User's manual for

Writer2LaTeX

Writer2LaTeX, Writer2BibTeX, Writer2xhtml and Calc2xhtml

version 1.4

© 2002–2014 Henrik Just

.0cmTable of Contents

1. Introduction

1.1. What is Writer2LaTeX?

Writer2LaTeX is a utility to convert *OpenDocument* text and spreadsheet documents[1] – in particular documents containing formulas – into other formats.

Actually it is a collection of four converters:

- **Writer2LaTeX** converts OpenDocument text documents to LaTeX 2e, and works together with...
- **Writer2BibTeX** which extracts bibliographic data from an OpenDocument text document and converts it to BibTeX format.
- **Writer2xhtml** converts OpenDocument text documents to XHTML 1.0 strict, XHTML 1.1, XHTML 1.1 + MathML 2.0, HTML5 or EPUB using CSS2 to convert style information.
- **Calc2xhtml** converts OpenDocument spreadsheet documents to XHTML 1.0 strict, XHTML 1.1 or HTML5, using CSS2 to convert style information.

Although Writer2LaTeX is a general OpenDocument converter, it is primarily designed for use with LibreOffice (or Apache OpenOffice.org). You can use Writer2LaTeX

- ...as an *export filter* for LibreOffice
- ...as a *command line utility*, independent of LibreOffice.
- ...as a *Java library* providing conversions from OpenDocument for other Java programs.

[1] In addition, Writer2LaTeX supports the old file formats for OpenOffice.org 1.x Writer and Calc.

Writer2LaTeX is a Java application, and thus should work on any platform that supports Java. You need a Java Runtime environment, **version 6** or later. Writer2LaTeX is developed and tested using OpenJDK (`http://openjdk.java.net/`).

This user's manual will explain how to install and use Writer2LaTeX.

Note: In this manual LO is used as an abbreviation of LibreOffice.

1.2. More about Writer2LaTeX and Writer2BibTeX

Writer2LaTeX is quite flexible: It can take advantage of several LaTeX packages, such as `hyperref`, `pifont`, `ulem`. It can create customized LaTeX code based on the styles and text in the document. Also it supports more than 25 different languages, latin, greek and cyrillic scripts and 8 input-encodings.

The flexibility makes it possible to use Writer2LaTeX from several philosophies:

- You can use LaTeX as a typesetting engine for your LO documents: Writer2LaTeX can be configured to create a LaTeX document with as much formatting as possible preserved. Note that the resulting LaTeX source will be readable, but not very clean.

  Be aware that even though Writer2LaTeX tries hard to cope with any document, you will only get good results for well structured documents, ie. documents that are formatted using *styles*. For other documents you will find that Writer2LaTeX uses the principle *garbage in – garbage out*!

- If you need to continue the work on your document in LaTeX your primary interest may be the content rather than the formatting. Writer2LaTeX can instructed to produce a LaTeX document which strips most of the formatting and hence produces a clean LaTeX source from *any* source document.

- Traditionally, LaTeX documents are written by hand using a text editor. Using a graphical frontend like LyX provides a more user friendly alternative. A companion extension named *Writer4LaTeX* is in available and provides the tools to make you use LO as a graphical frontend for LaTeX.

1.3. More about Writer2xhtml and Calc2xhtml

The primary goal for Writer2xhtml and Calc2xhtml is to provide *standards compliant* XHTML documents which can be customized to your specific needs.

- Standards compliance is necessary to ensure consistent results when the document is viewed in different browsers. It is also vital to ensure that the created document can be processed further by other tools.
- Customization means that you can control important aspects about the conversion. In particular you can control the style of the document:
  - You can let Writer2xhtml convert the style information in the source document and thus get an XHTML document that has the same general appearance as the original, but is adapted to an online environment.
  - You can create a document that adapts the style of the document to your own CSS style sheet.

2. Using the export filters

2.1. Installing of the filters

Writer2LaTeX can work as an export filter for LibreOffice Writer.

Two LO extensions are provided:

- **writer2latex.oxt** installs the LaTeX and BibTeX export filters in Writer
- **writer2xhtml.oxt** installs the XHTML and EPUB export filters in Writer and Calc

The two extensions are independent, you can install one or both depending on your needs.

Note: Before you install the Writer2LaTeX extensions, you need to set up LO to use Java. You can configure this in LO under **Tools – Options**. Of course this requires that you have installed a Java runtime environment on your system.

The extensions are installed and uninstalled using the Extension Manager in LO. If you need instructions about using the Extension Manager, see

`https://help.libreoffice.org/Common/Extension_Manager`

In case of installation troubles, please see the FAQ on Writer2LaTeXs web page:

http://writer2latex.sourceforge.net.

2.2. Using the filters

The filters provided by Writer2LaTeX are all *export filters*. This means that the filters are to be found in the **File – Export** menu in Writer or Calc.

Note: As Writer2LaTeX does not provide corresponding *import* filters, you should always save in OpenDocument format as well!

2.3. Using Writer2LaTeX and Writer2BibTeX

To export a Writer document to LaTeX, choose **LaTeX 2e** in the export dialog.

After you have typed in a file name, an options dialog will open. To get help, select an item and press **F1** or press the help button. Alternatively you can enable extended tips with **Shift-F1**.

Click **Export** to initiate the export or **Cancel** to close the dialog without exporting the document.

Normally you would export the bibliographic data to BibTeX as part of the export to LaTeX, but you may also export the bibliographic data alone. To do this, choose **BibTeX** in the export dialog. All bibliographic data in the document will be extracted and stored in a BibTeX file which can later be used by e.g. LaTeX documents.

2.4. Using Writer2xhtml

To export a Writer document to XHTML, choose one of the following formats in the export dialog:

- **XHTML 1.0 strict** will create an XHTML file which is compatible with the older HTML 4 standard. You can thus expect that the result will be viewable with any browser, but note that mathematical formulas are *not* supported.

- **XHTML 1.1** will create an XHTML file using the XHTML 1.1 standard, but without support for mathematical formulas.
- **XHTML 1.1 + MathML 2.0** will create an XHTML file which follows the standard for combining XHTML with mathematical formulas, using *MathML* for the formulas. Unfortunately, not all browsers support this.
- **HTML5** will create a HTML5 file using MathML for mathematical formulas and SVG for vector graphics. This format should be preferred for contemporary browsers.
- **EPUB** will create an electronic e-book suitable for viewing on variety of devices.

In all cases, Writer2xhtml uses CSS to format the document, either by converting the original formatting to CSS or by using a CSS style sheet selected by the user.

Note that the default file extension and the recommended MIME types varies with the output format:

Output format	*Default file extenstion*	*MIME type*
XHTML 1.0	`.html`	`text/html`
XHTML 1.1	`.xhtml`	`application/xhtml+xml`
XHTML 1.1 + MathML 2.0	`.xhtml`	`application/xhtml+xml`
HTML5	`.html`	`text/html` or `application/xhtml+xml`

Output format	*Default file extenstion*	*MIME type*
EPUB	`.epub`	`application/epub+zip`

After you have typed in a file name, an options dialog will open. To get help, select an item and press **F1** or press the help button. Alternatively you can enable extended tips with **Shift-F1**.

Click **Export** to initiate the export or **Cancel** to close the dialog without exporting the document.

2.5. Using Calc2xhtml

To export a Calc document to XHTML, choose **XHTML 1.0 strict**, **XHTML 1.1** or **HTML 5** in the export dialog.

After you have typed in a file name, an options dialog will open. To get help, select an item and press **F1** or press the help button. Alternatively you can enable extended tips with **Shift-F1**.

Click **Export** to initiate the export or **Cancel** to close the dialog without exporting the document.

2.6. Custom configuration

Each of the exports provides the possibility to use a custom format/style. To edit this, choose **Tools – Options – Writer2LaTeX** resp. **Writer2xhtml.**

All three exporters uses a configuration file in the user installation folder for LO.

- On unix-like systems this folder will usually be something like

```
home directory/.config/libreoffce/4/user
```

- On Windows it will usually be something like

  ```
  C:\Documents and Settings\username\OpenOffice.org2\user
  ```

  or

  ```
  C:\Documents and Settings\username\
  Application Data\LibreOffice\4\user
  ```

  (Note that this directory may be hidden.)

Writer2LaTeX uses a file named `writer2latex.xml`, and Writer2xhtml and Calc2xhtml shares a file named `writer2xhtml.xml`. These files are created automatically the first time you use the custom configuration.

See section 4 for the structure of the configuration file.

2.7. Configuration packages

Advanced users may add further formats/styles to the lists in the export dialog. This is done using *configuration packages*, which are custom extensions to LO containing further configurations for Writer2LaTeX or Writer2xhtml.

A configuration package can contain:

- A configuration file for Writer2LaTeX or Writer2xhtml, see section 4.
- An XHTML template (Writer2xhtml only).

- An LO template.
- An LO registry file to glue the parts together.

The Writer2LaTeX distribution contains a sample configuration package **xhtml-config-sample.oxt** that demonstrates this.

As a demonstration of the principles of configuration packages, you can install this into LO using the Extension Manager:

- If you export to XHTML, the dialog will show an additional entry **Sample custom style** in the Style list.
- If you open **Templates and Documents** in LO you will find a new folder **xhtml-sample-config**. This folder contains a Writer template. If you create a document based on this template, **Sample custom style** will be preselected when you export to XHTML.

