User's manual for

Writer2LaTeX

Writer2LaTeX, Writer2BibTeX, Writer2xhtml and Calc2xhtml

version 0.5

Table of Contents

1 Introduction

1.1 What is Writer2LaTeX?

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

Actually it is 4 converters in one:

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

The old file formats for OpenOffice.org 1.x (or StarOffice 6/7) Writer and Calc documents are also supported.

Although Writer2LaTeX is a general OpenDocument converter, it is primarily designed for use with OpenOffice.org/StarOffice You can use Writer2LaTeX

- ...as a command line utility, independent of OpenOffice.org/StarOffice.
- ...as an export filter for OpenOffice.org 2.x, StarOffice 8 or NeoOffice 2.0[1].
- ...from another Java program.

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

Writer2LaTeX is a Java application, and thus should work on any platform that supports Java. You need Sun's Java 2 Virtual Machine (Runtime Environment), **version 1.4** or **1.5**. You can download this from `http://java.sun.com/getjava/download.html`. AFAIK Writer2LaTeX doesn't run (unmodified) under any other Java interpreter.

Note: In this manual OOo is used as an abbreviation of OpenOffice.org/StarOffice/NeoOffice.

[1] If you want to use Writer2LaTeX as an export filter in older versions, please use version 0.4.

2 Installation

2.1 How to install Writer2LaTeX for command line usage

Writer2LaTeX can work as a standalone command line utility (that is without OOo).

Installation for Microsoft Windows

To install Writer2LaTeX under Microsoft Windows follow these instructions:

1. Unzip `writer2latex05.zip` into some directory. This will create a subdirectory `writer2latex05`.
2. Add this directory to your PATH environment variable.
3. Open the file `w2l.bat` with a text editor and replace the path at the top of the file with the full path to Writer2LaTeX, for example

   ```
   set W2LPATH="c:\writer2latex05"
   ```

   (If you have extracted to the root of drive C, you don't have to edit this line.)

   At a command line type `java -version` to verify that the Java executable is in your path. If this is not the case or you have several Java versions installed you should edit the next line to contain the full path to the Java executable, eg.

   ```
   set JAVAEXE="C:\j2sdk1.4.0_01\bin\java''
   ```

Installation for Unix and friends

1. Unzip `writer2latex05.zip` into some directory. This will create a subdirectory `writer2latex05`.
2. Add this directory to your PATH environment variable or create a symbolic link to the file `w2l` from some directory in yout PATH.
3. Open the fle `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/writer2latex05"
   ```

   (If you have extracted into your home directory, you don't have to edit this line.)

   Open a command shell and type `java -version` to verify that the Java executable is in your path. If this is not the case or you have several Java versions installed you should edit the next line to contain the full path to the Java executable, ie.

   ```
   set JAVAEXE="/path/to/java/executable/''
   ```

4. Add execute permissions to `w2l` as follows:

   ```
   chmod +x w2l
   ```

2.2 How to install Writer2LaTeX as an export filter

Writer2LaTeX can work as an export filter for OOo Writer. This requires OpenOffice.org 2.x, StarOffice 8 or NeoOffice 2.0.

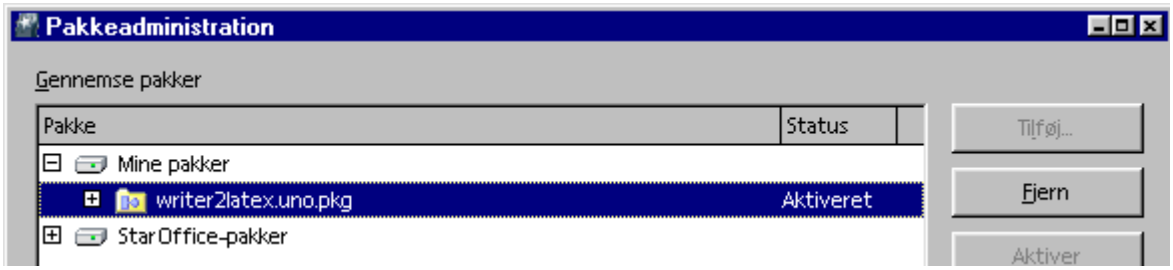
Note: OOo 2.0.4 includes Writer2LaTeX version 0.4 (LaTeX and BibTeX export only). I you install version 0.5, the built-in version will be disabled. If you uninstall version 0.5, the original version will reappear.

The following instructions covers all operating systems.

Important: Before you install Writer2LaTeX, you need to set up OOo to use Java. You can configure this in OOo under **Tools** – **Options**. Of course this requires that you have installed Java on your system.

Install Writer2LaTeX for a single user

1. Start OOo and chose **Tools** – **Extension Manager**. (**Tools** – **Package Manager** in versions prior to OOo 2.1).
2. Select **My packages** and select `writer2latex.uno.pkg` using the **Add** button.
3. You should now be able to see `writer2latex.uno.pkg` if you expand the list using the plus-icon:



4. Finally restart OOo.

Install Writer2LaTeX for all users

Note: If you want to install Writer2LaTeX for all users, you will normally need to log in as root/administrator.

Then the installation proceeds as follows:

1. Make sure that no OOo processes are running: Close all document windows and (under MS Windows) the Quick Starter.
2. From a command shell, navigate to the directory

   ```
   <OOo install>/program
   ```

   and type

   ```
   unopkg gui
   ```

   On unix-like systems you may have to type

