

OpenDocument Text Exporter for Emacs' Org Mode

Jambunathan K

1 Project Summary

Description

The Authoritative fork of Org mode's ODT exporter

URL <https://github.com/kjambunathan/org-mode-ox-odt>

Version

Depends on

Suggests

Enhances

Published

Author Jambunathan K <kjambuanthan at gmail.com>

Maintainer

Jambunathan K <kjambuanthan at gmail.com>

BugReports

<https://github.com/kjambunathan/org-mode-ox-odt/issues>

License

Downloads

ELPA URL

<https://kjambunathan.github.io/elpa/>

User Manual (online HTML)

<https://kjambunathan.github.io/org-mode-ox-odt/>

2 Structure of ELPA tar

```
./          ox-jabref.el
           ox-odt-pkg.el
           ox-odt.el

etc/styles/
           OrgOdtContentTemplate.xml
           OrgOdtStyles.xml
           README

LibreOffice
           LibreOffice/OrgModeUtilities.oxt

docs/      index.html
           org-odt
           org-odt.pdf
           style.css

etc/schema/
           od-manifest-schema.rnc
           od-schema.rnc
           schemas.xml

etc/schema/libreoffice/
           OpenDocument-dsig-schema-v1.3+libreoffice.rnc
           OpenDocument-manifest-schema-v1.3+libreoffice.rnc
           OpenDocument-schema-v1.3+libreoffice.rnc
           OpenDocument-schema-v1.3.rnc

etc/schema/odf1.0/
           OpenDocument-manifest-schema-v1.0-os.rnc
           OpenDocument-schema-v1.0-os.rnc
           OpenDocument-strict-schema-v1.0-os.rnc

etc/schema/odf1.1/
           OpenDocument-manifest-schema-v1.1.rnc
           OpenDocument-schema-v1.1.rnc
           OpenDocument-strict-schema-v1.1.rnc

etc/schema/odf1.2/
           OpenDocument-v1.2-os-dsig-schema.rnc
           OpenDocument-v1.2-os-manifest-schema.rnc
           OpenDocument-v1.2-os-schema.rnc
```

```
etc/schema/odf1.3/  
    OpenDocument-dsig-schema-v1.3.rnc  
    OpenDocument-manifest-schema-v1.3.rnc  
    OpenDocument-schema-v1.3.rnc
```

```
samples/biblatex-examples/  
    README.org  
    biblatex-examples.bib
```

```
samples/    masterdocument.org  
            org-mode-unicorn.png  
            pagebreak.odt  
            pagebreak.org  
            subdocument1.odt  
            subdocument1.org
```

3 Getting Started with ODT export

3.1 Pre-requisites for ODT export

The ODT backend depends on the following programs

Program	Purpose
<code>'zip'</code> ¹	To produce OpenDocument files
<code>'unzip'</code> ²	To unzip custom styles
<code>'identify'</code> ³	To identify the size of an inline image
<code>'latex'</code> ⁴	To compile L ^A T _E X fragments to <code>'dvi'</code> images
<code>'dvisgm'</code> ⁵	To convert <code>'dvi'</code> images to <code>'svg'</code>
<code>'dvipng'</code> ⁶	To convert <code>'dvi'</code> images to <code>'png'</code>
<code>'convert'</code> ⁷	To convert inline <code>'pdf'</code> to <code>'png'</code>
<code>'latexmlmath'</code> ⁸	To convert L ^A T _E X math snippets to MathML
<code>'mathtoweb'</code> ⁹	
<code>'jabref'</code> ¹⁰	To handle bibliography and citations

Of these, `'zip'` is essential. Rest are optional.

3.2 Installation

You can install the OpenDocument Text export backend using the Emacs package manager. The archive URL for the package is <https://kjambunathan.github.io/elpa/>.

