User's manual for

Writer2LaTeX

Writer2LaTeX, Writer2BibTeX, Writer2xhtml and Calc2xhtml

version 1.0.2

Table 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 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.

Although Writer2LaTeX is a general OpenDocument converter, it is primarily designed for use with OpenOffice.org and it's derivatives (e.g. StarOffice or NeoOffice). You can use Writer2LaTeX

- ...as an *export filter* for OpenOffice.org 2.2 or later[2], or equivalent versions of StarOffice or NeoOffice[3].
- ...as a *command line utility*, independent of OpenOffice.org.
- ...as a *Java library* providing conversions from OpenDocument for other Java programs.

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 5** or later[4]. You can download this from `http://www.java.com`. Writer2LaTeX also works with **OpenJDK 6**, but has not been tested with other java implementations.

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

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

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 25 different languages, latin, greek and cyrillic scripts and 8 input-encodings.

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

[2]For OpenOffice.org 2.0.4–2.1, you can use Writer2LaTeX 0.5.0.2, for OpenOffice.org 2.0–2.0.3, you can use Writer2LaTeX 0.5, for OpenOffice.org 1.1, you can use Writer2LaTeX 0.4.

[3]Unfortunately these variants does not use the same version numbers. For StarOffice you should use StarOffice 8, Product Update 6 or later. For NeoOffice I don't know how the version numbering relates to that of OpenOffice.org.

[4]The source is compatible with the legacy Java 1.4.

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*. 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 development and will provide the tools to make you use OOo 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 OOo Writer. This requires OpenOffice.org 2.2 or later, or equivalent versions of StarOffice or NeoOffice.

Two OOo extensions are provided:

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

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

Note: OOo 2.0.4 and later already includes Writer2LaTeX version 0.4 (LaTeX and BibTeX export only). If you install version 1.0, the built-in version will be hidden (to avoid confusion). If you uninstall version 1.0, the original version will reappear.

Also note: Before you install the Writer2LaTeX extensions, 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 a Java runtime environment on your system.

Important: If you have installed Writer2LaTeX 0.5, you must uninstall this version first. (This is not necessary if you have installed Writer2LaTeX 0.5.0.2.)

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

http://extensions.services.openoffice.org/resources/user/howto_install

2.2 Using the filters

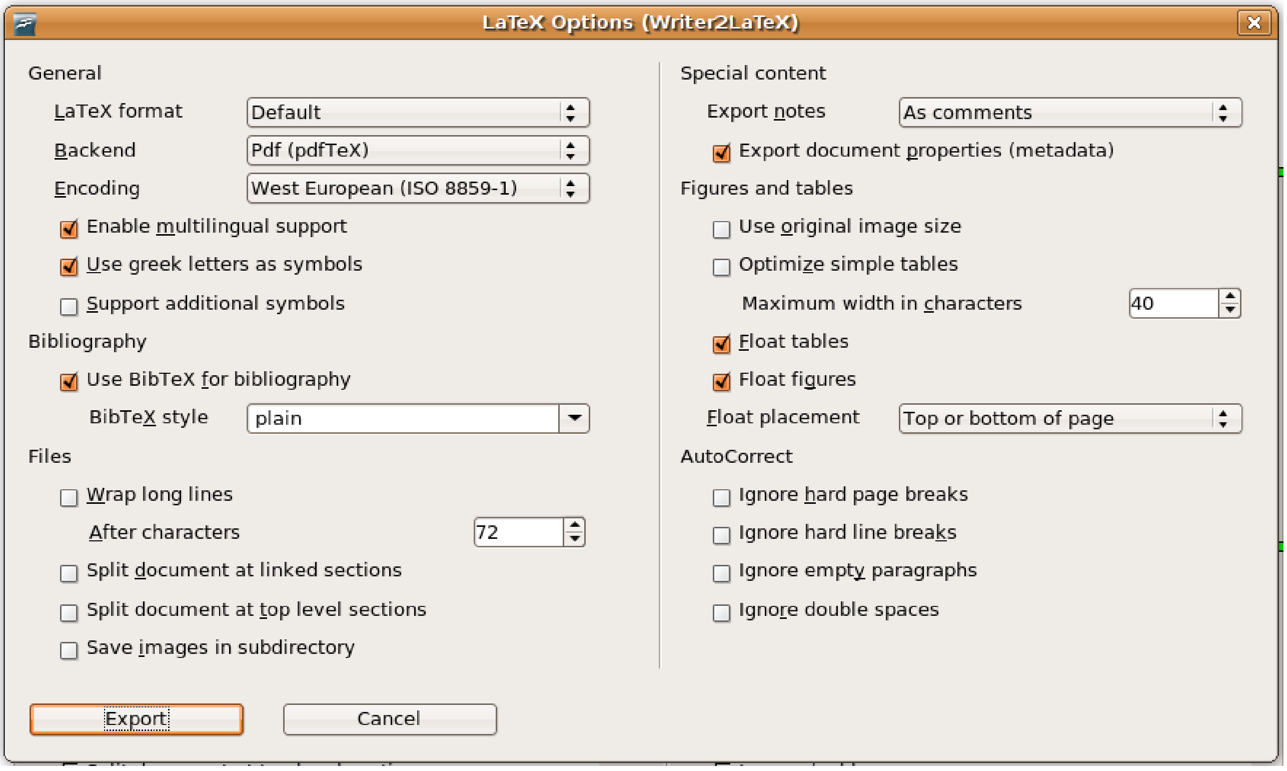
The filters provided by Writer2LaTeX are all *export filters*. This means that the filters are to be found in the **File** – **Export** menu in Writer or Calc.