```
./unopkg gui
```

3. Select **OpenOffice.org packages** and select `writer2latex.uno.pkg` using the **Add** button
4. You should now be able to see `writer2latex.uno.pkg` if you expand the list using the plus-icon, as above.

2.3 Uninstall Writer2LaTeX

To remove the Writer2LaTeX filters from your OOo installation, you should open the Extension Manager as described above, select `writer2latex.uno.pkg` and click **Remove.**

3 Using 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 25 different languages, latin, greek and cyrillic scripts and 8 inputencodings.

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

- You can use LaTeX as a typesetting engine for your OOo 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*.

- 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 be configured to produce a LaTeX document which strips most of the formatting and hence produces a clean LaTeX source from *any* source document.

- If you don't like to write LaTeX code by hand, you may use OOo as a simple graphical front-end for LaTeX. Using a special OOo Writer template and a special configuration file for Writer2LaTeX, you can create well-structured LaTeX documents that resembles "hand-written" LaTeX documents. You can compare this to the way LyX works.

  Writer2LaTeX does not provide an input filter for LaTeX. It is recommended to use Eitan M. Gurari's TeX4ht to convert LaTeX documents into OOo Writer format. Roundtrip editing OOo Writer ↔ LaTeX is not possible in general, but Writer2LaTeX+TeX4ht does provide some rudimentary support for this, see section 3.6.

3.1 The LaTeX package `ooomath.sty`

OOo Math has a few features that are not available in standard LaTeX packages. Hence Writer2LaTeX uses an optional package `ooomath.sty`[2] which implements these constructions. This packages is only needed for documents containing formulas. If it is not available, Writer2LaTeX will insert the necessary definitions in the LaTeX preamble.

It is sufficient to place `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 teTeX and MikTeX:

Instructions for teTeX (unix)

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

Open a shell and type

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

```
texconfig conf
```

This will list the configuration details for teTeX. 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 (you can create it if it doesn't exist).

Next you should type

```
texconfig rehash
```

to make teTeX refresh it's filename database.

Instructions for MikTeX (Windows)

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

Copy `ooomath.sty` to the `tex` subdirectory in the local texmf tree. With a standard installation this will be the directory

```
c:\localtexmf\tex
```

If this directory does not exist you should start "MikTeX Options" (you can find this in the Start Menu). On the tab page **Roots** you can see the location of 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.

3.2 Converting to LaTeX from the command line

To convert a file to LaTeX use the command line

```
w2l [-latex] [-config <configfile>] [options] <document to convert>
[<output path and/or file name>]
```

The parts in square brackets are optional.

This will produce a LaTeX file with the specified name. If no output file is specified, Writer2LaTeX will use the same name as the original document, but change the extension to `.tex`.

Examples:

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

or