A typical configuration look like this

```
(custom-set-variables
 '(package-archives
  (quote
   (("gnu" . "https://elpa.gnu.org/packages/")
    ("ox-odt" . "https://kjambunathan.github.io/elpa/")))))
```

In the `'*Packages*'` buffer, packages from this archive show up as below

```
JabrefExportChicagoODF 1.2.2      ... Jabref Plugin for export to Chicago Manual of St
ox-odt                    9.2.6.263 ... OpenDocument Text Exporter for Org Mode
```

¹ Info-ZIP
² Info-ZIP
³ ImageMagick
⁴ T_EX Live
⁵ T_EX Live
⁶ dvipng
⁷ ImageMagick
⁸ L^AT_EXML
⁹ MathToWeb
¹⁰ JabRef

3.3 Configuration

Here is a sample configuration.

```
(custom-set-variables
 '(org-odt-convert-process "LibreOffice")
 '(org-odt-preferred-output-format "docx")
 '(org-odt-transform-processes
  '(("Optimize Column Width of all Tables"
    "soffice" "--norestore" "--invisible" "--headless"
    "macro:///OrgMode.Utilities.OptimizeColumnWidth(%I)")
    ("Update All"
     "soffice" "--norestore" "--invisible" "--headless"
     "macro:///OrgMode.Utilities.UpdateAll(%I)")
    ("Reload"
     "soffice" "--norestore" "--invisible" "--headless"
     "macro:///OrgMode.Utilities.Reload(%I)")))
 '(org-jabref-command '("/opt/jabref/bin/JabRef" "-n"))
 '(org-latex-to-mathml-convert-command
  "java -jar %j -unicode -force -df %o %I")
 '(org-latex-to-mathml-jar-file
  "/home/kjambunathan/Downloads/mathtoweb.jar"))

(require 'ox-odt)
(require 'ox-jabref)

;; (setcdr (assq 'system org-file-apps-defaults-gnu) "xdg-open %s")
(setcdr (assq 'system org-file-apps-gnu) "xdg-open %s")

(advice-add 'org-open-file :around
  (lambda (orig-fun &rest args)
    ;; Work around a weird problem with xdg-open.
    (let ((process-connection-type nil))
      (apply orig-fun args)))))
```

Above configuration sets up the ODT backend as follows:

1. Use "LibreOffice" (i.e., 'soffice' executable) as the document converter
2. Generate a 'docx' document for every 'odt' document
3. Process the 'odt' document with a set of LibreOffice Basic Macros to
 - Optimize the column width of all tables
 - Update cross-references, table of contents etc.
 - (if you are already viewing a past version of a 'odt' file), re-load the new file in the same application window.
4. Tell where your 'JabRef' and 'mathtoweb' executables are located, and how they are invoked.
5. (if you are using a GNU system) open the 'odt' document with your preferred Open-Document viewer, presumably 'LibreOffice'.

6. Load ‘`ox-jabref`’ so as to produce documents with bibliography and citations .

4 ODT export commands

4.1 Exporting to ODT

C-c C-e o o ('org-odt-export-to-odt')

Export as OpenDocument Text file.

If 'org-odt-preferred-output-format' is specified, automatically convert the exported file to that format. See [Section 5.1 \[Automatically exporting to other formats\]](#), page 8.

C-c C-e o O

Export as OpenDocument Text file and open the resulting file.

If 'org-odt-preferred-output-format' is specified, open the converted file instead. See [Section 5.1 \[Automatically exporting to other formats\]](#), page 8.

5 Extending ODT export

The ODT exporter can interface with a variety of document converters and supports popular converters out of the box. As a result, you can use it to export to formats like ‘doc’ or convert a document from one format (say ‘csv’) to another format (say ‘ods’ or ‘xls’).

If you have a working installation of LibreOffice, a document converter is pre-configured for you and you can use it right away. If you would like to use ‘unoconv’ as your preferred converter, customize the variable ‘org-odt-convert-process’ to point to ‘unoconv’. You can also use your own favorite converter or tweak the default settings of the LibreOffice and ‘unoconv’ converters. See [Section 15.1 \[Configuring a document converter\]](#), page 30.

5.1 Automatically exporting to other formats

Very often, you will find yourself exporting to ODT format, only to immediately save the exported document to other formats like ‘doc’, ‘docx’, ‘rtf’, ‘pdf’ etc. In such cases, you can specify your preferred output format by customizing the variable ‘org-odt-preferred-output-format’. This way, the export commands (see [Section 4.1 \[Exporting to ODT\]](#), page 7) can be extended to export to a format that is of immediate interest to you.

5.2 Converting between document formats

There are many document converters in the wild which support conversion to and from various file formats, including, but not limited to the ODT format. LibreOffice converter, mentioned above, is one such converter. Once a converter is configured, you can interact with it using the following command.