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:


LaTeX Options (Writer2LaTeX)

General
- LaTeX format: Default
- Backend: Pdf (pdfTeX)
- Encoding: West European (ISO 8859-1)
- ✓ Enable multilingual support
- ✓ Use greek letters as symbols
- ☐ Support additional symbols

Bibliography
- ✓ Use BibTeX for bibliography
- BibTeX style: plain

Files
- ☐ Wrap long lines
- After characters: 72
- ☐ Split document at linked sections
- ☐ Split document at top level sections
- ☐ Save images in subdirectory

Special content
- Export notes: As comments
- ✓ Export document properties (metadata)

Figures and tables
- ☐ Use original image size
- ☐ Optimize simple tables
- Maximum width in characters: 40
- ✓ Float tables
- ✓ Float figures
- Float placement: Top or bottom of page

AutoCorrect
- ☐ Ignore hard page breaks
- ☐ Ignore hard line breaks
- ☐ Ignore empty paragraphs
- ☐ Ignore double spaces

The individual options are explained below. Click **Export** to initiate the export or **Cancel** to close the dialog without exporting the document.

General options

LaTeX format

Conceptually, Writer and LaTeX are quite different. A LaTeX document is usually based on a certain *document class*, that determines the general layout and formatting of the document. In addition the use of several LaTeX *packages* may change various aspects of the LaTeX document.

The result of a conversion into LaTeX will thus depend very much on which LaTeX packages are used and how much formatting it is desired to preserve.

Writer2LaTeX offers a number of default formats, all based on the standard LaTeX document class *article*. Each of the formats defines the LaTeX packages to use and the handling of formatting.

- **Ultra-clean article** will create a very basic LaTeX article, with almost no formatting preserved.
- **Clean article** will also create a default LaTeX article, but does preserve some basic formatting from the Writer document, such as boldface, color and hyperlinks.
- **Default** creates a LaTeX article preserving as much formatting as it is possible without any significant deviations from LaTeX standards.
- **Print optimized** on the other hand creates a LaTeX article preserving as much formatting as possible. The result will resemble the Writer document, but will look slightly

different from a standard LaTeX article (and the LaTeX code will be less readable).

- **Screen optimized (pdf)** also creates a LaTeX article preserving most of the formatting, but optimized for screen viewing (using the package `pdfscreen`) rather than printing.
- **Custom** is a user defined format, see section 2.7.

Advanced users can extend the list with further formats using *configuration packages*, see section 2.8.

Backend

When processing a LaTeX document, the final result is a document in a certain *backend* format. The handling of certain aspects of the document, in particular graphics, depends on the backend. With this option you can select the backend format.