```
w2l -config clean.xml mydocument.sxw
```

If you specify the `-config` option, Writer2LaTeX will load this configuration file before converting your document. You can read more about configuration in section 3.5. You can also specify any simple option described in this section directly on the command line, eg. to produce

a file suitable for processing with pdfLaTeX:

```
w2l -backend pdftex mydocument.sxw
```

The script `w2l` also provides a shorthand notation to use the sample configuration files included in `writer2latex05.zip`. The command line is

```
w2l [-ultraclean|-clean|-pdfscreen|-pdfprint|-article] <writer document
to convert> [<output path and/or file name>]
```

For example to produce a clean LaTeX file (ie. ignoring most of the formatting from the source document):

```
w2l -clean mydocument.sxw
```

It is recommended that you create your own scripts to support your own configuration file(s).

3.3 Converting to BibTeX from the command line

Writer2BibTeX extracts bibliography data to a BibTeX file. To do this use the commandline

```
w2l -bibtex <writer document to convert> [<output path and/or file name>]
```

You can also extract the data as part of the conversion to LaTeX, see section 3.5.

3.4 Using Writer2LaTeX and Writer2BibTeX as export filters

If you choose **File – Export** in Writer you should be able to choose **LaTeX 2e**, **BibTeX** as file type.

Note: You have to use the export menu because there is no import filter for LaTeX/BibTeX. You should always save in the native format of OOo as well!

3.5 Configuration

LaTeX export can be configured with a configuration file. Where the configuration is read from depends on how you use Writer2LaTeX:

If you use Writer2LaTeX as an export filter in OOo, the configuration is handled as follows:

- The file `writer2latex.xml` is read from the user installation directory of OOo

  On linux/unix usually something like `<home directory>/.OpenOffice.org2/user`

  On windows usually something like `<user profile>\OpenOffice.org2\user`

  If the file does not exist, it will be created automatically.

If, on the other hand, you use Writer2LaTeX from the command line, you will have to specify on the command line which configuration file to use.

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" />
  <option name="debug" value="false" />
  <heading-map max-level="6">
    <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-level-map writer-level="5" name="paragraph"
      level="4" />
    <heading-level-map writer-level="6" name="subparagraph"
    level="5" />
  </heading-map>
  <custom-preamble />
  <style-map name="Quotations" class="paragraph"
    before="\begin{quote}" after=\end{quote} />
  <string-replace input="LaTeX" latex-code="{\LaTeX}" />
</config>
```

The meaning of each part is explained in the following sections. Writer2LaTeX comes with five sample 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 most 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.
- `article.xml` to produce a LaTeX article, see section 3.6.

Basic options

- The option `backend` can have any of the values `generic`, `dvips`, `pdftex` (default) and `unspecified`. This will create LaTeX files suitable for any backend/dvi driver, dvips or pdfTeX respectively. The last value does not assume any specific backend. This option affects export of graphics: Only file types than can be handled by the backend are included; other types will be commented out. If you use `unspecified`, no graphics will be commented out.
- If the option `no_preamble` is set to `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.
- The option `inputencoding` can have any of the values `ascii` (default), `latin1`, `latin2`, `iso-8859-7`, `cp1250`, `cp1251`, `koi8-r` or `utf8`. The latter requires Dominique Unruh's `ucs.sty`.
- If the option `multilingual` is set to false, Writer2LaTeX will assume that the document is written in one language only – otherwise all the language information contained in the document will be used.
- The option `split_linked_sections` specifies that a linked section should be exported to a separate LaTeX-file. Default is `false`.
- The option `split_toplevel_section` specifies that all sections should be exported to a separate LaTeX-file, excluding nested sections. Default is `false`.
- The option `wrap_lines_after` 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.

Options for document structure

- The option `documentclass` is the name of the documentclass to use (default is `article`).
- The option `global_options` is a list of global options to add to the documentclass (the default value is an empty string).
- The `heading_map` section specifies how headings in OOo 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 OOo).

Table options

- The option `use_longtable` is used to specify that `longtable.sty` should be used to export tables which may break across pages. Default is `false`.
- The option `use_supertabular` 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`).