You can create your own configuration package based on this sample. Use a zip utility to unpack the extension. The following explains the individual parts of the sample configuration package.

2.7.1. The file description.xml

This files identifies the extension in LO. For your own configuration package you should choose a unique name for the identifier and a version number, eg.

```
<?xml version="1.0" encoding="UTF-8"?>
<description
xmlns="http://openoffice.org/extensions/description/2006"
xmlns:d="http://openoffice.org/extensions/description/2006">
```

```
    <identifier value="MyConfigPackage" />
    <version value="1.0" />
    </description>
```

2.7.2. The files META-INF/manifest.xml and Paths.xcu

These files should be left unchanged.

2.7.3. The folder template

Put your LO Writer template in this folder (it is recommended to use a subfolder with a descriptive name). You may add more that one templates, and if you don't want to include a Writer template you may leave it empty (do not delete the folder).

2.7.4. The folder config

Put your Writer2LaTeX/Writer2xhtml configuration in this folder. If you are using Writer2xhtml, you should also put your XHTML template here.

2.7.5. The file Options.xcu

This is the central configuration file that glues together the content of the configuration package. See the following example for an explanation of the structure.

```
    <?xml version='1.0' encoding='UTF-8'?>
    <oor:component-data oor:name="Options"
```

For LaTeX, Writer2xhtml should be replaced by Writer2LaTeX here:

```
oor:package="org.openoffice.da.Writer2xhtml"
xml:lang="en-US"
xmlns:oor="http://openoffice.org/2001/registry"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

XhtmlOptions may be replaced by XhtmlOptionsCalc or LaTeXOptions:

```
<node oor:name="XhtmlOptions">
<node oor:name="Configurations">
```

The configuration needs a unique name (you may define several configurations in the same package):

```
<node oor:name="myconfig1" oor:op="replace">
```

You can define options which are normally set in the filter dialog. In that case you can lock (disable) the corresponding parts of the dialogs. To do so, add a comma separated list of options as value here. See below for the options that can be locked for each of the three filters.

```
<prop oor:name="LockedOptions" oor:type="xs:string">
<value></value>
</prop>
```

The DisplayName is the name displayed in the style/format list in the filter dialog.

```
<prop oor:name="DisplayName" oor:type="xs:string" oor:localized="true">
<value>My Config Package</value>
</prop>
```

This path points to the configuration within the extension, you want to use:

```
    <prop oor:name="ConfigURL" oor:type="xs:string">
    <value>%origin%/config/myconfig.xml</value>
    </prop>
```

This property (XHTML only) points to the XHTML template within the extension, you want to use:

```
    <prop oor:name="TargetTemplateURL" oor:type="xs:string">
    <value>%origin%/config/mytemplate.xhtml</value>
    </prop>
```

This property (XHTML only) points to style sheet within the extension, you want to include (for EPUB export):

```
    <prop oor:name="StyleSheetURL" oor:type="xs:string">
    <value>%origin%/config/mytemplate.xhtml</value>
    </prop>
    </node>
    </node>
```

The next section defines the LO template you wish to connect with your configuration:

```
    <node oor:name="Templates">
```

The entry needs a unique name:

```
    <node oor:name="mytemplate1" oor:op="replace">
    <prop oor:name="TemplateName" oor:type="xs:string">
```

The name of the LO template is defined here (leave out .odt).

```
    <value>MyWriterTemplate</value>
```

```
</prop>
<prop oor:name="ConfigName" oor:type="xs:string">
```

The configuration to link to is defined here.

```
<value>myconfig1</value>
</prop>
</node>
</node>
</node>
</oor:component-data>
```

2.7.6. About locked options

The options you can specify for the LockedOptions property depends on the filter. The following list details which options are available to lock for each filter (see section 4).,

Writer2LaTeX

`backend`, `inputencoding`, `multilingual`, `greek_math`, `additional_symbols`[2], `use_bibtex`, `bibtex_style`, `wrap_lines_after`, `split_linked_sections`, `split_toplevel_sections`, `save_images_in_subdir`, `notes`, `metadata`, `display_hidden_text`, `original_image_size`, `simple_table_limit`, `float_tables`,

[2]This is a pseudo-option which locks all the options `use_pifont`, `use_ifsym`, `use_wasysym`, `use_eurosym` and `use_tipa`.

float_figures, float_options, ignore_hard_page_breaks, ignore_hard_line_breaks, ignore_empty_paragraphs, ignore_double_spaces

Writer2xhtml (XHTML export)

scaling, column_scaling, convert_to_px, image_size, notes, use_dublin_core, ignore_hard_line_breaks, ignore_empty_paragraphs, ignore_double_spaces, split_level, repeat_levels, save_images_in_subdir

Writer2xhtml (EPUB export)

scaling, column_scaling, relative_font_size, font_scaling, use_default_font, default_font_name, convert_to_px, image_size, ignore_hard_line_breaks, ignore_empty_paragraphs, ignore_double_spaces, display_hidden_text, notes, split_level, page_break_split, split_after, image_split, cover_image, external_toc_depth, include_toc

Calc2xhtml

scaling, column_scaling, convert_to_px, image_size, notes, use_dublin_core, display_hidden_sheets, display_hidden_rows_cols, display_filtered_rows_cols, apply_print_ranges, use_title_as_heading, use_sheetnames_as_headings, calc_split, save_images_in_subdir

3. Using the command line utility

3.1. How to install Writer2LaTeX for command line usage

Writer2LaTeX can work as a standalone command line utility (an installation of LO is not required).

Limitation: The export filters support conversion of embedded objects and graphics to a suitable format. The command line utility can only handle graphics in the original format.

3.1.1. Installation for Microsoft Windows

To install Writer2LaTeX under Microsoft Windows follow these instructions:

1. Unzip `writer2latex14.zip` into some directory. This will create a subdirectory `writer2latex14`.
2. Add this directory to your PATH environment variable (optional but recommended).

In some cases you may have to edit `w2l.bat` slightly: The batch file assumes that the java executable is in your path. To verify this, open a command prompt and type `java -version`. If this test fails (or if you have several Java versions installed and want to use a specific version): Open the file `w2l.bat` with a text editor and edit the approriate line to contain the full path to the Java executable, eg.

```
set JAVAEXE="C:\j2sdk1.7.0_67\bin\java”
```

3.1.2. Installation for Unix and friends

1. Unzip `writer2latex14.zip` into some directory. This will create a subdirectory `writer2latex14`.

2. Add this directory to your PATH environment variable (optional but recommended).
3. Add execute permissions to `w2l` as follows:

```
chmod +x w2l
```

In some cases you may have to edit the script slightly:

If you place w2l and writer2latex.jar in different directories, or if you choose to create a symbolic link to the script: Open the file `w2l` with a text editor and replace the path at the top of the file with the full path to Writer2LaTeX, eg.

```
W2LPATH="/home/username/writer2latex14"
```

Also, the script assumes that the java executable is in your path, or that the JAVA_HOME variable points to the locations. To verify the former, open a command shell and type `java -version`. To verify the latter, type `env`. If neither is the case or you have several Java versions installed you should edit this line to contain the full path to the Java executable, ie.

```
set MYJAVAEXE="/path/to/java/executable/”
```

3.2. Using the command line utility

To invoke the command line utility, use the command line

```
w2l <options> <source document/path> [<target document/path>]
```

The available options are

Group	*Option*	*Explanation*

Format	`-latex`	Convert to LaTeX (default)
	`-bibtex`	Convert to BibTeX
	`-xhtml`	Convert to XHTML 1.0 strict
	`-xhtml11`	Convert to XHTML 1.1
	`-xhtml+mathml`	Convert to XHTML + MathML
	`-xhtml+mathml+xsl`	Convert to XHTML + MathML with xsl (see section 2.4)
	`-html5`	Convert to HTML5
	`-epub`	Convert to EPUB

Config	`-config <file>`	Load configuration file (see section 4)
	`-ultraclean`	Load the LaTeX format *ultraclean*
	`-clean`	Load the LaTeX format *clean*
	`-pdfprint`	Load the LaTeX format *pdfprint*
	`-pdfscreen`	Load the LaTeX format *pdfscreen*
	`-cleanxhtml`	Load the XHTML format *cleanxhtml*
xhtml	`-template <file>`	Load an XHTML template
	`-stylesheet <file>`	Load a custom style sheet for inclusion in the document (EPUB export only)
	`-recurse`	Recurse into subdirectories (batch conversion)
Options	`-<option> <value>`	Set a configuration options (see section 4)

Some of the options are explained in more detail in the examples below.

3.2.1. Examples converting to LaTeX

The command line

```
    w2l mydocument.odt mypath/myoutputdocument.tex
```

will convert the document `mydocument.odt` in the current directory, and save the result in the subdirectory `mypath` in the document `myoutputdocument.tex`.

The command line

```
    w2l -config myconfig.xml mydocument.odt
```

will convert the document using the configuration file `myconfig.xml` (You can read more about configuration in section 4). As no output file is specified, Writer2LaTeX will use the same name as the original document, but change the extension to `.tex`.

You can also specify any simple option described in section 4 directly on the command line. Eg. to produce a file suitable for processing with pdfLaTeX:

```
    w2l -backend pdftex mydocument.odt
```

Instead of giving your own configuration file, you can use one of the standard configurations. For example to produce a clean LaTeX file (ie. ignoring most of the formatting from the source document):