- **Generic** will create a LaTeX document that can be processed with any flavour of LaTeX, usually with a file in DVI format as the result. Graphics is *not* supported with this backend.
- **Pdf (pdfTeX)** will create a LaTeX document that is suitable for processing with pdfLaTeX. Graphics are converted to a format that can be included in pdf files.
- **Postscript (dvips)** will create a LaTeX document that is suitable for generating documents in Postscript format, usually by post processing with *dvips*. Graphics will be converted to *Encapsulated postscript* format.
- **Unspecified** will create a LaTeX document with no particular backend in mind. All graphics will be exported in the original format, and it is up to the user to handle them.

If you have selected the format *Screen optiomized (pdf)*, you cannot select the backend, which will always be pdf.

Encoding

A LaTeX document is a *text file*, which always uses a certain *character encoding*. The character encoding is important if the LaTeX file is going to be edited in a text editor: You should select an encoding that is supported by your text editor. This setting is also important to get optimal support for international characters: If you for example use pdfTeX, searching in the final pdf document will only work for characters supported by the selected character encoding.

Currently, Writer2LaTeX supports 8 different encodings which together are suitable for a large number of languages written with either latin, greek or cyrillic letters. Currently asian (CJK) and bidirectional (CTL) scripts are not supported.

Enable multilingual support

If you check this option, all the language settings in the Writer document will be exported to LaTeX. Sometimes the language settings in a Writer document are not correct, so if you have a document that is written in one language only you may want to uncheck this option. This will produce a cleaner LaTeX file because you may avoid a large number of language selections.

Use greek letters as symbols

Greek letters used in latin text are often used as symbols, such as the number π or the word γ-radiation. By checking this option, all greek letters used within latin or cyrillic text will be treated as mathematical symbols, which will produce a slightly better result – and also not require that greek text fonts are available in the LaTeX installation. This option has no effect on greek text (provided the language is set correctly in the Writer document).

Support additional symbols

If you select this option, LaTeX will load some additional packages containing support for various symbols: A better looking euro-symbol, phonetic characters, dingbats and various other symbols and geometric shapes.

Bibliography options

Use BibTeX for bibliography

Usually the bibliography in a LaTeX document is generated by the companion program *BibTeX*. If you check this option, all the bibliographic references will be exported to BibTeX format for later processing with the BibTeX program.

BibTeX style

If you use BibTeX, you should also select a BibTeX *style* to be used when generating the bibliography. Select one of the predefined styles or type the name of any BibTeX style which is available in your LaTeX installation.

Files options

Wrap long lines

Checking this option wraps long lines in the generated LaTeX file. This enhances the readability if the file is later edited in a text editor. If you use a text editor that wraps lines automatically, you should uncheck this option.

After characters

If you choose to wrap long lines, they will be wrapped as soon as possible after this number of characters.

Split document at linked sections

Checking this option will create separate LaTeX files for sections in the Writer document with linked content. This can be an advantage if the LaTeX document is later edited in a text editor.

Split document at top level sections

Checking this option will create separate LaTeX files for all top level sections in the Writer document (but not for nested sections).

Save images in subdirectory

Writer2LaTeX normally saves images associated with the document in the same directory as the LaTeX document. If the document contains a large number of images it may be more convenient to save the images in a separate subdirectory. This option will create a subdirectory with the same name as the LaTeX document to store the images.

Options for special content

Export notes

This option determines how to export notes (also known as annotations) in the Writer document

- **Do not export** will ignore the notes
- **As comments** will export the notes as comments in the LaTeX file. They will not be visible in the final document after processing with LaTeX.
- **As marginal notes** will put the notes in the margin of the document. In this case they will be visible in the final document, but beware that the notes will be lost if the margin is too narrow.
- **As pdf annotations** will export the notes as pdf text annotations. If the pdf viewer supports it, you will be able to read the notes. Adobe Reader displays text annotations with a yellow icon, which allows you to open and read the note. If the document is not processed with pdfTeX, the notes will be converted to marginal notes.

Export document properties (metadata)

If you check this option, 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 not checked, only the title will be exported.

Options for figures and tables

Use 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 select this option, the original (unscaled) image size will be used.

Optimize simple tables

Normally Writer2LaTeX will generate tables with the same column widths as in the original document. For tables with simple content it may be more desirable to create tables which are as narrow as possible, with only one line of text per cell. Compare the table

Simple content	Simple content
Simple content	Simple content

to the optimized table

Simple content	Simple content
Simple content	Simple content

If you check this option, Writer2LaTeX will try to optimize tables.