- The option `use_tabulary` is used to specify that `tabulary.sty` should be used to export tables. Default is `false`.
- The option `simple_table_limit` can be set to any non-negative integer (default is `0`). Table cells in OOo 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 all these paragraphs contains 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`.
- The option `use_colortbl` 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.`
- The option `float_tables` can be used to specify that you want to include graphics and text boxes in a `table` environment. Default is `false`.
- The option `float_options` can be used to give placement options to the figure floats, eg. h for *here*. Default is empty (default placement).
- The option `table_sequence_name` can be set to a sequence name in the source document. OpenDocument has a very weak sense of table captions: A table caption is a paragraph containing a sequence number. If you use OOo's defaults, Writer2LaTeX can guess which sequence name to use. If it fails, you can give the name in this option (default is empty).
- The option `use_caption` can be used 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.

Graphics options

- The option `float_figures` can be used to specify that you want to include graphics and text boxes in a `figure` environment. Default is `false`.
- The option `float_options` can be used to give placement options to the figure floats, eg. h for *here*. Default is empty (default placement).
- The option `figure_sequence_name` 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 OOo's defaults, Writer2LaTeX can guess which sequence name to use. If it fails, you can give the name in this option (default is empty).
- The option `use_caption` can be used 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.
- The option `align_frames` can be used to specify, that all graphics and text boxes should be included in a `center` environment. If you don't want that, set this option to `false`. Default is `true.`

- The option `keep_image_size` can be used to specify that the size of images should not be exported, hence LaTeX should use the original size of the image. Default is `false`.
- The option `image_options` 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).
- The option `remove_graphics_extension` 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}`. This can be handy if you use an external tool to convert the graphics files (you should set the option `backend` to `unspecified` in this case).

Font and symbol options

- The option `greek_math` 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 awful.
- The option `use_ooomath` can have the values `true` or `false` (default). This enables the use of the LaTeX package `ooomath.sty`. If this package is not used, the necessary definitions will be included in the LaTeX preamble, which may become quite long – so using `ooomath.sty` is recommended.
- The option `use_pifont` can have the values `true` or `false` (default). This enables the use of *Zapf Dingbats* using the LaTeX package `pifont.sty`.
- The option `use_wasysym` can have the values `true` or `false` (default). This enables the use of the *wasy* symbol font using the LaTeX package `wasysym.sty`.
- The option `use_ifsym` can have the values `true` or `false` (default). This enables the use of the *ifsym* symbol font using the LaTeX package `ifsym.sty`.
- The option `use_bbding` can have the values `true` or `false` (default). This enables the use of the *bbding* symbol font (a clone of Zapf Dingbats) using the LaTeX package `bbding.sty`.
- The option `use_eurosym` can have the values `true` or `false` (default). This enables the use of the *eurosym* symbol font using the LaTeX package `eurosym.sty`.
- The option `use_tipa` can have the values `true` or `false` (default). This enables the use of phonetic symbols using the LaTeX package `tipa.sty`.

Options for various packages

- The option `use_hyperref` can have the values `true` (default) or `false`. This enables use of the package `hyperref.sty` to include hyperlinks in the LaTeX document.
- The option `use_color` can have the values `true` (default) or `false`. This enables use of the package `hyperref.sty` to apply color in the LaTeX document.
- The option `use_endnotes` can have the values `true` or `false` (default). This enables use of the package `endnotes.sty` to include endnotes in the LaTeX document. If set to `false`,

endnotes will be converted to footnotes.

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

Various options

- The option `notes` 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}`.
- The option `tabstop` 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"`, `"\hspace{2em}"`

Options for BibTeX

- The option `use_bibtex` can have the values `true` or `false` (default). This enables the use of BibTeX for bibliography generation. If it is set to `false`, the bibliography is included as text.
- The option `bibtex_style` can have any BibTeX style as value (default is `plain`). This is the BibTeX style to be used in the LaTeX document.

Options to control export of page formatting

- The option `page_formatting` 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.
- The option `use_geometry` 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.
- The option `use_fancyhdr` 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.