```
    w2l -clean mydocument.odt
```

3.2.2. Examples converting to BibTeX from the command line

Writer2BibTeX extracts bibliography data to a BibTeX file. For example

```
    w2l -bibtex mydocument.odt
```

will extract all bibliographic references from the document and store them in a file named `mydocument.bib`. You can also extract the data as part of the conversion to LaTeX, see section 4.

3.2.3. Examples converting to XHTML from the command line

The command line

```
    w2l -xhtml+mathml mydocument.odt
```

will convert the document to XHTML+MathML, using the filename `mydocument.xhtml`.

Likewise the commandline

```
w2l -xhtml -config myconfig.xml mydocument.odt myresult.html
```

will convert into XHTML using the specified configuration and file name.

To produce a *clean* xhtml file (see section 4.3), for example:

```
w2l -cleanxhtml mydocument.odt mypath/myoutputdoc.html
```

3.2.4. Examples converting to EPUB from the command line

The command line

```
w2l -epub -split_level 2 mydocument.odt
```

will convert to EPUB, divding the document at sections of level 2

Likewise the command line

```
w2l -epub -stylesheet mystyles.css -cleanxhtml -split_level 2
```

will create an EPUB file using the custom style sheet `mystyles.css` for formatting.

4. Configuration

4.1. Writer2LaTeX configuration

LaTeX export can be configured with a configuration file. The location of the configuration depends on how you use Writer2LaTeX: Please see the sections on the export filter and the command line application.

The configuration is a file in xml format. Here is a sample configuration file for producing a document of class `book`, converting only basic formatting and optimizing for pdfTeX.

```
<?xml version="1.0" encoding="UTF-8" ?>
<config>
<option name="backend" value="pdftex" />
<option name="documentclass" value="book" />
<option name="inputencoding" value="latin1" />
<option name="use_pifont" value="false" />
<option name="use_bibtex" value="false" />
<option name="bibtex_style" value="plain" />
<option name="formatting" value="convert_basic" />
<option name="page_formatting" value="convert_all" />
<heading-map max-level="4">
<heading-level-map writer-level="1" name="chapter" level="0" />
<heading-level-map writer-level="2" name="section" level="1" />
<heading-level-map writer-level="3" name="subsection"
level="2" />
```