Maximum width in characters

If you have chosen to optimize simple tables, you have to specify the maximum width of the table, measured in the number of characters. If you for example set the number to 50, only tables with a total width of 50 or fewer characters will be optimized.

Float tables

In Writer you can either choose that the rows of a table must be kept together on one page or that the table may split across page breaks. Keeping a table on one page may be desirable to increase the readability of the table, but it may also leave large white gaps at the bottom of the page. In LaTeX this problem is solved with *floating tables*: A table can automatically move to another position which fixes the whitespace problem. If you check this option, all tables that are not allowed to break across pages are exported as floating tables.

Float figures

A similar option is available for figures (graphics, text boxes). If you check this option, figures are converted to *floating figures* which can move in the document to reduce whitespace. This will not affect figures anchored *as character*.

Float placement

If you choose to let either tables or figures float, use this option to specify the placement of the floats:

- **Top or bottom of page** will place the floats either at the top or the bottom of a page.
- **Top of page** will place floats at the top of a page.
- **Bottom of page** will place floats at the top of a page.

- **Here or top of page** will place floats at their original position, if there is room left on the page, and otherwise at the top of a page.
- **Here or bottom of page** will place floats at their original position, if there is room left on the page, and otherwise at the bottom of a page.

In all cases it might happen that LaTeX creates some special pages which only contains floats. This will usually be the case if there are many floats compared to the amount of text.

AutoCorrect options

Ignore hard page breaks

Hard (or manual) page breaks are often used to optimize page breaks in the final editing of a document. In this case you will probably not want to export these page breaks, as LaTeX creates page breaks that are quite different from the page breaks in Writer. If you select this option, hard page breaks will be ignored when exporting the document.

Ignore hard line breaks

For similar reasons, you can select this option to ignore hard (manual) line breaks during export.

Ignore empty paragraphs

Empty paragraphs are sometimes used a simple means to create vertical spacing in Writer. In a well-structured document, an empty paragraph is probably a mistake. Hence you can select this option to ignore empty paragraphs in the document in the export. If you do not select the option, an empty paragraph is exported as vertical space.

Ignore double spaces

For similar reasons you can choose to ignore two or more spaces in a row using this option.

2.4 Using Writer2BibTeX

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.

2.5 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 (modern) browser, but note that mathematical formulas are *not* supported.
- **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.

- **XHTML 1.1 + MathML 2.0 (xsl)** will create a similar xhtml file, but using some XSL-transformations provided by the World Wide Web Consortium (W3C), the result will be viewable by a wider range of browsers, such as Internet Explorer with the MathPlayer plugin. See `http://www.w3.org/Math/XSL/` for details.

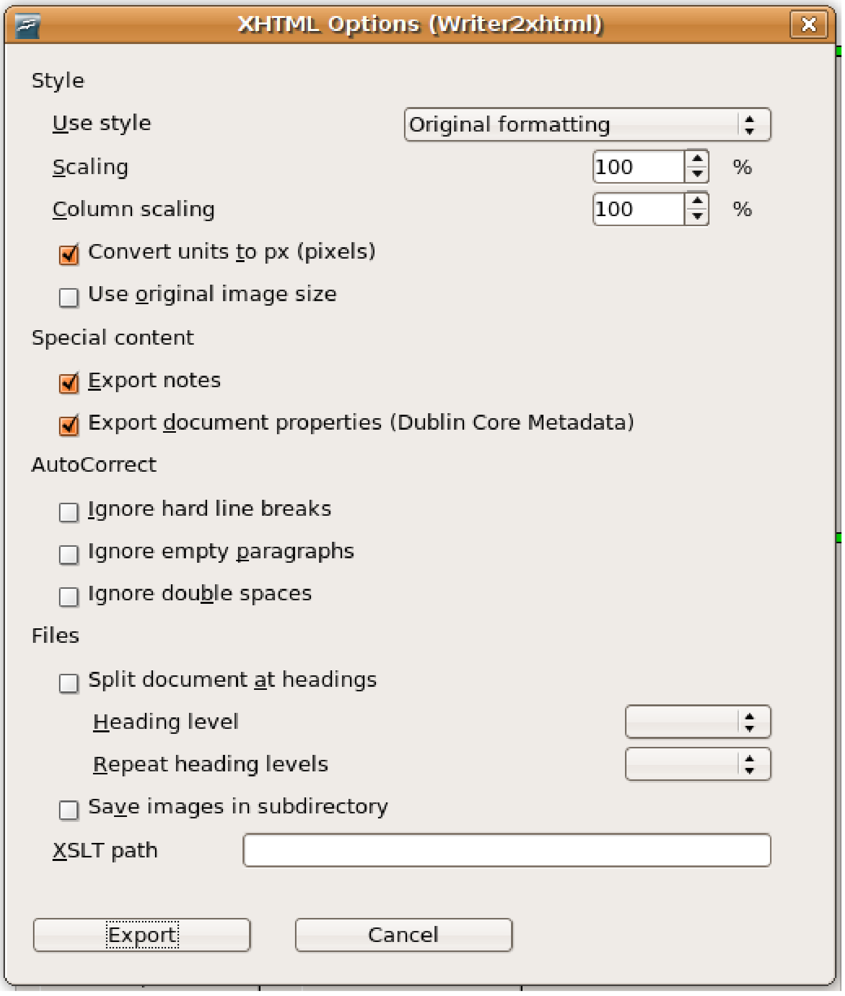
  This is how W3C's Math Working Group recommends to put ”math on the web”.