M-x org-odt-convert

Convert an existing document from one format to another. With a prefix argument, also open the newly produced file.

6 Applying custom styles

The ODT exporter ships with a set of OpenDocument styles (see [Section 15.2 \[Working with OpenDocument style files\]](#), page 30) that ensure a well-formatted output. These factory styles, however, may not cater to your specific tastes. To customize the output, you can either modify the above styles files directly, or generate the required styles using an application like LibreOffice. The latter method is suitable for expert and non-expert users alike, and is described here.

6.1 Applying custom styles - the easy way

1. Create a sample ‘example.org’ file with the below settings and export it to ODT format.

```
#+OPTIONS: H:10 num:t
```

2. Open the above ‘example.odt’ using LibreOffice. Use the Stylist to locate the target styles - these typically have the ‘Org’ prefix - and modify those to your taste. Save the modified file either as an OpenDocument Text (‘.odt’) or OpenDocument Template (‘.ott’) file.
3. Customize the variable ‘org-odt-styles-file’ and point it to the newly created file. For additional configuration options see [Section 15.2.2 \[Overriding factory styles\]](#), page 31.

If you would like to choose a style on a per-file basis, you can use the ‘#+ODT_STYLES_FILE’ option. A typical setting will look like

```
#+ODT_STYLES_FILE: "/path/to/example.ott"
```

or

```
#+ODT_STYLES_FILE: ("/path/to/file.ott" ("styles.xml" "image/hdr.png"))
```

6.2 Using third-party styles and templates

You can use third-party styles and templates for customizing your output. This will produce the desired output only if the template provides all style names that the ‘ODT’ exporter relies on. Unless this condition is met, the output is going to be less than satisfactory. So it is highly recommended that you only work with templates that are directly derived from the factory settings.

7 Links in ODT export

ODT exporter creates native cross-references for internal links. It creates Internet-style links for all other links.

A link with no description and destined to a regular (un-itemized) outline heading is replaced with a cross-reference and section number of the heading.

A `\ref{label}`-style reference to an image, table etc. is replaced with a cross-reference and sequence number of the labeled entity. See [Chapter 12 \[Labels and captions in ODT export\]](#), page 21.

8 List Tables in ODT export

A ‘List Table’, in simple terms, is a list that is typeset as a table. Use it to create tables with multi-paragraph content.

Why a ‘List Table’?

Org mode’s tables are line-oriented i.e., each row (and hence a cell) cannot span multiple lines. This choice has serious limitations. Specifically,

- a table cell cannot have more than a single paragraph
- if the sole paragraph has copious text, the table will overflow your display screen, and editing or reviewing such tables is very cumbersome and annoying.

A list table overcomes the above problem.

Lists that are marked with attribute ‘:list-table’ are called as list tables. They will be rendered as a table within the exported document.

8.1 Examples

A simple list table

Consider an example. The following list table

```
#+ATTR_ODT: :rel-width 80
#+ATTR_ODT: :list-table t
-
- Row 1, Col 1
- Row 1, Col 2
- Row 1, Col 3
- Row 1, Col 4
- -----
- Row 2, Col 1
- Row 2, Col 2
- Row 2, Col 3
- Row 2, Col 4
```

will be exported as though it were an Org table like the one show below.

```
| Row 1, Col 1 | Row 1, Col 2 | Row 1, Col 3 | Row 1, Col 4 |
| Row 2, Col 1 | Row 2, Col 2 | Row 2, Col 3 | Row 2, Col 4 |
```

Row 1, Col 1	Row 1, Col 2	Row 1, Col 3	Row 1, Col 4
Row 2, Col 1	Row 2, Col 2	Row 2, Col 3	Row 2, Col 4

8.2 A list table with rules, column size and alignments

List tables honor all attributes save for the column alignments.

```
#+ATTR_ODT: :list-table t
- | /      | <      | >      |      |
- | <12>   | <11>   | <11>   | <18>   |
-
  - Row 1, Col 1
  - Row 1, Col 2
  - Row 1, Col 3
  - Row 1, Col 4
- -----
  - Row 2, Col 1
  - Row 2, Col 2
  - Row 2, Col 3
  - Row 2, Col 4
```

