC levels are separated using a slash /, for details see page 21.

[10]These bookmarks are currently not evaluated when the help is compiled.

You can remove TOC entries from the list by selecting them and clicking **Remove Selected**.

5. Select **Add parent bookmark tag** to create a new bookmark set of TOC entries. If you want to add TOC entries to an existing bookmark set you need to clear this box (see next procedure).
6. Click **Ok**.

Adding TOC Entries To An Existing Bookmark Set

1. Place the cursor inside the bookmark set of TOC entries where you want to add TOC entries.

You cannot mix different types of bookmarks (index entries, help ids, and TOC entries).

2. Choose **Help Authoring - Bookmarks - Insert TOC Entries**.
3. Compile the list of TOC entries that you want to add to the bookmark set like described above.
4. Clear the **Add parent bookmark tag** box.
5. If the box is checked a *new* bookmark set with the specified TOC entries will be created after the set at the cursor position.
6. Click **Ok**.

Determining A Help ID

The help ID inserted into the help file must either be the **symbolic ID** or an **UNO command** (see "*hid*" Branch on page 22). You can determine the **numerical ID** or the **UNO command** from the UI by setting an environmental variable `HELP_DEBUG` and setting it to `TRUE` before you start OpenOffice.org.

If the variable is set you will see the help ID of an element together with its extended tip whenever you rest the mouse over it (provided the extended tips are enabled). This help ID can either be

- a **numerical ID**, in this case it must be converted to the symbolic ID before inserting it into the help file (see below)
- an **UNO command** this can be inserted into the help file without need for conversion

To convert the numerical help id into a symbolic help id you need a matching table called `help_hid.lst` that can be found in the `helpers` directory of the `helpcontent2` module (see *Structure of the CVS Help Module* on page 13).

You can either use this mapping table to look up a symbolic help id yourself, or you can place it into your local user/configuration directory of OpenOffice.org to allow the corresponding help authoring macro convert it for you.

Adding A Help ID

1. Place the cursor where you want the Help ID to appear.

Remember that a Help ID transforms to an anchor target in the help file. Therefore, the Help ID must be placed directly **above** the text it refers to and above any extended tip that it corresponds to.

Help IDs that refer to the complete help topic must be placed at the beginning of the help topic.

2. Choose **Help Authoring - Bookmarks - Insert Help ID**.
3. Insert the Help ID in the **Help ID** text box
4. If you only have the numerical help ID, click **Convert to Symbol** to convert it to the symbolic Help ID. If this button is disabled you need to place a `help_hid.lst` file into the `user/configuration` directory of your OpenOffice.org installation.
5. Click **Ok**.

Switching Content

Inline Switching

Inline switching uses conditional tags to switch parts of a paragraph for different context situations. An inline switch consists of an outer `switchinline` element that encloses one or more `caseinline` elements that define the conditions and optionally one `defaultinline` element. The complete switch must be in one paragraph.

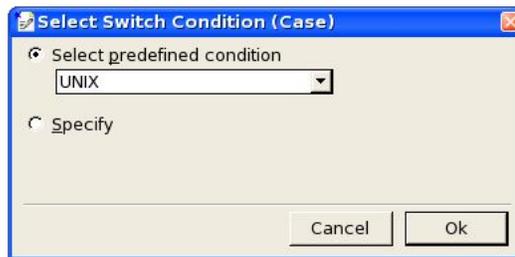
```
<switchinline select="switch_type">
  <caseinline select="condition_1"></caseinline>
  <caseinline select="condition_2"></caseinline>
  ...
  <defaultinline></defaultinline>
</switchinline>
```

1. Place the cursor where you want the inline switch to start or select the text passage that you want in the first condition.

2. Choose **Help Authoring - Switching - Open Switchinline**



3. Select a switch type from the dialog. Currently, there are three switch types available:
 - **System** switches are used to switch between different platforms (Windows, Unix,...).
 - **Application** switches are used to switch between different applications (Writer, Calc, Draw,...)
 - **Distribution** switches are used to switch between open source and commercial distributions (OpenOffice.org, StarOffice,...) [11]
4. Click **Ok**
5. Select the first condition (`caseinline`).