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

After you have typed in a file name, an options dialog will open:


XHTML Options (Writer2xhtml)

Style

Use style: Original formatting

Scaling: 100 %

Column scaling: 100 %

[✓] Convert units to px (pixels)

[ ] Use original image size

Special content

[✓] Export notes

[✓] Export document properties (Dublin Core Metadata)

AutoCorrect

[ ] Ignore hard line breaks

[ ] Ignore empty paragraphs

[ ] Ignore double spaces

Files

[ ] Split document at headings

Heading level

Repeat heading levels

[ ] Save images in subdirectory

XSLT path

Export Cancel

The individual options are explained below. Click **Export** to initiate the export or **Cancel** to close the dialog without exporting the document.

Style options

Use style

This option allows you to choose between various styles to apply to the xhtml document.

- **Original formatting** produces an xhtml document which uses the same style as the original Writer document. The document will look quite similar to the original when viewed in a browser.
- **Chocolate**, **Midnight**, **Modern**, **Oldstyle**, **Steely**, **Swiss**, **Traditional** and **Ultramarine** formats the document with one of the 8 *core styles* provided by the World Wide Web Consortium, see `http://www.w3.org/StyleSheets/Core/`.
- **Custom** is a user defined format. You can define your own style by providing a CSS style sheet and a mapping from Writer styles to your CSS styles. See section 2.7 for details on this.

Advanced users can extend the list with further styles using *configuration packages*, see section

2.8.

Scaling

Viewing the document in a web browser may require different dimensions (e.g. font sizes) than the original Writer document. Using this option you can define a percentage used to scale all dimensions, thus with the setting 140, all dimensions will be 40% larger than in the Writer document. Depending on the style you have selected and on the option *Use original image size*, some dimensions may be unaffected by this option.

Column scaling

This is a similar option, which only affects tables. Thus you can further widen or narrow the columns of tables if you wish.

Convert units to px (pixels)

In Writer, font sizes are usually given in *points* and other dimensions in e.g. *cm* or *inches*. For xhtml it is recommended to use the unit *px* instead, and using this option you can require that all dimensions are converted to px. If you choose not to check this option,the original units will always be used.

Use 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 xhtml document, but if you select this option, the original (unscaled) image size will be used.

Options for special content

Export notes

If you select this option, notes (also known as annotations) in the Writer document are exported as comments in the xhtml document. They will not be directly visible in the browser, only in the xhtml source. If the option is not selected, notes are completely ignored.

Export document properties (Dublin Core Metadata)

If you select this option, the document properties (**File – Properties**) are exported using the *Dublin Core standard*. See `http://dublincore.org/` for details on this.

AutoCorrect options

Ignore hard line breaks

Sometimes hard (or manual) line breaks are used in Writer to optimize the placement of the line breaks. Since line breaking in a browser is completely different, you may want to ignore all hard line breaks by selecting this option.