Above list table will be exported as if it were an Org table like the one below

```
| /      | <      | >      |      |
| <12>   | <11>   | <11>   | <18>   |
| Row 1, Col 1 | Row 1, Col 2 | Row 1, Col 3 | Row 1, Col 4 |
|-----+-----+-----+-----|
| Row 2, Col 1 | Row 2, Col 2 | Row 2, Col 3 | Row 2, Col 4 |
```

Row 1, Col 1	Row 1, Col 2	Row 1, Col 3	Row 1, Col 4
Row 2, Col 1	Row 2, Col 2	Row 2, Col 3	Row 2, Col 4

Here is a real-world table that uses the same template as the one above

```
#+ATTR_ODT: :list-table t
- | /      | <      | >      |      |
- | <12>   | <11>   | <11>   | <18>   |
-
  - Day
  - Min Temp
  - Max Temp
  - Summary
- -----
  - Monday
  - 11C
```

```

- 22C
-
  1. A clear day with lots of sunshine.
  2. Late in the day, a strong breeze will bring down the temperatures.
-----
- Tuesday
- 9C
- 19C
-
  1. Cloudy with rain, across many northern regions.
  2. Clear spells across most of Scotland and Northern Ireland, but
    rain reaching the far northwest.

```

Day	Min Temp	Max Temp	Summary
Monday	11C	22C	1. A clear day with lots of sunshine. 2. Late in the day, a strong breeze will bring down the temperatures.
Tuesday	9C	19C	1. Cloudy with rain, across many northern regions. 2. Clear spells across most of Scotland and Northern Ireland, but rain reaching the far northwest.

8.3 List table with table and figures

List tables are particularly well-suited for creating captioned side-by-side images. Note that with conventional ‘org’ tables, even though you can place images side-by-side using a table, you cannot caption them.

```

#+ATTR_ODT: :list-table t
-
-
  #+CAPTION: A Unicorn
  [[./org-mode-unicorn.png]]
-
  #+CAPTION: Yet another Unicorn

```

```
[[./org-mode-unicorn.png]]
```



(a) A Unicorn



(b) Yet another Unicorn

8.4 List table can contain other tables


```
#+ATTR_ODT: :list-table t
-
- Comments
- A Dog & A Cat
- -----
- This table contains
- A Dog
- A cat

| Animal | What it does |
|-----+-----|
| Dog    | Barks        |
| Cat    | News         |

-

#+ATTR_ODT: :list-table t
-
-
  #+CAPTION: A Dog
  #+ATTR_ODT: :scale .8
  [[./org-mode-unicorn.png]]
-
  #+CAPTION: A Cat
  #+ATTR_ODT: :scale .8
  [[./org-mode-unicorn.png]]
```

- - - - -

Comments		A Dog & A Cat	
This table contains			
<ul style="list-style-type: none">A DogA cat			
Animal	What it does		
Dog	Barks	(a) A Dog	(b) A Cat
Cat	News		

9 Tables in ODT export

Export of native Org mode tables (See [Section “Tables” in org](#)) and simple ‘`table.el`’ tables is supported. However, export of complex ‘`table.el`’ tables - tables that have column or row spans - is not supported. Such tables are stripped from the exported document.

By default, a table is exported with top and bottom frames and with rules separating row and column groups (See [Section “Column Groups” in org](#)). Furthermore, all tables are typeset to occupy the same width. If the table specifies alignment and relative width for its columns (See [Section “Column Width and Alignment” in org](#)) then these are honored on export.¹

You can control the width of the table by specifying ‘`:rel-width`’ property using an ‘`#+ATTR_ODT`’ line.

For example, consider the following table which makes use of all the rules mentioned above.

```
#+ATTR_ODT: :rel-width 50
```

Area/Month	Jan	Feb	Mar	Sum
/	<			<
<l13>	<r5>	<r5>	<r5>	<r6>
North America	1	21	926	948
Middle East	6	75	844	925
Asia Pacific	9	27	790	826
Sum	16	123	2560	2699

On export, the table will occupy 50% of text area. The columns will be sized (roughly) in the ratio of 13:5:5:5:6. The first column will be left-aligned and rest of the columns will be right-aligned. There will be vertical rules after separating the header and last columns from other columns. There will be horizontal rules separating the header and last rows from other rows.

If you are not satisfied with the above formatting options, you can create custom table styles and associate them with a table using the ‘`#+ATTR_ODT`’ line. See [Section 15.4 \[Customizing tables in ODT export\]](#), page 32.