Options to control export of other formatting

In Writer, formatting is controlled by styles. You can control how much formatting is exported using the following options[3]. Note that these options has a major impact on the structure of the LaTeX document created.

- 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 as well as 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 is used for listings, headings are reformatted using the `\@startsection` command etc.
  - `convert_all` will preserve *all* supported formatting.
- The option `ignore_empty_paragraphs` 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 `\bigskip`.
- The option `ignore_double_spaces` 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 (\ ).
- The option `ignore_hard_page_breaks` 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).
- The option `ignore_hard_line_breaks` can have the values `true` or `false` (default). Setting the option to `true` will instruct Writer2LaTeX to ignore hard line breaks (shift-Enter).

Options for strict handling of content

The following options can be used if you want a very strict control with the content allowed in the document. The options

- `other_styles`
- `image_content`
- `table_content`

Can all have the values `accept` (default), `ignore`, `warning` and `error`.

This controls how various content should be handled by Writer2LaTeX. The option

[3]Note that these options have changed a lot since version 0.3.2.

`other_styles` controls paragraph and text content, for which there is no style map (see below). The other options control images and tables.

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.

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. The following examples are from the sample 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" class="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" class="paragraph" before="\part{" after="}"
line-break="false" />
```

This is a rule, that maps paragraph 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" class="paragraph-block"
next="Preformatted Text" before="\begin{verbatim}" after="\end{verbatim}"
/>
<style-map name="Preformatted Text" class="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" class="paragraph" before="\title{" after="}"
line-break="false" />
<style-map name="author" class="paragraph" before="\author{" after="}"
line-break="false" />
<style-map name="date" class="paragraph" before="\date{" after="}"
line-break="false" />
<style-map name="Title" class="paragraph-block" next="author;date"
before="" after="\maketitle" />
<style-map name="author" class="paragraph-block" next="Title;date"
before="" after="\maketitle" />
```

```
<style-map name="date" class="paragraph-block" next="Title;author"
before="" after="\maketitle" />
```

This will produce code like this:

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

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

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

String replace

Often LaTeX requires special care to typeset certain constructions. For example according to german typografical 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 OOo document, the `latex-code` is the LaTeX code to export for this text.

Math symbols

In OOo 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`.

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.

3.6 Using OpenOffice.org as a frontend for LaTeX

Writer2LaTeX has some simple support for using OOo as a frontend for LaTeX. The long term goal of this is to turn Writer into a near-wysiwyg LaTeX editor somewhat like LyX.

Here is a short description:

Create a new document based on the template LaTeX-article.stw.

This template contains a number of styles that corresponds to LaTeX code:

OOo Writer style	***OOo Writer class***	***LaTeX code***
Title[4]	paragraph	`\title{...}`[5]
author	paragraph	`\author{...}`
date	paragraph	`\date{...}`
abstract title	paragraph	renews `\abstractname`
abstract	paragraph	`abstract` environment
part	paragraph	`\part{...}`
Heading 2	paragraph	`\section{...}`
Heading 3	paragraph	`\subsection{...}`
Heading 4	paragraph	`\subsubsection{...}`
Heading 5	paragraph	`\paragraph{...}`
Heading 6	paragraph	`\subparagraph{...}`
flushleft	paragraph	`flushleft` environment
flushright	paragraph	`flushright` environment
center	paragraph	`center` environment
verse	paragraph	`verse` environment
quote	paragraph	`quote` environment
quotation	paragraph	`quotation` environment
Preformatted text	paragraph	`verbatim` environment[6]
theorem	paragraph	`theorem` environment
itemize	paragraph	`itemize` list
enumerate	paragraph	`enurerate` list
List Heading	paragraph	`description` list (item label)
List Contents	paragraph	`description` list (item text)
verb	text	`\verb|...|`
Emphasis	text	`\emph{...}`
Strong Emphasis	text	`\textbf{...}`