Ignore empty paragraphs

Empty paragraphs are sometimes used a simple means to create vertical spacing in Writer. In a well-structured document, an empty paragraph is probably a mistake. Hence you can select this option to ignore empty paragraphs in the document in the export.

Ignore double spaces

For similar reasons you can choose to ignore two or more spaces in a row using this option.

File options

Split document at headings

To make a long Writer document easier to read in the browser, you can use this option to split the document in several small files. Writer2xhtml will add a simple navigation panel that lets you move between pages. The navigation links will be in the same language as the document (as defined under **Tools – Options – Language Settings – Languages**)[5]. Note that this option has no effect for headings inside tables.

Heading level

If you have chosen to split the document at headings, you can use this option to define at which level splitting should occur. For example 2 to split the document at all headings of level 1 or 2.

Repeat heading levels

To help the reader to identify the current position within the document, you can use this option to repeat the parent headings whenever the document is split. If you for example split at headings of level 3 and set this option to 2, the headings of level 1 and 2 will be repeated before the heading of level 3, providing precise information as to where in the document the section belongs.

Save images in subdirectory

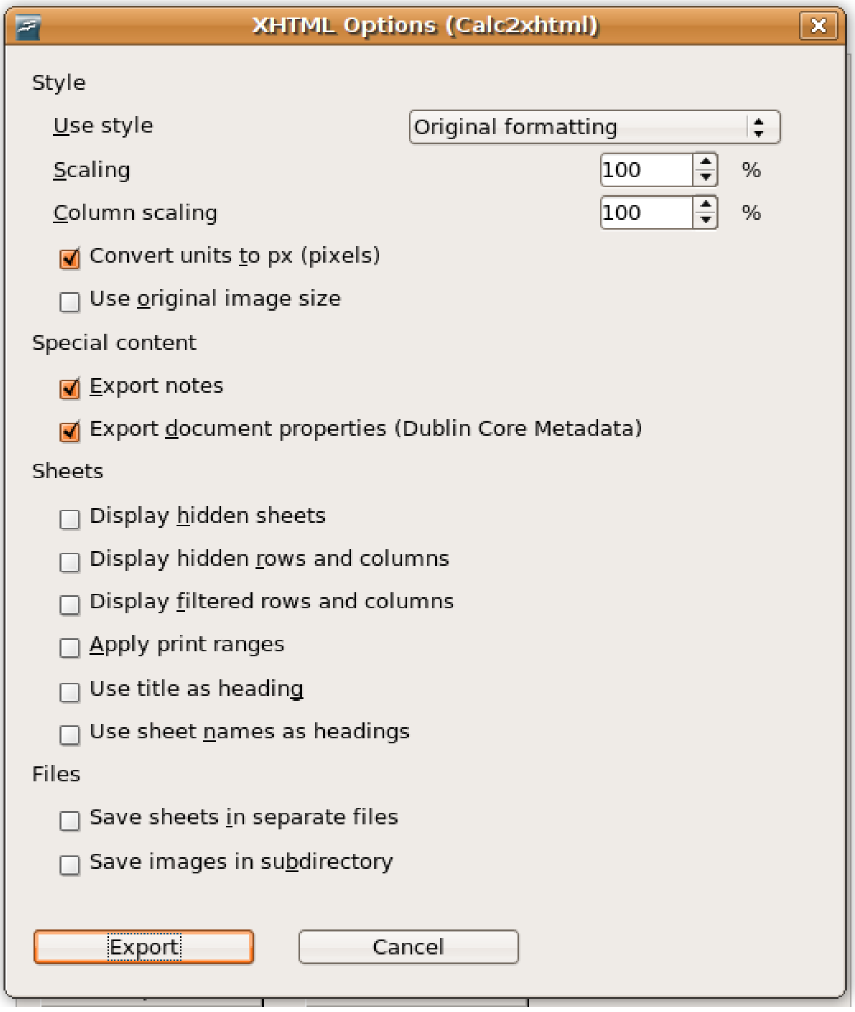
Writer2xhtml normally saves images associated with the document in the same directory as the xhtml document. If the document contains a large number of images it may be more convenient to save the images in a separate subdirectory. This option will create a subdirectory with the same name as the xhtml document to store the images.