¹ See [MathToWeb](#)

10 Images in ODT export

10.1 Embedding images

You can embed images within the exported document by providing a link to the desired image file with no link description. For example, to embed ‘img.png’ do either of the following:

```
[[file:img.png]]
[[./img.png]]
```

10.2 Embedding clickable images

You can create clickable images by providing a link whose description is a link to an image file. For example, to embed a image org-mode-unicorn.png which when clicked jumps to <http://Orgmode.org> website, do the following

```
[[http://orgmode.org][./org-mode-unicorn.png]]
```

10.3 Sizing and scaling of embedded images

You can control the size and scale of the embedded images using the ‘#+ATTR_ODT’ attribute.

The exporter specifies the desired size of the image in the final document in units of centimeters. In order to scale the embedded images, the exporter queries for pixel dimensions of the images using one of a) ImageMagick’s identify program or b) Emacs ‘create-image’ and ‘image-size’ APIs.¹ The pixel dimensions are subsequently converted in to units of centimeters using ‘org-odt-pixels-per-inch’. The default value of this variable is set to ‘display-pixels-per-inch’. You can tweak this variable to achieve the best results.

The examples below illustrate the various possibilities.

Explicitly size the image

To embed ‘img.png’ as a 10 cm x 10 cm image, do the following:

```
#+ATTR_ODT: :width 10 :height 10
[[./img.png]]
```

Scale the image

To embed ‘img.png’ at half its size, do the following:

```
#+ATTR_ODT: :scale 0.5
[[./img.png]]
```

Scale the image to a specific width

To embed ‘img.png’ with a width of 10 cm while retaining the original height:width ratio, do the following:

```
#+ATTR_ODT: :width 10
[[./img.png]]
```

¹ Use of ImageMagick is only desirable. However, if you routinely produce documents that have large images or you export your Org files that has images using a Emacs batch script, then the use of ImageMagick is mandatory.

Scale the image to a specific height

To embed ‘img.png’ with a height of 10 cm while retaining the original height:width ratio, do the following

```
#+ATTR_ODT: :height 10
[[./img.png]]
```

10.4 Anchoring of images

You can control the manner in which an image is anchored by setting the ‘:anchor’ property of it’s ‘#+ATTR_ODT’ line. You can specify one of the the following three values for the ‘:anchor’ property - ‘as-char’, ‘paragraph’ and ‘page’.

To create an image that is anchored to a page, do the following:

```
#+ATTR_ODT: :anchor "page"
[[./img.png]]
```

11 Math formatting in ODT export

The ODT exporter has special support for handling math.

11.1 Working with \LaTeX math snippets

\LaTeX math snippets (See [Section “ \$\text{\LaTeX}\$ fragments” in \[org\]\(#\)](#)) can be embedded in the ODT document in one of the following ways:

1. MathML

This option is activated on a per-file basis with

```
#+OPTIONS: LaTeX:t
```

With this option, \LaTeX fragments are first converted into MathML fragments using an external \LaTeX -to-MathML converter program. The resulting MathML fragments are then embedded as an OpenDocument Formula in the exported document.

You can specify the \LaTeX -to-MathML converter by customizing the variables ‘`org-latex-to-mathml-convert-command`’ and ‘`org-latex-to-mathml-jar-file`’.

If you prefer to use MathToWeb¹ as your converter, you can configure the above variables as shown below.

```
(setq org-latex-to-mathml-convert-command
      "java -jar %j -unicode -force -df %o %I"
      org-latex-to-mathml-jar-file
      "/path/to/mathtoweb.jar")
```

You can use the following commands to quickly verify the reliability of the \LaTeX -to-MathML converter.

M-x org-export-as-odf

Convert a \LaTeX math snippet to OpenDocument formula (‘.odf’) file.

M-x org-export-as-odf-and-open

Convert a \LaTeX math snippet to OpenDocument formula (‘.odf’) file and open the formula file with the system-registered application.

2. PNG images

This option is activated on a per-file basis with

```
#+OPTIONS: LaTeX:dvipng
```

With this option, \LaTeX fragments are processed into PNG images and the resulting images are embedded in the exported document. This method requires that the dvipng program be available on your system.