OOo Writer style	*OOo Writer class*	*LaTeX code*
textrm	text	`\textrm{...}`
textsf	text	`\textsf{...}`
texttt	text	`\texttt{...}`
textup	text	`\textup{...}`
textsl	text	`\textsl{...}`
textit	text	`\textit{...}`
textsc	text	`\textsc{...}`
textmd	text	`\textmd{...}`
textbf	text	`\textbf{...}`
tiny	text	`{\tiny ...}`
scriptsize	text	`{\sciptsize ...}`
footnotesize	text	`{\footnotesize ...}`
small	text	`{\small ...}`
normalsize	text	`{\normalsize ...}`
large	text	`{\large ...}`
Large	text	`{\Large ...}`
LARGE	text	`{\LARGE ...}`
huge	text	`{\huge ...}`
Huge	text	`{\Huge ...}`

If you use these styles and uses the configuration file `article.xml` when you convert your document with Writer2LaTeX, you will get a result that resembles a handwritten LaTeX file. Note that hard formatting and any other styles will be ignored.

Roundtrip editing

Writer2LaTeX does not provide a filter, that converts LaTeX files back into OOo Writer format. This is however possible with Eitan M. Gurari's TeX4ht system (http://www.cse.ohio-state.edu/~gurari/TeX4ht/mn.html). If you use Writer2LaTeX (with `article.xml`) together with TeX4ht's OOo mode (`oolatex`), simple roundtrip edition LaTeX ↔ OOo Writer is supported. Beware of information loss if you do this – do not use this roundtrip for existing LaTeX or Writer documents!

As a genereal rule, you should save your document in the native OOo Writer format and convert to LaTeX when you are finished (or want to see the result).

[4]The use of italics in this table indicates styles that are predefined in OOo. The names of these styles will be localized if you use a non-english version of OOo.

[5]Also `\maketitle` is added at the end of a sequence of Title, author and date.

[6]Only characters available in the inputenc are accepted. Other characters are converted to question marks and other content is discarded, eg. footnotes.

4 Using Writer2xhtml and Calc2xhtml

Writer2xhtml is producing standards compliant XHTML files, in particular it can be used to put math on the web using the XHTML + MathML combination. Thus Writer2xhtml can convert into any of these XHTML variants:

- XHTML 1.0 strict, which follows the guidelines for HTML compatibility, so that the output should be viewable with any browser that supports HTML 4.
- XHTML 1.1 + MathML 2.0, which currently is viewable with the Mozilla and Amaya browsers only.
- XHTML 1.1 + MathML 2.0 using XSL transformations from the W3C Math Working Group to make the file viewable also in some browsers that needs a plugin to display MathML, eg. Internet Explorer with MathPlayer plugin.

  This is how W3C's Math Working Group recommends to put "math on the web".

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 + MathML 2.0	`.xhtml`	`application/xhtml+xml`
XHTML 1.1 + MathML 2.0 (with xsl transformation)	`.xml`	`application/xml`

Writer2xhtml is quite flexible; in particular with respect to the handling of formatting:

- 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 with an online look and feel.
- You can use your own style sheet and let Writer2xhtml convert the content only. You can map styles in OOo to xhtml elements and css classes from your style sheet, see sections 4.3 and 4.4

Calc2xhtml is a companion to Writer2xhtml that produces XHTML 1.0 strict from your Calc documents.

4.1 Converting to XHTML from the command line

To convert a file to XHTML use the command line

```
w2l [options] <document/directory to convert>
  [<output path and/or file name>]
```

The available options are

- `-xhtml`, `-xhtml+mathml` and `-xhtml+mathml+xsl` specifies the output format (if you leave this out, the output format will be LaTeX!).

- `-recurse` to specifiy that batch conversion of a directory should recurse into subdirectories.
- `-template filename` to specify a template file. Writer2xhtml will use this file as a template for the converted document. The template must contain an element with the attribute `id="content"`. This element should accept block content, eg. `div` or `td`. Optionally it can also contain elements with attributes `id="header"` and `id="footer"`. These will be used for navigation links.
- `-config filename` to specify a configuration file. Writer2xhtml will load this configuration file before converting your document. You can read more about configuration in section 4.3.
- `-option value` to set any simple configuration option, where option the name of a simple option, see section 4.3.