You can either select one of the pre-defined conditions from the list or specify your own condition string.

The pre-defined conditions are processed when the help is displayed. If you want to specify your own condition string you will have to ensure that the condition is processed when the help is compiled (see page 14) and displayed. Usually, you will only use the pre-defined conditions.

6. Now enter further conditions by selecting text and choosing **Help Authoring - Switching - Insert Caseinline** for the corresponding switch type.

[11] Not currently evaluated.

There is no text allowed (including spaces or line breaks) between a closing and an opening tag inside an inline switch, e.g.

```
</caseinline> <caseinline> wrong  
</caseinline><caseinline> correct
```

7. Optionally insert a default condition by selecting text and choosing **Help Authoring - Switching - Insert Defaultinline**.
8. Insert a closing `switchinline` element **directly** after the last `caseinline` element by choosing **Help Authoring - Switching - Close Switchinline**.

If you have inserted a default condition like described above the `switchinline` element will automatically be closed.

If you are not sure if you have actually created a valid switch choose **Help Authoring - Validate** and you will be notified of any errors.

Switching Complete Sections Or Paragraphs

Other than inline switches this type of switches one or more paragraphs including graphics and tables. Similar to inline switches they consist of an outer `switch` element that encloses one or more `case` elements that define the conditions and optionally one `default` element. Each of those elements must be in a paragraph of its own that is assigned the `hlp_aux_switch` style. The macros handle all that for you.

1. Place the cursor where you want the switch to start.

If you are in a non-empty paragraph the switch will start **before** that paragraph.

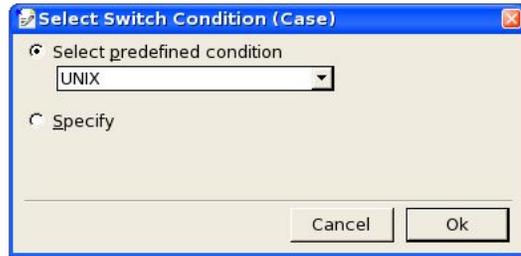
2. Choose **Help Authoring - Switching - Open Switch**



3. Select a switch type from the dialog. Currently, there are three switch types available:

- **System switches** are used to switch between different platforms (Windows, Unix,...).
- **Application switches** are used to switch between different applications (Writer, Calc, Draw,...)
- **Distribution switches** are used to switch between open source and commercial distributions (OpenOffice.org, StarOffice,...).

4. Select the first condition (*case*).



You can either select one of the pre-defined conditions from the list or specify your own condition string.

The pre-defined conditions are processed when the help is displayed. If you want to specify your own condition string you will have to ensure that the condition is processed when the help is compiled (see page 14) and displayed. Usually, you will only use the pre-defined conditions.

5. Place the cursor where the condition ends.

If you are in a non-empty paragraph the condition will end **after** that paragraph.

6. Choose **Help Authoring - Switching - Close Case** to close a condition.
7. Now insert further conditions by

- placing the cursor in the *first* paragraph of the condition
- choosing **Help Authoring - Switching - Open Case**
- placing the cursor in the *last* paragraph of the condition
- choosing **Help Authoring - Switching - Close Case**.

There are no paragraphs allowed between a closing and an opening tag inside an switch, e.g.

Wrong:

```
</case>
{Some other text, even an empty paragraph}
<case>
```

Correct:

```
</case>
<case>
```

8. You can optionally enter a default condition by

- placing the cursor in the *first* paragraph of the default condition
- choosing **Help Authoring - Switching - Open Default**

- placing the cursor in the *last* paragraph of the default condition
 - choosing **Help Authoring - Switching - Close Default**.
9. Close the switch by placing the cursor **directly** behind the last `case` element and choosing **Help Authoring - Switching - Close Switch**.

If you have inserted a default condition the switch will automatically be closed.

If you are not sure if you have actually created a valid switch choose **Help Authoring - Validate** and you will be notified of any errors.

Miscellaneous

Extended Tips

Extended tips come in two flavors, **visible** and **hidden**. Visible extended tips are part of the normal help text while hidden extended tips are hidden from the normal help content. Each extended tip is assigned a help id to which it responds.