2.6 Using Calc2xhtml

To export a Calc document to xhtml, choose **XHTML 1.0 strict** in the export dialog.

After you have typed in a file name, an options dialog will open:

[5]At the moment only a small number of languages are supported: English, Danish, German, Finnish, French, Spanish, Italian, Croatian, Russian and Ukrainian.



Style

Use style: Original formatting

Scaling: 100 %

Column scaling: 100 %

[✓] Convert units to px (pixels)

[ ] Use original image size

Special content

[✓] Export notes

[✓] Export document properties (Dublin Core Metadata)

Sheets

[ ] Display hidden sheets

[ ] Display hidden rows and columns

[ ] Display filtered rows and columns

[ ] Apply print ranges

[ ] Use title as heading

[ ] Use sheet names as headings

Files

[ ] Save sheets in separate files

[ ] Save images in subdirectory

The individual options are explained below (only where the differ from the options in Writer2xhtml). Click **Export** to initiate the export or **Cancel** to close the dialog without exporting the document.

Style options

Use style

This options works like in Writer2xhtml, except that the core styles from W3C are not displayed (as they are not suitable for table documents).

Sheet options

Display hidden sheets

If you have chosen to hide some sheets in Calc, you can select this option if you want to display them in the xhtml document anyway.

Display hidden rows and columns

The same applies, if you have chosen to hide some columns or rows in your spreadsheet.

Display filtered rows and columns

When you export the document, some rows or columns may be invisible because you have applied a *filter* in Calc. If you select this option, the invisible rows and columns will be exported to xhtml anyway.

Apply print ranges

If you check this option, the xhtml document will display the parts of the document which are selected for printing using *print ranges* in Calc. The display in the browser will thus be similar to what you get when you are *printing* the document from Calc. If the option is not checked, the result will instead resemble what you see when you *edit* the document in Calc.

Use title as heading

If you check this option, Calc2xhtml will insert the document title (**File – Properties – Description – Title**) as heading at the top of the xhtml document.

Use sheet names as headings

If you check this option, Calc2xhtml will insert the name of each sheet as a heading above the sheet in the xhtml document.

File options

Save sheets in separate files

If you select this option, Calc2xhtml will produce a separate file for each sheet, otherwise all sheets will be exported to the same xhtml file. In any case, a simple navigation panel showing all sheet names will be added.

2.7 Custom configuration

Each of the exports provides the possibility to use a custom format/style. Currently you have to manually edit a configuration file to define it. All three exporters uses a configuration file in the user installation folder for OOo.

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

  `home directory/.OpenOffice.org2/user`

  or

  `home directory/.OpenOffice.org/3/user`

- On Windows it will usually be something like

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

`C:\Documents and Settings\username\OpenOffice.org\3\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.

2.8 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 OOo 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 OOo template.
- An OOo 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 OOo 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 OOo 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.

The file description.xml

This files identifies the extension in OOo. 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>
```

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

These files should be left unchanged.

The folder template

Put your OOo 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).

The folder config

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

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>
                </node>
            </node>
```

The next section defines the OOo 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 OOo 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>
```

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	***Writer2xhtml***	***Calc2xhtml***
`backend`		
`inputencoding`		
`multilingual`	`scaling`	
`greek_math`	`column_scaling`	
`additional_symbols`[6]	`convert_to_px`	
`use_bibtex`	`original_image_size`	`scaling`
`bibtex_style`	`notes`	`column_scaling`
`wrap_lines_after`	`use_dublin_core`	`convert_to_px`
`split_linked_sections`	`display_hidden_sheets`	`original_image_size`
`split_toplevel_sections`	`display_hidden_rows`	`notes`
`save_images_in_subdir`	`_cols`	`use_dublin_core`
`notes`	`display_filtered_rows`	`ignore_hard_line_breaks`
`metadata`	`_cols`	`ignore_empty_paragraphs`
`original_image_size`	`apply_print_ranges`	`ignore_double_spaces`
`simple_table_limit`	`use_title_as_heading`	`split_level`
`float_tables`	`use_sheetnames_as`	`repeat_levels`
`float_figures`	`_headings`	`save_images_in_subdir`
`float_options`	`calc_split`	
`ignore_hard_page_breaks`	`save_images_in_subdir`	
`ignore_hard_line_breaks`	`xslt_path`	
`ignore_empty_paragraphs`		
`ignore_double_spaces`		

Writer2LaTeX	***Writer2xhtml***	***Calc2xhtml***

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

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 OOo 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.

Installation for Microsoft Windows

To install Writer2LaTeX under Microsoft Windows follow these instructions:

1. Unzip `writer2latex102.zip` into some directory. This will create a subdirectory `writer2latex10`.
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:\writer2latex10"
   ```