This will produce an XHTML file with the specified name. If no output file is specified, Writer2xhtml will use the same name as the original document, but a different file extension.

Examples:

```
w2l -xhtml+mathml+xsl mydocument.sxw
```

or

```
w2l -xhtml -config myconfig.xml mydocument.sxw
```

The script `w2l` also provides a shorthand notation to use the sample configuration file included in `writer2latex05.zip`. The command line is

```
w2l -cleanxhtml <writer document to convert> [<output path and/or file
name>]
```

This configuration file produces a "clean" xhtml file (see section 4.4), for example:

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

It is recommended that you create scripts to support your own configuration files.

4.2 Using Writer2xhtml as an export filter

If you choose **File − Export** in Writer you should be able to choose **XHTML 1.0 strict**, **XHTML 1.1 + MathML 2.0** or **XHTML 1.1 + MathML 2.0 (xsl)** as file type. Using Calc2xhtml as an export filter is not yet supported.

Note: You have to use the export menu because Writer2xhtml does not provide an import filter for XHTML. You should always save in the native format of OOo as well!

4.3 Configuration

XHTML export can be configured with a configuration file. Where the configuration is read from depends on how you use Writer2xhtml:

If you use Writer2xhtml as an export filter in OOo, the configuration is handled as follows:

- The file `writer2latex.xml` is read from the user installation directory of OOo

On linux/unix usually something like <home directory>/`.OpenOffice.org2/user`

On windows usually something like `<user profile>\OpenOffice.org2\user`

If the file does not exist, it will be created automatically.

If, on the other hand, you use Writer2xhtml from the command line, you will have to specify on the command line which configuration file to use.

The configuration is a file in xml format. Here is a sample configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
  <option name="xhtml_custom_stylesheet" value="/mystyle.css" />
  <option name="xhtml_ignore_styles" value="false" />
  <option name="xhtml_use_dublin_core" value="true" />
  <option name="xhtml_convert_to_px" value="true" />
  <option name="xhtml_split_level" value="1" />
  <xhtml-style-map name="mystyle" class="paragraph" element="p"
css="mycssstyle" />
</config>
```

Options

- The option `xhtml_no_doctype` can have the values `true` or `false` (default). When this option is `true`, 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.
- The option xhtml_encoding (default UTF-8) is used to specify the character encoding to use for the xhtml document.
- The option `xhtml_custom_stylesheet` is used to specify an URL to your own, external stylesheet. If the value is empty or the option is not specified, no external stylesheet will be used.
- The option `xhtml_formatting` is used to specify how much text formatting (character, paragraph and list formatting) to export[7]. 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.

[7]This and the following options replaces the former option `xhtml_ignore_styles`.

  - `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.

- The option `xhtml_frame_formatting` is used for the same purpose for frame formatting.
- The option `xhtml_section_formatting` is used for the same purpose for section formatting. (But note that OOo does not offer section styles currently).
- The option `xhtml_table_formatting` is used for the same purpose for table formatting. (But note that OOo does not offer table styles currently).
- The option `xhtml_ignore_table_dimensions` is used to specify that you don't want table dimensions (table width, column width and row height) to be exported, but want to leave the layout of the tables to the browser.
- The option `xhtml_use_dublin_core` is used to specify if Dublin Core Meta data should be exported (the format will be as specified in http://dublincore.org/documents/dcq-html/). If the value is `false`, it will not be exported.
- The option `xhtml_convert_to_px` can have the values `true` (default) or `false`. When this option is `true`, 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 `xhtml_scaling` option. Eg. a scaling `75%` will change the resolution to 72ppi.
- The option `xhtml_scaling` is used to specify a scaling of all formatting, ie. to get a different text size than the original document. The value must be a percentage.
- The option `xhtml_column_scaling` is used to specify an additional scaling for table colums. The value must be a percentage.
- The option `xhtml_split_level` 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.
- The option `xhtml_calc_split` is used to specify that the Calc 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 option `xhtml_uplink` is used to specify a link which brings the user up in the page hierarchy. For example `"../index.html"`.
- The option `xhtml_directory_icon` is used to specify an (icon) image that represents a directory. This is used when Writer2xhtml creates index pages for a directory.
- The option `xhtml_document_icon` is used to specify an (icon) image that represents a document. This is used when Writer2xhtml creates index pages for a directory.
- The option `xhtml_use_list_hack` is used to fix a problem with continued lists. This will export a list that continues on level 2 or below like `<ol><ol><li>...</li></ol></ol>`, which is *not* valid in xhtml, but works in browsers. Also two deprecated attributes are used to continue numbering. Default is `false`.

- The option `xhtml_tabstop_style` can be used 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; }`.
- The option `ignore_double_spaces` 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.
- The option `ignore_empty_paragraphs` can have the values `true` (default) or `false`. Setting the option to `true` will instruct Writer2xhtml to ignore empty paragraphs..
- The option `ignore_hard_line_breaks` can have the values `true` or `false` (default). Setting the option to `true` will instruct Writer2xhtml to ignore hard line breaks (shift-Enter).

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 OOo paragraph styles to xhtml elements -->
  <xhtml-style-map name="Text body" class="paragraph"
           element="p" css="(none)" />
  <xhtml-style-map name="Sender" class="paragraph"
           element="address" css="(none)" />
  <xhtml-style-map name="Quotations" class="paragraph"
           block-element="blockquote" block-css="(none)"
           element="p" css="(none)" />

  <!-- map OOo text styles to xhtml elements -->
  <xhtml-style-map name="Citation" class="text"
           element="cite" css="(none)" />
  <xhtml-style-map name="Emphasis" class="text"
           element="em" css="(none)" />

  <!-- map hard formatting attributes to xhtml elements -->
  <xhtml-style-map name="bold" class="attribute"
           element="b" css="(none)" />
  <xhtml-style-map name="italics" class="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.
- `class` specifies the styles class in Writer; this can either be `text`, `paragraph`, `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` and `subscript`.
- `element` specifies the XHTML element to use when converting this style. This is not used for frame and list styles.
- `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 styles. It is used to specify a block XHTML element, that should surround several exported paragraphs with this style.
- `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 OOo style" class="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 OOo style</p>
  <p class=’’par_style’’>Yet another</p>
</div>
```