11.2 Working with MathML or OpenDocument formula files

For various reasons, you may find embedding \LaTeX math snippets in an ODT document less than reliable. In that case, you can embed a math equation by linking to its MathML (‘.mml’) source or its OpenDocument formula (‘.odf’) file as shown below:

¹ See [MathToWeb](#)

```
[[./equation.mml]]  
or  
[[./equation.odf]]
```

12 Labels and captions in ODT export

You can label and caption various category of objects - an inline image, a table, a \LaTeX fragment or a Math formula - using ‘`#+LABEL`’ and ‘`#+CAPTION`’ lines. See [Section “File Archives” in `emacs`](#). ODT exporter enumerates each labeled or captioned object of a given category separately. As a result, each such object is assigned a sequence number based on order of it’s appearance in the Org file.

In the exported document, a user-provided caption is augmented with the category and sequence number. Consider the following inline image in an Org file.

```
#+CAPTION: Bell curve
#+LABEL:   fig:SED-HR4049
[[./img/a.png]]
```

It could be rendered as shown below in the exported document.

Figure 2: Bell curve

You can modify the category component of the caption by customizing the variable ‘`org-odt-category-strings`’. For example, to tag all embedded images with the string ‘`Illustration`’ (instead of the default ‘`Figure`’) use the following setting.

```
(setq org-odt-category-strings
      '(("en" "Table" "Illustration" "Equation" "Equation")))
```

With this, previous image will be captioned as below in the exported document.

Illustration 2: Bell curve

13 Literal examples in ODT export

Export of literal examples (See [Section “Literal examples” in org](#)) with full fontification is supported. Internally, the exporter relies on `htmlfontify.el` to generate all style definitions needed for a fancy listing.¹ The auto-generated styles have `OrgSrc` as prefix and inherit their color from the faces used by Emacs `font-lock` library for the source language.

If you prefer to use your own custom styles for fontification, you can do so by customizing the variable `org-odt-create-custom-styles-for-srcblocks`.

You can turn off fontification of literal examples by customizing the variable `org-odt-fontify-srcblocks`.

¹ Your `htmlfontify.el` library must at least be at Emacs 24.1 levels for fontification to be turned on.

14 Bibliography and Citations in ODT export

The ODT export back-end uses JabRef to produce Bibliography and Citations.

‘org’ doesn’t have a *standard* markup for bibliography and citation references. This is true for this export backend as well. So, the syntax described in the next section is *specific* to the ODT backend, and doesn’t carry over to other backends.

Bibliography and Citation-specific Keywords in ODT export

An ‘org’ file with bibliography and citations look like

```
#+BIB_FILE: "novices.bib"
#+ODT_JABREF_CITATION_STYLE: "Numeric"
```

Some text content

```
#+BIBLIOGRAPHY:
```

‘BIB_FILE’

Path to the bibliography file

‘ODT_JABREF_CITATION_STYLE’

Citation style to use. You can choose one of the following options

- ‘"Numeric"’
- ‘"Chicago (full-note)"’
- ‘"Chicago (author-date)"’

‘BIBLIOGRAPHY’ -

Bibliography is inserted here.

How to cite?

The ODT backend recognizes following citation references

‘\cite{ }’ snippets

Pandoc’s Berkeley-style Citations¹

The semi-official Org-mode citation syntax was designed by Richard Lawrence with additions by contributors on the emacs-orgmode mailing list. It is based on John MacFarlane’s pandoc Markdown syntax. It’s dubbed Berkeley syntax due the place of activity of its creators, both philosophers at UC Berkeley.

Simple in-text citation

This is the simplest form of citation. It consists of the citation ID prefixed by ‘@’.

Example

@WatsonCrick1953 showed that DNA forms a double helix.

In-text citation list

Citations presented in the text unparenthesized are called in-text citations. The syntax for these citations is

¹ See section titled *Berkeley-style citations* at <https://pandoc.org/org.html>

[cite: =PREFIX=; =INDIVIDUAL-REFERENCE=; ... =INDIVIDUAL

where the initial ‘PREFIX’ and final ‘SUFFIX’ are optional. At least one ‘INDIVIDUAL-REFERENCE’ must be present. The colon and semicolons here are literal and indicate the end of the ‘TAG’ and the end of a ‘PREFIX’ or ‘INDIVIDUAL-REFERENCE’ respectively.

An ‘INDIVIDUAL-REFERENCE’ has the format:

‘PREFIX’ ‘KEY’ ‘SUFFIX’

The ‘KEY’ is obligatory, and the prefix and suffix are optional.