```
<heading-level-map writer-level="4" name="subsubsection"
level="3" />
</heading-map>
<custom-preamble />
<style-map name="Quotations" family="paragraph"
before="\begin{quote}" after=\end{quote} />
<string-replace input="LaTeX" latex-code="{\LaTeX}" />
</config>
```

Writer2LaTeX comes with five standard configuration files:

- `ultraclean.xml` to produce a *clean* LaTeX file, ie. almost all the formatting is ignored.
- `clean.xml` is a less radical version; preserves hyperlinks, color and some character formatting.
- `pdfscreen.xml` to produce a LaTeX file which is optimized for screen viewing using the package `pdfscreen.sty`.
- `pdfprint.xml` to produce a LaTeX file which is optimized for printing with pdfTeX.

In addition, you can find a sample configuration file suitable for documents originating from Google Docs in the directry `samples/config`.

The following subsections explains the available options. The options written in italics can be set using the dialog if you use Writer2LaTeX as an export filter.

4.1.1. General options

These options are used to control general aspects of the generated LaTeX document.

`documentclass`	This options defines the name of the LaTeX documentclass to use (default is `article`).
`global_options`	This option is a list of global options to add to the documentclass (the default value is an empty string).
`backend`	This option can have any of the values `generic`, `dvips`, `pdftex` (default), `xetex` and `unspecified`. This will create LaTeX files suitable for any backend/dvi driver, dvips, pdfTeX or XeTeX respectively. The last value does not assume any specific backend. This value of the option affects export of graphics: Only file types than can be handled by the backend are included. If you use the filter, other graphics will be converted to a suitable format. If you use the command line application, other types will be commented out. If you use `unspecified`, no graphics will be commented out, nor converted.

inputencoding	The option `inputencoding` can have any of the values `ascii` (default), `latin1`, `latin2`, `iso-8859-7`, `cp1250`, `cp1251`, `koi8-r` or `utf8`. This option has no effect if the backend is XeTeX, in this case the encoding is always utf-8.
multilingual	If this option is set to `true` (default), Writer2LaTeX will export all language information in the document. If `backend` is `xetex`, the package `polyglossia.sty` will be used, otherwise the package `babel.sty`. If the option is set to `false`, Writer2LaTeX will assume that the document is written in one language only. If `backend` is `xetex`, no language information will be exported, otherwise the language used for the majority of the text in the document will be exported using `babel.sty`.
greek_math	This option can have the values `true` (default) or `false`. This means that greek letters in latin or cyrillic text are rendered in math mode. This behaviour assumes that greek letters are used as symbols in this context, and has the advantage that greek text fonts are not required. It is *not* used in greek text, where it would be look awful. This option has no effect if the `backend` is set to `xetex`.

use_pifont	Setting this option to `true` enables the use of *Zapf Dingbats* using the LaTeX package `pifont.sty`. Default is `false`. This option and the following five font options has no effect if the backend is XeTeX.
use_ifsym	Setting this option to `true` enables the use of the *ifsym* symbol font using the LaTeX package `ifsym.sty`. Default is `false`.
use_wasysym	Setting this option to `true` enables the use of the wasy symbol font using the LaTeX package `wasysym.sty`. Default is `false`.
use_bbding	Setting this option to `true` enables the use of the *bbding* symbol font (a clone of Zapf Dingbats) using the LaTeX package `bbding.sty`. Default is `false`.
use_eurosym	Setting this option to `true` enables the use of the `eurosym` font using the LaTeX package `eurosym.sty`. Default is `false`.
use_tipa	Setting this option to `true` enables the use of phonetic symbols using the LaTeX packages `tipa.sty` and `tipx.sty`. Default is `false`.
use_ooomath	This option can have the values `true` or `false` (default). This enables the use of the LaTeX package ooo`math.sty` (see section 6). This package defines number of LaTeX macros used to convert formulas from LO to LaTeX. If this package is not used, the necessary definitions will be included in the LaTeX preamble, which may become quite long – so using ooo`math.sty` is recommended for documents with formulas.

use_lastpage	This option can have the values true or false (default). This enables use of the package lastpage.sty to represent the page count.

4.1.2. Options for bibliography (BibTeX)

These options controls the handling of the bibliography.

use_bibtex	Setting this option to true enables the use of BibTeX for bibliography generation. If it is set to false (default), the bibliography is included as static text.
bibtex_style	This option can have any BibTeX style as value (default is plain). This is the BibTeX style to be used in the LaTeX document.
use_natbib	Setting this option to true loads the LaTeX package natbib.sty.
natbib_options	Use this option to provide options to natbib.sty.
external_bibtex_files	Use this option to give a list of external BibTeX files. If the list is non-empty, bibliographic references in the document will be interpreted as keys in these files. If it is empty (default), the bibliographic references will be exported to a BibTeX file (provided use_bibtex is set to true).

`zotero_bibtex_files`	Use this option to give a list of external BibTeX files. If the list is non-empty and `use_bibtex` is set to `true`, Zotero references in the document will be interpreted as keys in these files. Also the Zotero bibliography, if any, will be exported as a LaTeX bibliography. This will take advantage of the LaTeX package `natbib.sty`. If `use_natbib` is set to `true`. Otherwise (default), Zotero references and bibliography will be exported as text.
`jabref_bibtex_files`	Use this option to give a list of external BibTeX files. If the list is non-empty and `use_bibtex` is set to `true`, JabRef references in the document will be interpreted as keys in these files. Also the JabRef bibliography, if any, will be exported as a LaTeX bibliography. This will take advantage of the LaTeX package `natbib.sty`. If `use_natbib` is set to `true`. Otherwise (default), JabRef references and bibliography will be exported as text.

`include_original_citations`	If you convert Zotero or JabRef references, you can set this option to `true` (default is `false`) to include the original citation inserted by Zotero/JabRef as a comment in the LaTeX source.

4.1.3. File options

These options controls the creation of files associated with the main LaTeX document.

`wrap_lines_after`	The option specifies that Writer2LaTeX should try to break lines in the LaTeX source as soon as possible after this number of characters. Default is `72`. If you use a text editor which supports wrapping of long lines, you may want to set this option to `0`: In this case Writer2LaTeX will not wrap lines.
`split_linked_sections`	This option specifies that a linked section should be exported to a separate LaTeX-file. Default is `false`.
`split_toplevel_sections`	This option specifies that all sections should be exported to a separate LaTeX-file, excluding nested sections. Default is `false`.
`save_images_in_subdir`	Images contained in the document are normally placed in the same directory as the LaTeX document. If the document contains a large number of images, it may be more convenient to put the images in a subdirectory. Set this option to `true` to do this.

4.1.4. Options for special content

notes	This option can have any of the values comment (default), ignore, marginpar, pdfannotation. This specifies what to do with notes (annotations) in the document: They can be ignored, converted to LaTeX comments, converted to \marginpar or converted to pdf annotations (which will default to \marginpar if the document is not processed with pdfLaTeX). In addition, you can give any LaTeX command (inluding the backslash), and the notes will be exported as \yourcommand{the note}.
metadata	If you set his option to true (default), Writer2LaTeX will export the title, author and date of the document as found under **File – Properties**. Furthermore, if you have chosen pdf as the backend, the title, author, subject and keywords will be exported to the pdf document and will be viewable if the pdf viewer supports it. If the option is false , only the title will be exported.

4.1.5. Figure and table options

The first options are used to control the handling og floating or non-floating figures and tables.

float_figures	Use this option to specify that you want to include graphics and text boxes in a floating figure environment. Default is false. This option has no effect on graphics and text boxes that are anchored *as character*. These are always considered to be part of the normal text flow. If you want a figure to float, anchor it *to paragraph* or *to character*.

float_tables	Use this option to specify that you want to include tables in a floating `table` environment. Default is `false`.
float_options	Use this to give placement options to the figure and table floats, eg. `h` for *here*. Default is empty (default placement).
`align_frames`	Use this option to specify, that all graphics and text boxes should be centered. If you don't want that, set this option to `false`. Default is `true`.
`use_caption`	Use this option if you want to take advantage of the LaTeX package `caption.sty`. Currently Writer2LaTeX only uses the support for non-floating captions from this package.
`figure_sequence_name`	This option can be set to a sequence name in the source document. OpenDocument has a very weak sense of figure captions: A figure caption is a paragraph containing a sequence number. If you use LO's defaults, Writer2LaTeX can guess which sequence name to use. If it fails, you can give the name in this option (default is empty).
`table_sequence_name`	This is a similar option for tables.

These options controls the export of tables:

simple_table_limit	You can set this option to any non-negative integer (default is 0). Table cells in LO can contain any number of paragraphs, so normally Writer2LaTeX exports tables with p columns. For simple tables where all cells only contains a single line it is better to use l, c and r columns. If all cells in a table contains at most one paragraph, and the *total* width of the table is less than this number of characters, the table will be exported with l, c and r columns. This option has no effect on tables using tabulary.
use_longtable	This option is used to specify that longtable.sty should be used to export tables which may break across pages. Default is false.
use_supertabular	This option is used to specify that supertabular.sty should be used to export tables which may break across pages. Default is true. (You should only set one of the options use_longtable and use_supertabular to true).
use_tabulary	This option is used to specify that tabulary.sty should be used to export tables. Default is false.
use_colortbl	This option is used, if you want to apply background color to tables using the package colortbl.sty. The value can be true or false (default). This option has no effect unless you also set the option use_color to true.

`table_first_head_style`	This option is used to produce advanced tables, that are not supported in Writer. You can set this option to the name of a paragraph style. If the first paragraph of the first cell in a row is formatted with this paragraph style, the row in question will be used for the first head in a multipage table.
`table_head_style`	Likewise this option specifies a paragraph style that identifies a repeating head in a multipage table (like a normal table head in Writer).
`table_foot_style`	This option specifies a paragraph style that identifies a repeating foot in a multipage table.
`table_last_foot_style`	This option specifies a paragraph style that identifies the last foot in a multipage table.

These options controls the export of figures:

`original_image_size`	Often images in a Writer document are scaled up or down from their original size. Normally the same scaling will be used in the LaTeX document, but if you set this option[3] to `true`, the original (unscaled) image size will be used. The default value is `false`.

`remove_graphics_extension`	This option can be used to specify, that the file extension on graphics files should be removed. You will thus get eg. `\includegraphics{myimage}` rather than `\includegraphics{myimage.png}`.
`image_options`	This option can be used to specify some options that should be applied to all images (ie. all `\includegraphics` commands). For example `"width=\linewidth"`. Default is empty (no options).

4.1.6. AutoCorrect options

`ignore_hard_page_breaks`	This option can have the values `true` or `false` (default). Setting the option to `true` will instruct Writer2LaTeX to ignore hard page breaks (but not soft page breaks specified in paragraph styles).
`ignore_hard_line_breaks`	This option can have the values `true` or `false` (default). Setting the option to `true` will instruct Writer2LaTeX to ignore hard line breaks (shift-Enter).
`ignore_empty_paragraphs`	This option can have the values `true` (default) or `false`. Setting the option to `true` will instruct Writer2LaTeX to ignore empty paragraphs; otherwise they are converted to a `\bigskip`.

[3] In previous versions, this option was called `keep_image_size`, but has been renamed to avoid confusion (the old name is still supported).

ignore_double_spaces	This option can have the values `true` (default) or `false`. Setting the option to `true` will instruct Writer2LaTeX to ignore double spaces, otherwise they are converted to `\ `.

.0cmFormatting options

In Writer, formatting is controlled by styles. You can control how much formatting is exported using the following options. Note that these options has a major impact on the structure of the LaTeX document created.

`formatting`	The option `formatting` can have any of these values: ● `ignore_all` will instruct Writer2LaTeX to ignore *all* character, paragraph, heading, list and footnote formatting contained in the document. ● `ignore_most` will preserve basic character formatting. ● `convert_basic` (default) will preserve basic character formatting, paragraph justification and all numberings (lists, headings, footnotes). ● `convert_most` will convert all supported formatting, except that paragraph formatting and font size is only converted if it is set by a style. To be able to preserve formatting, an environment is created for all paragraph styles, custom lists are used for listings, headings are reformatted using the `\@startsection` command etc. ● `convert_all` will preserve *all* supported formatting.
`page_formatting`	This option can have any of the values `ignore_all`,`convert_header_footer`,`convert_all` This will ignore all page formatting, convert the header and footer (using custom page styles) or convert all supported formatting, including page geometry and footnote rule.

use_geometry	Setting this option to true specifies that the package geometry.sty should be used to export the geometry of the page (page size, margins etc.). Default is false, which will export the geometry using the low level LaTeX commands.
use_fancyhdr	Setting this option to true specifies that the package fancyhdr.sty should be used to export the header and footer of the page. Default is false, which will export the header and footer using the low level LaTeX page style commands.
use_color	This option can have the values true (default) or false. This enables use of the package color.sty to apply color in the LaTeX document.
use_ulem	This option can have the values true or false (default). This enables use of the package ulem.sty to support underlining and crossing out in the LaTeX document.
use_hyperref	This option can have the values true (default) or false. This enables use of the package hyperref.sty to include hyperlinks in the LaTeX document.
tabstop	This option is used to specify what to do with tabulator stops in the document. Normally these are converted to spaces, but with this option you can specify any LaTeX code, that should be used instead. For example "\quad{}" or "\hspace{2em}"

use_endnotes	This option can have the values true or false (default). This enables use of the package endnotes.sty to format the endnotes in the LaTeX document. If set to false, endnotes will be converted to footnotes.

4.1.7. Options for including or excluding content

The following options can be used to control which content to export.

`no_preamble`	If this option is set to `true`, (default is `false`), Writer2LaTeX will not create the a LaTeX preamble, nor include `\begin{document}` and `\end{document}`. This is useful if the document is to be included in another LaTeX document. Note that in this case you will have to make sure that all packages/definitions needed are available in the master LaTeX document.
`no_index`	If this option is set to `true`, (default is `false`), Writer2LaTeX will not export indexes (e.g. table of contents, bibliopgrahy). This option is also intended for the case that the document is to be part of a larger LaTeX document, which may contain global indexes.
`other_styles`	This option can all have the values `accept` (default), `ignore`, `warning` and `error`. This controls how to export paragraph and text content, for which there is no style map (see below). If the value of this option is `accept`, the content is handled as normal. If the value is `ignore`, the content is ignored silently. The values `warning` and `error` issues a message on the terminal resp. in the generated LaTeX code. This option thus lets you control that only content with accepted styles is exported.
`image_content`	This option has the same values, and is used to exclude image content.
`table_content`	This option also has the same values and is used to exclude table content.
`display_hidden_text`	If this option is set to `true` (default is `false`), paragraphs and text portions marked as hidden will be exported. Otherwise they will be ignored.

4.1.8. Headings

The `heading_map` section specifies how headings in LO should map to LaTeX. Eg. the first line in the sample above specifies that the toplevel heading (**Heading 1**) should map to `\chapter`, which is of level 0 in LaTeX. Up to 10 levels are supported (the same number as in LO).

4.1.9. Style maps

In addition you can specify maps from styles in Writer to your own LaTeX styles in the configuration. Currently this is possible for text styles, paragraph styles and list styles. In addition a few direct formatting attributes can be mapped to LaTeX code. The following examples are from the standard configuration file `article.xml`.

This is a simple rule, that maps text formatted with the text style **Emphasis** to the LaTeX code `\emph{...}`:

```
<style-map name="Emphasis" family="text" before="\emph{" after="}" />
```

This is another simple rule, that maps paragraphs formatted with the paragraph style **part** to the LaTeX code `\part{...}`. The attribute `line-break` ensures that no line breaks are inserted between the code and the text.

```
<style-map name="part" family="paragraph" before="\part{" after="}"
line-break="false" />
```

This is a rule, that maps paragraphs formatted with style **Preformatted Text** to the LaTeX environment `verbatim`. The attribute `verbatim` ensures that the content of the paragraph is exported verbatim (this implies that characters not available in the `inputenc` are converted to question marks and that other content is discarded, eg. footnotes). The `paragraph-block` entry specifies code to go before and after an entire block of paragraphs. The `name` attribute specifies the style of the first paragraph; the `next` attribute specifies the style(s) of subsequent paragraphs in the block.