In the current implementation, the `hid` attribute of the extended tip elements `AVIS` and `AHID` are not evaluated. The corresponding Help IDs must be placed as bookmarks in a paragraph before the extended tip. The extended tip will be shown for *all help IDs* specified as bookmarks *after the last extended tip* in the file.

1. Select the part of the paragraph that you want to use as extended tip.
An extended tip must not be spread over multiple paragraphs.
2. Choose **Help Authoring - Insert Visible Extended Tip** or **Help Authoring - Insert Hidden Extended Tip**.
If you enclose text by a hidden extended tip this text portion will *no longer be visible* in the help viewer.
3. Insert a help ID to be assigned to the extended tip. Note the comment about help IDs above.

Sorting

Sorting is a new feature in OpenOffice.org Help. It can be used to sort sections based on their content. This is useful, e.g., for glossaries or other sorted lists that are localized and would otherwise lose their correct sort order.

1. Place the cursor *before* the first section to be sorted.

In the current implementation the `sort` element sorts the content of sections. The `sort` element must not have any other child elements than `sections`.

2. Choose **Help Authoring - Sorting - Open Sort**
3. Place the cursor *after* the last section to be sorted.
4. Choose **Help Authoring - Sorting - Close Sort**

Validating

There is a basic validation procedure available that tests the help files before they are exported from OpenOffice.org. To call it choose **Help Authoring - Validate**.

Note that this procedure catches some of the most common and severe errors but it is not fool-proof. It does *not* perform a XML validation on the file.

Troubleshooting

A Help File Cannot Be Opened

The reason for that is probably an invalid help `xhp` file. To verify this, open the help file in any XML or text editor and check its validity. Fix any invalid syntax and reload the file.

If the XML file is valid and you cannot open *any* help file, see below.

A Help File Cannot Be Saved

The reason for that could be insufficient access rights to the directory or file you want to save to. Change the access rights accordingly.

If you cannot save *any* help file, see below.

No Help File Can Be Opened Or Saved

The reason for that is probably a corrupted XSLT export/import filter. Occasionally, installing the help authoring filter produces an error in the XSL files. To repair this proceed as follows:

If You Are Familiar With XSLT

1. Change to the `user/xslt/Help` directory of your OpenOffice.org installation.
2. Open the import and/or export xsl file.
3. Got to the end of the file and check if there is any obvious duplicated content. Occasionally, the last lines of the stylesheet get duplicated. There must only be one single `</xsl:stylesheet>` tag in the file.

If You Are Unfamiliar With XSLT

1. Get the help authoring filter package (see *Installing the Import/Export Filters* on page 72) and unpack it to a temporary directory.
2. Copy the *.xsl files from the Help subdirectory of the files that you just unpacked to the user/xslt/Help directory of your OpenOffice.org installation overwriting the existing files.
3. Adjust the filters as described in *Customizing the Import/Export Filters* on page 73.

The Image Or File Links In The Exported Help Files Are Wrong

The reason for that are probably wrong path settings in the import and export filters.

1. Change to the user/xslt/Help directory of your OpenOffice.org installation.
2. Open the import and export xsl file.
3. Adjust the path settings as described in *Customizing the Import/Export Filters* on page 73.

Paragraph Content Has Vanished On Reload

The reason for that probably is the use of a wrong paragraph format. Remember, that for paragraphs with content you must use one of the predefined h1p_* paragraph styles. See also *Paragraphs and Paragraph Formatting* on page 75.