   (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:\Program Files\java\j2sdk1.5.0_22\bin\java"
   ```

Installation for Unix and friends

1. Unzip `writer2latex102.zip` into some directory. This will create a subdirectory `writer2latex10`.
2. Add this directory to your PATH environment variable.
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/writer2latex10"
```

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
	`-xhtml+mathml`	Convert to xhtml + MathML
	`-xhtml+mathml+xsl`	Convert to xhtml + MathML with xsl (see section 2.5)
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
	`-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.

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
```

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.

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
```

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.

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. The support for XeTeX is currently incomplete. More comprehensive support for XeTeX is planned for the next version of Writer2LaTeX (version 1.2).
`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 `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 (default).
`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.
`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 `ooomath.sty`. This package defines number of LaTeX macros used to convert formulas from OOo 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 `ooomath.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.

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.
`external_bibtex_files`	Set this option to `true` if the bibliographic references in the document should be interpreted as keys in one or more external BibTeX files. If set to `false` (default), the bibliographic references will be exported to a BibTeX file (provided `use_bibtex` is set to `true`).

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.

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.

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`.
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 included in a `center` environment. 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 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).
`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 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 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`.

These options controls the export of tables:

`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[7] 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).

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.
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 \ .

Formatting 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 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.

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

`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.

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.

Headings

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).

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`.

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.

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}" />
```

Math symbols

In OOo Math you can add user-defined symbols. Writer2LaTeX already understands the pre-defined 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.

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.

Style options

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

`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.
`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`).
`custom-stylesheet`	Use this options 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[8]. 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 OOo does not offer section styles currently).
`table_formatting`	Used for the same purpose for table formatting. (But note that OOo does not offer table styles currently).
`ignore_table_dimensions`	Set this option to `true` if 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. Default is `false`.
`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_list_hack`	This option is used to fix a problem with continued lists. If you set this options to `true` (default is `false`), 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.

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.
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%`.
natural_image_size[9]	Use this option to specify that the size of images should not be exported, hence LaTeX should use the original size of the image. Default is `false`.

Options for special content

use_dublin_core	Use this option 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 (default is `true`).

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

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

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`.

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 OOo).

File options

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.
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).
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"`.

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.

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.

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" 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 OOo 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`[10] specifies the style family 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" family="paragraph"
           block-element="div" block-css="block_style"
           element="p" css="par_style" />
```

[10]Previously this attributed was called class.

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 `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.

4.3 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 style family***	***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`

OOo Writer style	***OOo 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.

5 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`[11] 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 and 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 (you can create it if it doesn't exist).

Next you should type

```
texconfig rehash
```

to make TeX 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.

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

6 Using Writer2LaTeX from another application

6.1 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
Enumeration docs = dataOut.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`.

6.2 Using Writer2LaTeX from a Basic macro

You can also access Writer2LaTeX through OOo'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. 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).

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 (spreadsheet)	`org.openoffice.da.calc2xhtml`
xhtml + MathML	`org.openoffice.da.writer2xhtml.mathml`
xhtml + MathML using xsl	`org.openoffice.da.writer2xhtml.mathml.xsl`

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

http://api.openoffice.org/docs/common/ref/com/sun/star/util/PathSubstitution.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 `sConfigUrl = "*ultraclean.xml"`.

6.3 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 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 `PropertyValues` 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 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 `PropertyValues` to pass options to the filter (see above).

6.4 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()
```

7 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).

If you are using Writer2LaTeX as an export filter in OOo, 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 – OpenOffice.org – Java**. Click **Parameters**, and add the parameter `-Xmx128M` (or higher).

A few memory optimizations are planned for the next version (1.2), which should make it possible to convert a wider range of documents without increasing the heap size in java.