```
<style-map name="Preformatted Text" family="paragraph-block"
next="Preformatted Text" before="\begin{verbatim}"
after="\end{verbatim}" />
```

```
<style-map name="Preformatted Text" family="paragraph" before=""
after="" verbatim="true" />
```

This is a more elaborate set of rules, that maps paragraphs formatted with styles **Title**, **author** and **date** (in any order) to `\maketitle` in LaTeX.

```
<style-map name="Title" family="paragraph" before="\title{" after="}"
line-break="false" />
<style-map name="author" family="paragraph" before="\author{" after="}"
line-break="false" />
<style-map name="date" family="paragraph" before="\date{" after="}"
line-break="false" />
<style-map name="Title" family="paragraph-block" next="author;date"
before="" after="\maketitle" />
<style-map name="author" family="paragraph-block" next="Title;date"
before="" after="\maketitle" />
<style-map name="date" family="paragraph-block" next="Title;author"
before="" after="\maketitle" />
```

This will produce code like this:

```
\title{Configuration}
```

```
\author{Henrik Just}
\date{2006}
\maketitle
```

The next example maps a paragraph formatted with the **theorem** list style to a LaTeX environment named `theorem`. Note that there are two entries for a list style: The first one to specify the LaTeX code to put before and after the entire list. The second one to specify the LaTeX code to put before and after each list item.

```
<style-map name="theorem" family="paragraph" before="" after="" />
<style-map name="theorem" family="list" before="" after="" />
<style-map name="theorem" family="listitem" before="\begin{theorem}"
after="\end{theorem}" />
```

When you override a style, all formatting specified in the original document will be igored.

Finally an example using direct formatting attributes:

```
<style-map name="italic" family="text-attribute" before="\emph{"
after="}" />
```

Currently the only supported names are `italic`, `bold`, `small-caps`, `superscript` and `subscript`.

4.1.10. String replace

Often LaTeX requires special care to typeset certain constructions. For example according to German typographical rules, an abbreviation like z.B. should be typeset with a small space before the B. You can specify this in the configuration:

```
<string-replace input="z.B." latex-code="z.\,B." />
```

The `input` is the text in the LO document, the `latex-code` is the LaTeX code to export for this text.

Another example is French quotations marks (« Je parle français ») which should be converted to the LaTeX macros \fg and \og. This can be achieved using this rule:

```
    <string-replace input="&#xAB;&#xA0;" latex-code="\fg " />
    <string-replace input="&#xA0;&#xBB;" latex-code="\og " />
```

The final example ensures that the LaTeX logo is typeset correctly

```
    <string-replace input="LaTeX" latex-code="{\LaTeX}" />
```

4.1.11. Math symbols

In LO Math you can add user-defined symbols. Writer2LaTeX already understands the predefined symbols such as `%alpha`. If you define your own symbols, you can add an entry in the configuration that specifies LaTeX code to use. The `math-symbol-map` element is used for this:

```
    <math-symbol-map name="ddarrow" latex="\Downarrow" />
```

This example will map the symbol `%ddarrow` to the LaTeX code `\Downarrow`.

4.1.12. Custom preamble

The text you specify in the element `custom-preamble` will be copied verbatim into the LaTeX preamble. For example:

```
    <custom-preamble>\usepackage{palatino}</custom-preamble>
```

to typeset your document using the postscript font palatino.

4.2. Writer2xhtml and Calc2xhtml configuration

Also the XHTML export can be configured with a configuration file in xml format. This is a sample configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
<option name="custom_stylesheet" value="/mystyle.css" />
<option name="ignore_styles" value="false" />
<option name="use_dublin_core" value="true" />
<option name="convert_to_px" value="true" />
<option name="split_level" value="1" />
<xhtml-style-map name="mystyle" family="paragraph" element="p"
css="mycssclass" />
</config>
```

The following subsections explains the available options. The options written in italics can be set using the dialog if you use Writer2xhtml as an export filter.

4.2.1. Style options

You can control some general aspects of the generated XHTML documents using these technical options.

template_ids	This option is used to specifiy the id's used for XHTML templates. These should be provided as a comma separated list defining the id for content,header,footer and panel in that order. The list can be truncated if you don't need them all. The default is empty, which is equivalent to content,header,footer,panel.
pretty_print	Set this option to false (default is true) if you don't want "pretty print" (using indentations and line breaks) in the XHTML output.
no_doctype	If you set this options to true (default is false), Writer2xhtml will not include the !DOCTYPE declaration in the converted document. The !DOCTYPE is required for a valid XHTML document: This option should only be used if you need to process the document further.
encoding	This option is used to specify the character encoding to use for the XHTML document. Currently supported encodings are UTF-8 (default), UTF-16, ISO-8859-1 and US-ASCII. Characters not supported by the encoding are exported as numeric character entities.
hexadecimal_entities	When this option is set to true (default) numeric character entities are exported using hexadecimal numbers, otherwise decimal numbers are used

`use_named_entities`	If you set this options to `true` (default is `false`), Writer2xhtml will use named character entities as defined by (X)HTML. If you export to XHTML+MathML, also named MathML entities will be used.
`add_bom`	In rare cases, it may be required to ad a BOM (Byte Order Mark) to the XHTML document. Most applications will not need this, but you can set this options to `true` to enable this (default is `false`).
`multilingual`	Set this to `false` (default is `true`) to remove language information from the file (except on the root element)
`separate_stylesheet`	Set this to true (default is false) to generate a separate CSS file if the XHTML document is split over several files (thus avoiding repeating the style information in every file).
`custom_stylesheet`	Use this option to give an URL to your own, external CSS stylesheet. If the value is empty or the option is not specified, no external stylesheet will be used. For more advanced solutions (eg. different style sheets for screen viewing and printing) you can use an XHTML template – see below.

The following options are used to control the conversion of the formatting in the source document. If you use an external CSS style sheet, this is important to define.

`formatting`	The option `formatting` is used to specify how much text formatting (character, paragraph and list formatting) to export[4]. Possible values are `convert_all` (default): Convert all formatting to CSS. `ignore_styles`: Convert hard formatting but not formatting by styles. Use this value if you use a custom stylesheet, but still want to be able to add some hard formatting (eg. a centered paragraph, some bold text etc.) `ignore_hard`: Convert formatting by styles, but no hard formatting (except as given by attribute style maps, see below). Use this if the document is well structured using styles, so that any hard formatting should be considered an error. `ignore_all`: Convert no formatting at all. Use this value if you use a custom stylesheet *and* the document is well structured using styles, so that any hard formatting should be considered an error.

`frame_formatting`	Used for the same purpose for frame formatting.
`section_formatting`	Used for the same purpose for section formatting. (But note that LO does not offer section styles currently).
`table_formatting`	Used for the same purpose for table formatting. (But note that LO does not offer table styles currently).
`table_size`[5]	This option defines how to export table dimension. The possible values are: `none`: Do not export table dimensions (table width, column width and row height), leaving the layout of the tables to the browser. `auto` (default): Convert the dimensions in the source document using relative or absolute values as defined. `relative`: Convert the dimensions in the source document, but always using relative values for table width and column width.

list_formatting	This option determines how list formatting is exported. Possible values are[6]: css1: List formatting is exported using CSS1. This only provides basic support for list labels, and currently the browsers default indentations are used. css1_hack: This value is used to fix a problem with continued lists. Writer2xhtml will export a list that continues on level 2 or below like `<ol><ol><li>...</li></ol></ol>` This is *not* valid in XHTML, but works in browsers. Also two deprecated attributes are used to continue numbering. hard_labels: If you use this value, list labels are exported as part of the text. This adds full support for list labels (e.g. labels of the form 1.2.3). Unlike the other values indentations of the list are exported as well.

tabstop_style	Used this option to specify a style used for tabstops. Normally tabstops are exported as spaces, but with this option the space will be contained in a `span` element, eg. `<span class="tabstop"> </span>` You can then define a CSS rule like eg. `tabstop { width: 2em; }`
use_default_font	Set this option to `true` (default is `false`) to ignore all font information in the document and use a default font for the entire exported document.
default_font_name	Use this option to supply a font name to use if the option `use_default_font` is set to `true`. A blank value will not insert ant font information.

In addition, a number of options defines how dimensions in the source document should be handled.

convert_to_px	When this option is `true` (default), Writer2xhtml will convert all units to px, otherwise the original units are used. The resolution is assumed to be 96ppi, you can change this with the `scaling` option. Eg. a scaling of `75%` will change the resolution to 72ppi. For EPUB export this option will export font sizes as percentages (and use px for other dimensions).

[4] This and the following options replaces the former option `ignore_styles`.

[5] This option replaces the former option `ignore_table_dimensions`. (The former values correspond to the values `none` and `auto`).

[6] In previous versions, this option was called `list_hack`, but was renamed to support the new value `hard_labels`. (The old name is still supported.)

scaling	Use this option to specify a scaling of all formatting, ie. to get a different text size than the original document. The value must be a percentage, default is `100%`.
column_scaling	Use this option to specify an additional scaling for table colums. The value must be a percentage, default is `100%`.
image_size[7]	Use this option to specify how to export the size of images and text boxes: Possible values are `absolute` or `auto` (default, export absolute size), `relative` (export the size as a percentage of the current text width) and `none` or `original_image_size` (do not export size information; hence the browser or reader will use the original (unscaled) image size).
`relative_font_size`	Set this option to `true` (default) `false` to export all font sizes as percentages rather than using absolute dimensions. The font size is calculated relative to the default font size in the document.
`font_scaling`	Use this option to specify a scaling for all font sizes if `relative_font_size` is set to `true`. Default is `100%`.