Appendix

XML Help DTD

```
<!--
Version 20-Apr-2004
-->

<!ELEMENT ahelp (#PCDATA | embedvar | br | comment | emph | item | link |
switchinline | variable)*>
<!ATTLIST ahelp
  hid CDATA #REQUIRED
  visibility (hidden | visible) #IMPLIED
>

<!ELEMENT alt (#PCDATA)>
<!ATTLIST alt
  xml-lang CDATA #REQUIRED
  id CDATA #REQUIRED
  localize CDATA #IMPLIED
>

<!ELEMENT body (section | paragraph | table | comment | bookmark | switch |
embed | list | sort)*>

<!ELEMENT bookmark (bookmark_value)*>
<!ATTLIST bookmark
  branch CDATA #REQUIRED
  xml-lang CDATA #REQUIRED
  id CDATA #REQUIRED
  localize CDATA #IMPLIED
>

<!ELEMENT bookmark_value (#PCDATA | embedvar)*>

<!ELEMENT br EMPTY>

<!ELEMENT caption (#PCDATA | embedvar | br | emph | item | link | switchinline
| variable)*>
<!ATTLIST caption
  xml-lang CDATA #REQUIRED
  id CDATA #REQUIRED
  localize CDATA #IMPLIED
>

<!ELEMENT case (paragraph | table | comment | bookmark | embed | link | list |
switch | section | sort)*>
<!ATTLIST case
  select CDATA #REQUIRED
```

```
>

<!ELEMENT caseinline (#PCDATA | image | embedvar | br | emph | item | link |
switchinline | variable | ahelp | object)*>
<!ATTLIST caseinline
  select CDATA #REQUIRED
>

<!ELEMENT comment (#PCDATA)>

<!ELEMENT created (#PCDATA)>
<!ATTLIST created
  date CDATA #REQUIRED
>

<!ELEMENT default (paragraph | table | comment | bookmark | embed | link |
list | switch | section | sort)*>

<!ELEMENT defaultinline (#PCDATA | image | embedvar | br | emph | item | link
| switchinline | variable | ahelp | object)*>

<!ELEMENT embed EMPTY>
<!ATTLIST embed
  href CDATA #REQUIRED
  role CDATA #IMPLIED
  level CDATA #IMPLIED
>

<!ELEMENT embedvar EMPTY>
<!ATTLIST embedvar
  href CDATA #REQUIRED
  markup (keep | ignore) #IMPLIED
>

<!ELEMENT emph (#PCDATA | item | comment)*>

<!ELEMENT filename (#PCDATA)>

<!ELEMENT helpdocument (meta, body)>
<!ATTLIST helpdocument
  version CDATA #REQUIRED
>

<!ELEMENT history (created, lastedited)>

<!ELEMENT image (caption* | alt+)?>
<!ATTLIST image
  src CDATA #REQUIRED
  width CDATA #IMPLIED
  height CDATA #IMPLIED
  id CDATA #REQUIRED
>

<!ELEMENT item (#PCDATA)>
<!ATTLIST item
  type CDATA #REQUIRED
>

<!ELEMENT lastedited (#PCDATA)>
<!ATTLIST lastedited
  date CDATA #REQUIRED
```

```

>
<!ELEMENT link (#PCDATA | embedvar | emph | item | variable | switchinline)*>
<!ATTLIST link
  href CDATA #REQUIRED
  name CDATA #IMPLIED
  type CDATA #IMPLIED
  target CDATA #IMPLIED
>

<!ELEMENT list (listitem | (comment)*)+>
<!ATTLIST list
  type CDATA #REQUIRED
  startwith CDATA #IMPLIED
  format (1 | i | I | a | A) #IMPLIED
  bullet (disc | circle | square) #IMPLIED
  sorted (asc | desc) #IMPLIED
>

<!ELEMENT listitem (comment | section | paragraph | table | switch | embed |
bookmark)*>
<!ATTLIST listitem
  format (1 | i | I | a | A) #IMPLIED
  bullet (disc | circle | square) #IMPLIED
  class CDATA #IMPLIED
>

<!ELEMENT meta (topic, history?)>

<!ELEMENT object EMPTY>
<!ATTLIST object
  type CDATA #REQUIRED
  id CDATA #REQUIRED
  data CDATA #REQUIRED
  height CDATA #IMPLIED
  width CDATA #IMPLIED
>

<!ELEMENT paragraph (#PCDATA | image | comment | embedvar | br | emph | item |
link | switchinline | variable | ahelp | object | bookmark)*>
<!ATTLIST paragraph
  role CDATA #REQUIRED
  level CDATA #IMPLIED
  id CDATA #REQUIRED
  l10n CDATA #IMPLIED
  xml-lang CDATA #REQUIRED
  oldref CDATA #IMPLIED
  localize CDATA #IMPLIED
>

<!ELEMENT section (section | paragraph | table | list | comment | bookmark |
embed | switch | sort )*>
<!ATTLIST section
  id CDATA #REQUIRED
>

<!ELEMENT sort (section+)>
<!ATTLIST sort
  order (asc | desc) #IMPLIED
>