Note that the rules for hard formatting are only used when `xhtml_ignore_styles` is set to `true`. 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.

4.4 Using OpenOffice.org 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 OpenOffice.org):

OOo Writer style	***OOo Writer class***	***XHTML element***
Text body	paragraph style	`p`
Sender	paragraph style	`address`
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`
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.

Warning: Some elements are not allowed inside `pre`, so this might in some cases lead to invalid documents. This will be fixed in a later version of Writer2xhtml.

5 Using Writer2LaTeX from another Java application

Version 0.5 features a new API to use Writer2LaTeX from another Java application. Please see the javadoc for the package `writer2latex.api` for details.

Here's a simple example[8] showing how to convert a file to LaTeX using a custom configuration (excluding exception handling):

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.Enumeration;
import writer2latex.api.*;
import writer2latex.util.Config;

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

// Create a configuration
Config config = new Config();
config.read(new FileInputStream("myconfig.xml"));
config.setOption("inputencoding","latin1");
converter.setConfig(config);

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

// Write the files
Enumeration docEnum = dataOut.getDocumentEnumeration();
while (docEnum.hasMoreElements()) {
    OutputFile docOut = (OutputFile) docEnum.nextElement();
    FileOutputStream fos =
        new FileOutputStream(docOut.getFileName());
    docOut.write(fos);
    fos.flush();
    fos.close();
}
```

[8] The handling of the configuration will change in version 1.0.

6 Troubleshooting

If you have to convert a 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 -Xmx128M -jar writer2latex.jar bigFile.sxw out.tex
```

In the example, the heap size is set to 128 Megabyte of RAM. If you still get the "heap space" error, try setting the available memory to 256 or 512 Megabyte (assuming that your computer has enough physical RAM).

A few memory optimizations are planned for version 1.0, which should make it possible to convert a wider range of documents without increasing the heap size in java.