[7] This option replaces old options `keep_image_size` and `original_image_size` (the old names are still supported).

4.2.2. Options for special content

`formulas`	If you are not exporting to XHTML+MathML or HTML5, this option defines how formulas are treated. The possible values are `starmath` (default) to export the formula in StarMath notation, `latex` to export the formula in LaTeX notation, `image+starmath` and `image+latex` to export the formula as an image, with an `alt` attribute giving the formula in StarMath or LaTeX notation.
`use_mathjax`	If you export to XHTML+MathML or HTML5, you can set this option to `true` (default is `false`) to load the MathJax library on pages with fomulas. This will ensure that formulas are viewable on all modern browsers, even if they do not support MathML natively[8].
`embed_svg`	If you export to HTML5, you can set this option to `true` to export vector graphics embedded in the HTML documents as SVG (scalable vector graphics). If set to `false` (default), external SVG image files will be used.
`endnotes_heading`	In LO the endnotes are set on a separate page at the end of the document. It is not possible to give this page a heading, but you can use this option to add a heading. In EPUB export this heading will also appear in the navigation table. Default is empty (no heading).

footnotes_heading	In LO the footnotes may be set on a separate page at the end of the document (if configured to do so). It is not possible to give this page a heading, but you can use this option to add a heading. In EPUB export this heading will also appear in the navigation table. Default is empty (no heading).
use_dublin_core	Use this option to specify if Dublin Core Meta data should be exported. For the XHTML export, the format will be as specified in http://dublincore.org/documents/dcq-html/). For the EPUB export this option has no effect, cf. section Fejl: Henvisningskilde ikke fundet for EPUB meta data. If the value is `false`, it will not be exported (default is `true`).
notes	If this option is set to `true` (default), notes in the document will be exported as XHTML comments. These are not directly visible in the browser. If you don't want to include notes, set this option to `false`.
display_hidden_text	If this option is set to `true` (default is `false`), paragraphs and text portions marked as hidden will be exported. Otherwise they will be ignored.

`incluce_toc`	If this option is set to `true` (default), the table of contents is exported. If it is set to `false`, the table of contents is ignored. The latters possibility is mainly intended for EPUB, which also provides an external navigation table.

4.2.3. AutoCorrect options

`ignore_double_spaces`	This options can have the values `true` (default) or `false`. Setting the option to `true` will instruct Writer2xhtml to ignore double spaces, otherwise they are converted to non-breaking spaces.
`ignore_empty_paragraphs`	This option can have the values `true` (default) or `false`. Setting the option to `true` will instruct Writer2xhtml to ignore empty paragraphs..
`ignore_hard_line_breaks`	This option can have the values `true` or `false` (default). Setting the option to `true` will instruct Writer2xhtml to ignore hard line breaks (Shift-Enter in LO).

[8]This replaces the output format (available until Writer2LaTeX 1.2) using an XSLT style sheet for the same purpose.

4.2.4. File options

external_toc_depth	In addition to the text content, an EPUB document contains a table of contents, which can be used for navigation in the reader. This table is generated by Writer2xhtml from the headings in your document. This option is used to specify the number of levels to include in the table. The default value is `auto`, which determines the depth from the option `split_value`. If you want to set the depth independent from `split_value`, set this option to a positive integer.
split_level	This option is used to specify that the Writer documents should be split in several documents and the outline level at which the splitting should happen (the default `0` means no split). This is convenient for long documents. Each output document will get a simple navigation panel in the header and the footer (with labels in the same language as the document).
repeat_levels	If you split the document, you can use this option to specify that headings of higher levels should be repeated on page breaks. This may help the user to identify the current position in the document. Default is `5` (all levels are repeated).

page_break_split	An alternative method to split the document is to use the original page breaks. Possible values are none (default): Do not split at page breaks. styles: Split at page breaks which are defined in styles. explicit: Split at all explicit page breaks (page breaks defined in styles and manual page breaks) all: Split at all page breaks. Automatic page breaks may occur within a paragraph, list or table, but Writer2xhtml will not split until this structure has ended. Also in this case, each output document will get a simple navigation panel in the header and the footer.
split_after	This option (which only has effect for EPUB export) is used to automatic split long documents. When a single file exceeds the number of characters defined by this option (in 1000s), the document will be split at the first possible break point. The value 0 disables automatic split.
image_split	This option (which only has effect for EPUB export) is used to convert large images to "full screen" images. The value of the option can be either none or a percentage. If set to a percentage, an image which is wider than this percentage and has an aspect ratio of at least 3:4 is placed in a separate file.

`cover_image`	If you set this option to `true` (default is `false`), the first image in the document is used as cover image in EPUB export.
`save_images_in_subdir`	Images contained in the document are normally placed in the same directory as the XHTML document. If the document contains a large number of images, it may be more convenient to put the images in a subdirectory. Set this option to `true` to do this.
`uplink`	This option is used to specify a link which brings the user up in a page hierarchy. For example `"../index.html"`.

4.2.5. Options specific for spreadsheet documents

`calc_split`	Set this option to `true` if you want spreadsheet documents should be split in several documents (one for each sheet). This is convenient for large spreadsheets. Each output document will get a simple navigation panel in the header and the footer. The default value is `false`, which means that the entire spreadsheet will be converted to a singe XHTML document.

display_hidden_sheets	Set this option to `true` if you want to export sheets that are defined as hidden. Default is `false`.
display_hidden_rows_cols	Set this option to `true` if you want to export rows or columns that are defined as hidden. Default is `false`.
display_filtered_rows_cols	Set this option to `true` if you want to export rows or columns that are not visible due to a filter. Default is `false`.
apply_print_ranges	I you set this option to `true`, the print ranges defined in the document will be used. The content of the result will thus be identical to the content of printed output. If you set the option to `false` (default), the content of the output will be identical to the content that you can see when editing the document.
use_title_as_heading	If you set this option to `true` (default), the title of the document will be included in the XHTML document as a heading.
use_sheet_names_as_headings	If you set this option to `true` (default), the sheet name will be added as a heading above each table in the XHTML document.

4.2.6. Options for batch conversion

`directory_icon`	Used to specify an URL for an (icon) image that represents a directory. This is used when Writer2xhtml creates index pages for a directory.
`document_icon`	Used to specify an URL for an (icon) image that represents a document. This is used when Writer2xhtml creates index pages for a directory.

4.2.7. Style maps

In addition to the options, you can specify that certain styles in Writer should be mapped to specific XHTML elements and CSS style classes. Here are some examples showing how to use some of the built-in Writer styles to create XHTML elements:

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
<!-- map LO paragraph styles to xhtml elements -->
<xhtml-style-map name="Text body" family="paragraph"
element="p" css="(none)" />
<xhtml-style-map name="Sender" family="paragraph"
element="address" css="(none)" />
<xhtml-style-map name="Quotations" family="paragraph"
block-element="blockquote" block-css="(none)"
element="p" css="(none)" />

<!-- map LO text styles to xhtml elements -->
```

```
<xhtml-style-map name="Citation" family="text"
element="cite" css="(none)" />
<xhtml-style-map name="Emphasis" family="text"
element="em" css="(none)" />

<!-- map hard formatting attributes to xhtml elements -->
<xhtml-style-map name="bold" family="attribute"
element="b" css="(none)" />
<xhtml-style-map name="italics" family="attribute"
element="i" css="(none)" />
</config>
```

An extended version of this is distributed with Writer2LaTeX, please see the file `cleanxhtml.xml`.

The attributes of the `xhtml-style-map` element are used as follows:

- `name` specifies the name of the Writer style.
- `family`[9] specifies the style family in Writer; this can either be `text`, `paragraph`, `heading`, `frame`, `list` or `attribute`. The last value does not specify a real style, but refers to hard formatting attributes. The possible names in this case are `bold`, `italics`, `fixed` (for fixed pitch fonts), `superscript`, `subscript`, `underline` and `overstrike`.
- `element` specifies the XHTML element to use when converting this style. This is not used for frame and list styles.

[9]Previously this attribute was called class.

- `css` specifies the CSS style class to use when converting this style. If it is not specified or the value is "`(none)`", no CSS class will be used.
- `block-element` only has effect for paragraph and heading styles. For paragraphs it is used to specify a block XHTML element, that should surround several exported paragraphs with this style. For headings it is used to specify the element containing the entire heading (the `element` is used for the text content only, excluding the label).
- `block-css` specifies the CSS style class to be used for this block element. If it is not specified or the value is "`(none)`", no CSS class will be used.

For example the rules above produces code like this:

```
<p>This paragraph is Text body</p>
<address>This paragraph is Sender</address>
<blockquote>
<p>This paragraph is Quotations</p>
<p>This paragraph is also Quotations</p>
</blockquote>
<p>This paragraph is also Text body and has some <em>text with emphasis
style</em> and uses some <b>hard formatting</b>.</p>
```

You can use your own Writer styles together with your own CSS style sheet to create further style mappings, for example:

```
<xhtml-style-map name="Some LO style" family="paragraph"
block-element="div" block-css="block_style"
element="p" css="par_style" />
```

to produce output like this:

```
<div class=”block_style”>
<p class=”par_style”>Paragraph with Some LO style</p>
<p class=”par_style”>Yet another</p>
</div>
```

Note that the rules for hard formatting are only used when `formatting` is set to `ignore_hard` or `ignore_all`. It is not recommended to rely on these rules, using real text styles is preferable. They are included because the use of hard character formatting is very common even in otherwise well-structured documents.

XHTML templates

You can use your own XHTML document as a template for the generated XHTML documents. This should be an ordinary XHTML file (do not include DOCTYPE declaration) with some special elements:

- An element with the id `content` is used to fill the text content. If no such element exists, the `<body>` element is used. If there is no `<body>` element in the template, the root element is used.
- Elements with the id `header` or `footer` (optional) will be filled with a simple navigation panel using a first/previous/next/last scheme (for spreadsheet documents, sheet names are used for navigation).
- An element with the id `panel` (optional) will be filled with a simple navigation panel using a table of contens-like scheme.

You can change the names of the id attributes using the `template_ids` option.

A simple template including a header might look like this:

```
<html>
<head>
<title/>
</head>
<body>
<div id='header' />
<div id='content' />
</body>
</html>
```

As the template does not include footer and panel nodes, these elements will not be included.

A template with all the elements, suitable for HTML5 might look like this:

```
<html>
<head>
<title/>
</head>
<body>
<header><nav id='header' /></header>
<aside><nav id='panel' /></aside>
<div id='content' />
<footer><nav id='footer' /></footer>
```

```
</body>
</html>
```

The absolute mininal template is this:

```
<div/>
```

The `div`-element will be used as the content container. The generated document will not be a complete XHTML document (no `<html>`, `<head>` and `<body>` nodes). It will however still be a well-formed XML file that can be handled with standard tools. The use case for this is that you can produce XHTML fragments suitable for inclusion in e.g. a CMS.

Note: Make sure to set the option `no_doctype` to `true` in this case!

4.3. Using LibreOffice to create XHTML documents

The configuration file `cleanxhtml.xml` that is distributed with Writer2LaTeX, can be used to create semantically rich XHTML content, which can be formatted with your own stylesheet (you should edit the file to add the URL to the stylesheet you want to use).

A subset of the built-in styles in Writer are mapped to XHTML elements (note that the style names are localized, so this is for the english version of LibreOffice):

LO Writer style	*LO Writer style family*	*XHTML element*
Text body	paragraph style	`p`
Sender	paragraph style	`address`

LO Writer style	*LO Writer style family*	*XHTML element*
Quotations	paragraph style	`blockquote`
Preformatted Text	paragraph style	`pre`
List Heading	paragraph style	`dt` (in `dl`)
List Contents	paragraph style	`dd` (in `dl`)
Horizontal Rule	paragraph style	`hr`
Citation	text style	`cite`
Definition	text style	`dfn`
Emphasis	text style	`em`
Example	text style	`samp`
Source Text	text style	`code`
Strong Emphasis	text style	`strong`
Teletype	text style	`tt`
User entry	text style	`kbd`
Variable	text style	`var`
bold	hard formatting attribute	`b`
italics	hard formatting attribute	`i`

LO Writer style	*LO Writer style family*	*XHTML element*
fixed pitch font	hard formatting attribute	`tt`
superscript	hard formatting attribute	`sup`
subscript	hard formatting attribute	`sub`

So by using these styles only, you will create well-structured XHTML documents. See the document `sample-xhtml.sxw` for an example of how to use this.

4.3.1. Links

LO does not support all kind of XHTML link attributes, for example you cannot set `title` or `rel`. Writer2xhtml provides a solution for thus using the `name` attribute: You can define values for all attributes by providing a semicolon separated list of names and values, eg.

```
title=My title;rel=next
```

will create an XHTML link like

```
<a href="..." title="My title" rel="next">
```

If the name attributes does not contain such a list, the value is used for the name and title attribute:

```
My name
```

will create an XHTML link like

```
<a href="..." name="My name" title="My name">
```

5. Special features for the EPUB export

5.1. Meta data

Writer2xhtml always exports the title of the document, and also the subject, keywords and description if they are non-empty.

The EPUB standard specifies a number of meta data elements not supported by ODF. Writer2xhtml supports these elements using user-defined meta data. To add user-defined meta data choose **File-Properties**, **User-defined properties**. (The export filter includes a custom editor for these.) The following properties are supported:

- **Identifier**: Each EPUB document must have a unique ID. Normally Writer2xhtml generates a Universal Unique ID (UUID) for this purpose, but you may override this with your own ID.

  To do this add a new property, enter **Identifier** (case is not important) as name and the ID as value.

  An identifier may follow a specific identification scheme, e.g. ISBN. To specify an identification scheme, append this to the name separated by a period, e.g. **Identifier.ISBN**.

  It is possible to have several identifiers, in this case append a number to the name, e.g. **Identifier1.ISBN** and **Identifier2**. The first identifier is used as the unique ID.

- **Creator**: A primary creator or author of the publication.

  Enter **Creator** as name and the creator's name as value.

  A creator may have a special role, you can specify this with a three letter code after the word **creator**, e.g. **creator.aut** for the author or **creator.ill** for the illustrator. For the complete list of three letter codes see the EPUB specification (http://www.idpf.org/2007/opf/OPF_2.0_final_spec.html).

You can define several creators, in this case add a number to the word **creator**, e.g. **creator1.aut** and **creator2**. The creators will be sorted according to the numbers. Note that some readers may only present the first creator.

If no creator is defined, Writer2xhtml will export the default creator given in the document (this is usually taken from LO's user settings).

- **Contributor**: A party whose contribution to the publication is secondary to those named in creator elements. Otherwise it is handled like Creator, and the same rules apply.
- **Date**: Date of publication. The date must be in the format YYYY-MM-DD (year-month-date) or more generally in the format specified in http://www.w3.org/TR/NOTE-datetime.

  A date may be associated with a special event such as **creation**, **publication** or **modification**. To define this, add the event after the word **date**, e.g. **date.publication**.

  You can give several dates, in this case add a number to the word date, e.g. **date1.creation**, **date2.modification**.

  If you don't define any dates, Writer2xhtml will include the date the document was last modified.

You can only have one instance of the remaining properties, hence they cannot be numbered. Also no additional data can be appended to the name.

- **Publisher**: The publisher of the document.
- **Type**: Terms describing general categories, functions, genres, or aggregation levels for content.

- **Format**: The media type or dimensions of the resource.
- **Source**: Information regarding a prior resource from which the publication was derived.
- **Relation**: An identifier of an auxiliary resource and its relationship to the publication.
- **Coverage**: The extent or scope of the publication's content.
- **Rights**: A statement about rights, or a reference to one.

5.2. Hidden hedings

If the entire text of a heading is striked out (using any strike-out style), this heading will be hidden in the text. It will however still be visible in the EPUB table of contents.

6. The LaTeX package `ooomath.sty`

LO/AOO Math has a few features that are not available in standard LaTeX packages. To support those features, Writer2LaTeX will insert definitions in the LaTeX preamble. As an alternative, Writer2LaTeX provides an optional package `ooomath.sty`[10] which implements these constructions. This packages is only needed for documents containing formulas. Setting the option `use_ooomath` to true enables the use of this package (see section 4.1)

It is sufficient to put `ooomath.sty` in the same directory as the converted LaTeX document. It will however be more convenient if you install it in your TeX distribution. The proper place will usually be the "local texmf tree", please see the documentation of your TeX distribution. Below are specific instructions for TeX Live on Linux and and MikTeX on Windows:

6.0.1. Instructions for TeX Live (Linux)

If you use teTeX or TeX Live on Linux you can install `ooomath.sty` as follows:

Open a shell and type

```
texconfig conf
```

This will list the configuration details for TeX. Under the heading "Kpathsea" you will see a list of directories searched by TeX. You can put `ooomath.sty` in the subdirectory `tex` of any of these directories. Usually the directory

```
/home/<user name>/texmf/tex
```

can be used for the local texmf tree (you can create it if it doesn't exist).

[10] This pakcage replaces `writer.sty` used by older versions of Writer2LaTeX.

Next you should type

```
    texconfig rehash
```

to make TeX refresh it's filename database.

6.0.2. Instructions for MikTeX (Windows)

If you use MikTeX you can install `ooomath.sty` as follows:

First you should create a local texmf tree if it does note already exist. Start MikTeX settings (**All Programs – MikTeX – Maintenance – Settings**). Choose the **admin** variants if you want to install for all users. If the list on the tab page **Roots** is empty, create a suitable directory such as `C:\localtexmf`, click **Add** and select this directory.

Copy `ooomath.sty` to the `tex` subdirectory in the local texmf tree. If the subdirectory `tex` does not exist, you can create it.

Next you should start "MikTeX Options". On the tab page **General**, click the button **Refresh Now** to make MikTeX refresh it's filename database.

7. Using Writer2LaTeX from another application

7.1. Using Writer2LaTeX from a Java application

Writer2LaTeX features a simple API to convert documents from another Java application. Please see the javadoc for `writer2latex.jar` (the package `writer2latex.api`) for details.

The API offers a stream based as well as a file based interface for conversions.

Here's a simple example showing how to convert a file to LaTeX using a custom configuration (excluding exception handling) using the file based methods of the API.