```

```
<!ELEMENT switch ((case | comment)* | default?)*>
<!ATTLIST switch
  select (sys | appl | distrib | target | lang | ver) #REQUIRED
>

<!ELEMENT switchinline ((caseinline)+, (defaultinline?))>
<!ATTLIST switchinline
  select (sys | appl | distrib | target | ver | lang) #REQUIRED
>

<!ELEMENT table (caption*, tablerow+)>
<!ATTLIST table
  name CDATA #IMPLIED
  width CDATA #IMPLIED
  height CDATA #IMPLIED
  unit CDATA #IMPLIED
  class CDATA #IMPLIED
  id CDATA #REQUIRED
>

<!ELEMENT tablecell (section | paragraph | comment | embed | bookmark | image
| list)*>
<!ATTLIST tablecell
  colspan CDATA #IMPLIED
  rowspan CDATA #IMPLIED
  width CDATA #IMPLIED
  class CDATA #IMPLIED
  unit CDATA #IMPLIED
>

<!ELEMENT tablerow (tablecell+)>
<!ATTLIST tablerow
  height CDATA #IMPLIED
  class CDATA #IMPLIED
  unit CDATA #IMPLIED
>

<!ELEMENT title (#PCDATA)>
<!ATTLIST title
  xml-lang CDATA #REQUIRED
  id CDATA #REQUIRED
  localize CDATA #IMPLIED
>

<!ELEMENT topic (title+, filename, bookmark*)>
<!ATTLIST topic
  id CDATA #REQUIRED
  indexer (exclude | include) #IMPLIED
  status (DRAFT | FINAL | PUBLISH | STALLED | DEPRECATED) #IMPLIED
>

<!ELEMENT variable (#PCDATA | ahelp | embedvar | br | emph | item | link |
variable | image | object | switchinline)*>
<!ATTLIST variable
  id CDATA #REQUIRED
  visibility (hidden | visible) #IMPLIED
>
```

Glossary

Application – A OpenOffice.org "module" for different document types. There are the following applications: *Writer* for text documents, *Calc* for spreadsheets, *Impress* for presentations, *Draw* for drawings, *Math* for formulas, *Basic* for Macros.

Active Help - A synonym for an extended tip.

Anchor – A location inside a help file which serves as a bookmark to which the help viewer jumps on displaying the help for a certain context.

Attribute – Component of an XML element carrying information that specifies the element in greater detail, for example, the `role` attribute in the `paragraph` element.

Bookmark – 1. A help function that allows to set user-defined bookmarks to help topics to make them easier to access. 2. An element of a help XML file that is used to define anchor points for help ids or keywords.

Build List – The file `build.lst` controls the build process of a module by defining module directories to be built and application dependencies between them.

Cascading Style Sheet – The style sheet used to define the layout of a help page displayed in the help viewer.

Context-Sensitive Help – When called from within the OpenOffice.org application the help receives information about the user interface context (like active dialog, selected element). This information is used by the help to display information related to that context provided this relation is defined in the help files. *Help IDs* are used to define this relation.

CVS – Concurrent Versioning System, a widespread version control system that is also used by OpenOffice.org. See `tools.openoffice.org`.

CVS Module – A part of the CVS that contains code for a section of the OpenOffice.org product.

Dependency Files – When a module is re-compiled only changed files and files that depend on them need to be compiled again. Dependency files describe these

dependencies. These files are used by the `make` utility.

DTD – Document Type Definition, a file that describes the document syntax for an XML document. The DTD is needed to validate an XML file.

Embedding – In OpenOffice.org help files can contain references to parts of other help files that are dynamically inserted when the help is displayed.

Extended Tip – Yellow "bubble" on the application user interface that contains information about the element under the mouse cursor. Extended tips appear when the mouse cursor rests over a user interface element. In OpenOffice.org 1.1.x they are enabled/disabled using **Help – Extended Tips**.

Full-Text Search – A help function that allows to search through the text of the set of help files. The function uses a search index that is created when the help files are compiled and built. Help files can be excluded from this search index using the `exclude` value in the `indexer` attribute of the `topic` element.

Help Authoring Template – The help authoring filter contains XSLT import/export filters and a help authoring template that specifies the layout of the help documents inside OpenOffice.org.

Help Compiler – A program that compiles the help files into an intermediate "object" format that is used by the help linker to assemble the final help files that are installed with OpenOffice.org.

Help Content Provider – A service inside OpenOffice.org that provides the Help to the help viewer.

Help IDs – Numerical or symbolic identifiers that are defined for user interface elements in the application code. Help IDs can be used to identify the context in which the help is called and to define a relation between an application context and the help topic that is displayed.

Help Module – Each OpenOffice.org application has a help module associated:
Writer, Calc, Draw, Impress, Math, Basic.

Help Section – A subdirectory of `helpcontent2/source/text`. Each help module contains the help files of one or more help sections.

Help Topic – The contents of a help file. Usually, a help topic describes one task or a logical group of reference information.

Help Viewer - OpenOffice.org component that displays the help files and provides help functionality.

Icon – An image that is taken from the resource repository of the application itself. Icons are stored in different CVS modules and after installation are available in the `images.zip` file.

- Image** – Graphical content that is specific to the help files. All images are stored in the `helpimg` directory of the `res` CVS module and after installation are available in the `images.zip` file.
- Block Image** – An image that is on a paragraph of its own. Block images can have captions.
- Inline Image** – An image that is part of another paragraph and surrounded by text content. Inline images may not have captions.
- Import/Export Filter** – XSLT files that control the conversion of the help files from `xhp` to OpenOffice.org and vice versa. Using a template they also control the appearance of the files in OpenOffice.org.
- Index of Keywords** – A two-level list of keywords associated with help topics. Keywords are explicitly defined in the help files.
- Instructional Information** – Information in OpenOffice.org help that provides instructions on how to fulfill tasks.
- makefile** – File that describes the processes for "making" (compiling/linking) files inside a directory. Used by the `make` utility.
- Meta Data** – Help file data that describe the help file, like file name, topic title, creation date. These are stored inside the `meta` element of the help file.
- Nested Sections** – Sections containing other subsections. Nested sections are supported in the help files.
- Nested Tables** – Tables containing other tables. Nested tables are *unsupported* in the help files.
- Node** – A node is a part of the help content tree that is used to group help topics. See "*Contents*" Branch on page 21.
- Output Tree** – A directory tree (aka *solver*) that takes all files that are produced on "making" (compiling/linking) source files. See `tools.openoffice.org`.
- Platform** – Operating System, like Linux, Solaris x86, Solaris SPARC, or Windows.
- Reference Information** – Information in OpenOffice.org help that explains the effect or function of a user interface element.
- Role** – In the help files that type of a paragraph is specified by its `role` attribute.
- Section Delimiter Line** – A section inside OpenOffice.org is delimited by two gray lines.
- Solver** – see *Output Tree*.

Style Sheet – A document containing commands for transforming an XML file (transformation style sheet) or for displaying an XML or HTML file (cascading style sheet).

Symbolic Name – The help IDs used in the applications can be transformed to symbolic names that are defined in the list of help ids, `hid.lst`. They are symbolic identifiers that give the number a somewhat descriptive name.

Tool Tip – A synonym for an extended tip.

Topic ID – Each help file (aka topic) has a unique topic id to be identified. It usually is created from the help file name.

Transformation Style Sheet – The style sheet used for transformation.

Transformation – In this context, the process of converting the XML format of the help document. The major transformation takes place when the help is displayed. The transformation style sheet `main_transform.xsl` is used for that.

UNO Command Name – One type of help ids that is used in the applications. Other than "normal" help ids which are numerical, these command names are symbolic identifiers and don't need to be converted.

Validation – The process of checking the validity of a help file. See *Validating* on page 101.