```
import java.io.File;
import writer2latex.api.*;

// Create a LaTeX converter
Converter converter =
ConverterFactory.createConverter("application/x-latex");

// Configure the converter
Config config = converter.getConfig();
config.read(new File("myconfig.xml"));
config.setOption("inputencoding","latin1");

// Convert the document
ConverterResult result =
converter.convert(new File("mydocument.odt"),
"mydocument.tex");

// Write the files
result.write(new File("mydirectory"));
```

Using the stream based methods the conversion may look like this (assuming the option `save_images_in_subdir` is set to false):

```
import java.io.FileInputStream;
import java.io.FileOutputStream;

// Convert the document
ConverterResult result =
converter.convert(new FileInputStream("mydocument.odt"),
"mydocument.tex");

// Write the files
Iterator<OutputFile> docs = result.iterator();
while (docs.hasNext()) {
OutputFile docOut = (OutputFile) docs.next();
FileOutputStream fos =
new FileOutputStream("mydirectory/"+docOut.getFileName());
docOut.write(fos);
fos.flush();
fos.close();
```

```
    }
```

Writer2LaTeX also offers an interface for batch conversion of a directory into XHTML. For at simple example, see the source of `Application.java`.

7.2. Using Writer2LaTeX from a Basic macro

You can also access Writer2LaTeX through OOo/LO's api. Here's an example using a Basic macro, but the principle is the same for any other language with a UNO binding.

Writer2LaTeX is used as any other filter in OOo/LO. Using the parameter `FilterData`, you can provide specific options for Writer2LaTeX: You can give an URL for a configuration file to use and/or you can provide values for simple options (the order does not matter, the configuration file is always read first). In addition (XHTML export only), you can define a target template and an included style sheet.

This example exports a document to LaTeX using a specific configuration, but overriding the value of the option `use_colortbl`.

```
    Dim sUrl As String
    sUrl = <url to document>

    Dim sConfigUrl As String
    sConfigUrl = <url to config>

    Dim oFilterData(1) As New com.sun.star.beans.PropertyValue
    oFilterData(0).Name = "ConfigURL"
    oFilterData(0).Value = sConfigUrl
    oFilterData(1).Name = "use_colortbl"
    oFilterData(1).Value = "true"

    Dim oProps(2) As New com.sun.star.beans.PropertyValue
    oProps(0).Name = "FilterName"
    oProps(0).Value = "org.openoffice.da.writer2latex"
    oProps(1).Name = "Overwrite"
    oProps(1).Value = true
    oProps(2).Name = "FilterData"
    oProps(2).Value = oFilterData

    ThisComponent.StoreToURL(sUrl, oProps())
```

The table lists the names of the filters provided by Writer2LaTeX:

Format	***FilterName***
LaTeX	`org.openoffice.da.writer2latex`
BibTeX	`org.openoffice.da.writer2bibtex`
XHTML (text document)	`org.openoffice.da.writer2xhtml`
XHTML 1.1 (text document)	`org.openoffice.da.writer2xhtml11`
XHTML (spreadsheet)	`org.openoffice.da.calc2xhtml`

XHTML 1.1 (spreadsheet)	`org.openoffice.da.calc2xhtml11`
XHTML + MathML	`org.openoffice.da.writer2xhtml.mathml`
XHTML + MathML using xsl	`org.openoffice.da.writer2xhtml.mathml.xsl`
HTML5 (text document)	`org.openoffice.da.writer2xhtml5`
HTML5 (spreadsheet)	`org.openoffice.da.calc2xhtml5`
EPUB	`org.openoffice.da.writer2xhtml.epub`

This table lists the special properties available for the filter data (all are optional):

Property	***Purpose***
`ConfigURL`	Sets the URL for the configuration to use
`TargetTemplateURL`	Sets the URL for an XHTML template to use (XHTML and EPUB only)
`StyleSheetURL`	Sets the URL for a CSS style sheet to include (EPUB only)
`ResourceURL`	Sets the URL for a folder containing resources (images and fonts) referred in the style sheet (EPUB only). All files contained in this folder will be included with the style sheet in the same directory as the style sheet. The media type will be determined from the file extension. If you want to define the media type yourself, use the more complex property `Resources`.

Resources	Sets a list of resources (images and fonts) referred in the style sheet (EPUB only). This property is a semicolon separated list. Each entry is of the form `URL[[::file name]::mime type]` where the parts in square brackets are optional. For example `file://mycomputer/home/myself/images/bg.png::background.png:` to point to a png image which is referenced by the file name `background.png` in the style sheet. The resource file will be placed in the same directory as the style sheet.

The URLs can contain variables such as `$(user)` for the user installation of OOo/LO. Thus for example "`$(user)/myconfig.xml`" can be used to point to a configuration within the user installation. See

http://api.libreoffice.org/docs/idl/ref/servicecom_1_1sun_1_1star_1_1util_1_1PathSubstitution.html

for a list of available variables.

As a special feature, you can require one of Writer2LaTeX's standard configurations. To do this, the URL should start with an asterisk, for example "`*ultraclean.xml`".

7.3. Batch conversion with UNO

Writer2LaTeX also offers a uno service

```
org.openoffice.da.writer2xhtml.BatchConverter
```

providing batch conversion of a complete directory into another format (usually XHTML) with index pages. This service implements the interface `org.openoffice.da.writer2xhtml.XBatchConverter`, which provides a single method

```
// method
// org::openoffice::da::writer2xhtml::XBatchConverter::convert
void convert ( [in] string sSourceURL,
[in] string sTargetURL,
[in] sequence<com::sun::star::beans::PropertyValue> lArguments,
[in] XBatchHandler handler );
```

- The `sSourceURL` specifies the URL of the source directory
- The `sTargetURL` specifies the URL of the target directory
- The `handler` is an implementation of the call back interface `org.openoffice.da.writer2xhtml.XBatchHandler`, which is used to provide user interaction during the conversion process. See the IDL definition for documentation. If you use the batch conversion from a Basic macro, the interface must be implemented using `CreateUnoListener`.

The available arguments (for the parameter `lArguments`) are specified in this table

Argument	*Description*
`Recurse`	Set to `true` (default) if you want to convert subdirectories

`Uplink`	You can set this to an URL, which will be used as an uplink on the index page for the top level directory
`DirectoryIcon`	You can set this to an URL pointing to an image that represents a directory
`DocumentIcon`	You can set this to an URL pointing to an image that represents a document
`TemplateURL`	You can set this to an URL pointing to an XHTML template that should be used to generate the index page(s). Note that if you want to provide an XHTML template for the documents as well, this must be done using the FilterData (and the templates may be different).
`IndcludePdf`	Set this to `true` (default) if you want to include a pdf version of each file in addition to the XHTML version
`UseTitle`	Set this to `true` (default) if you want to use the document title in the index page rather than the file name
`UseDescription`	Set this to `true` (default) if you want to include the description of the document in the index page.
`WriterFilterName`	You can set this to the name of any Writer export filter you have available in your OOo/LO installation. The default is the XHTML export filter provided by Writer2xhtml (`org.openoffice.da.writer2xhtml`).

`WriterFilterData`	The structure of this argument depends on the filter, but for the default filter it is a sequence of `PropertyValue`s to pass options to the filter (see above).
`CalcFilterName`	You can set this to the name of any Calc export filter you have available in your OOo/LO installation. The default is the XHTML export filter provided by Writer2xhtml (`org.openoffice.da.calc2xhtml`).
`CalcFilterData`	The structure of this argument depends on the filter, but for the default filter it is a sequence of `PropertyValue`s to pass options to the filter (see above).

7.4. Converting from StarMath with a Basic macro

In addition to converting a complete document, you can also convert a single formula from StarMath to LaTeX. To do this, the uno service

```
org.openoffice.da.writer2latex.W2LStarMathConverter
```

is provided. This service supports two methods

```
string convertFormula ( [in] string sStarMathFormula );
string getPreamble ( );
```

- The method `convertFormula` converts a StarMath string to a LaTeX string
- The method `getPreamble` returns a LaTeX preamble suitable for processing the converted formulas.

This small example is a Basic macro that converts a few formulas and displays the result. Note that the last conversion triggers a definition of the LaTeX macro `\defeq` in `getPreamble()`.

```
Dim smc As Object
smc = CreateUnoService( _
"org.openoffice.da.writer2latex.W2LStarMathConverter")
MsgBox smc.convertFormula("1 over 2")
MsgBox smc.convertFormula("int from 1 to infty f(x)dx")
MsgBox smc.convertFormula("sqrt 3")
MsgBox smc.convertFormula("f(x) def x^2-1")
MsgBox smc.getPreamble()
```

8. Troubleshooting

Writer2LaTeX can convert quite large files. But if you have a very large document, you could get the following error message :

```
Exception in thread "main" java.lang.OutOfMemoryError:  Java heap space
```

In that case, you need to manually increase the memory available to the java virtual machine, for example using the following command to convert your document:

```
java -Xmx512M -jar writer2latex.jar bigFile.sxw out.tex
```

In the example, the heap size is set to 512 Megabyte of RAM. If you still get the "heap space" error, try setting the available memory even higher (assuming that your computer has enough physical RAM).

If you are using Writer2LaTeX as an export filter in OOo/LO, this problem will result in a generic error message saying that that document could not be written. To increase the heap size in this case, choose **Tools – Options – LibreOffice – Advanced**. Click **Parameters**, and add the parameter `-Xmx512M` (or higher).