A ‘PREFIX’ or ‘SUFFIX’ is arbitrary text (except ;,], and citation keys).

Example

[cite: See; @Mandelkern1981; and @Watson1953]

Parentetical citation

Citations surrounded by parantheses. The syntax is identical to in-text citations, except for the additional parentheses enclosing the initial cite tag.

Example

[(cite): See; @Mandelkern1981; and @Watson1953]

14.1 Configure JabRef with ‘Chicago.ODF’ custom export

Set up JabRef with ‘Chicago.ODF’ custom export

Install Jabref².

Navigate to ‘package-user-dir’, and locate the ‘JabrefExportChicagoODF-1.2.2.tar’³. Extract it to get a set of ‘layout’ files.

Launch JabRef GUI. Navigate to ‘Options -> Manage custom exports’, and add an entry for each of the layout files as below.

Export name	Main layout file	Extension
Chicago.ODF.abstract	<whatever>/Chicago.ODF.abstract.layout	xml
Chicago.ODF.biblio	<whatever>/Chicago.ODF.biblio.layout	xml
Chicago.ODF.footend	<whatever>/Chicago.ODF.footend.layout	xml
Chicago.ODF.footend.short	<whatever>/Chicago.ODF.footend.short.layout	xml
Chicago.ODF.note	<whatever>/Chicago.ODF.note.layout	xml
Chicago.ODF.reference	<whatever>/Chicago.ODF.reference.layout	xml
Chicago.ODF.text	<whatever>/Chicago.ODF.text.layout	xml

Table 14.1: How to setup JabRef-5.0’s ‘Manage custom exports’

² [L^AT_EX XML](#)

³ See Jabref Plugin for export to Chicago Manual of Style in [OpenDocument](#) format. This a port of Jabref’s Plugin for export to Chicago Manual of Style in [RTF](#) format.

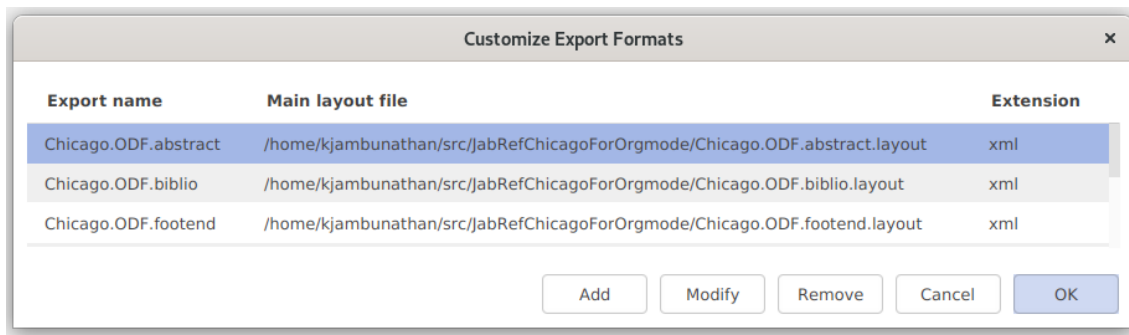


Figure: Screenshot of JabRef-5.0's 'Manage custom exports'

Verify that the Chicago ODF plugin is successfully registered. In a terminal, do

```
/opt/jabref/bin//JabRef -n -h
```

Ensure that the 'Available export formats' mentions the 'Chicago.ODF'.

```
Available export formats: html, simplehtml, docbook5, docbook4, din1505,
bibordf, tablerefs, listrefs, tablerefsabsbib, harvard, iso690rtf,
iso690txt, endnote, oocsv, ris, misq, bibtexml, oocalc, ods, MSBib,
mods, xmp, pdf, Chicago.ODF.abstract, Chicago.ODF.biblio,
Chicago.ODF.footend, Chicago.ODF.footend.short, Chicago.ODF.note,
Chicago.ODF.reference, Chicago.ODF.text
```

Configure Emacs

Tell Emacs about 'JabRef's executable.

Use this if you have 'JabRef' executable.

```
(custom-set-variables
 '(org-jabref-command '("/opt/jabref/bin/JabRef" "-n")))
```

```
(require 'ox-jabref)
```

Use this if you have JabRef as a 'jar' file.

```
(custom-set-variables
 '(org-jabref-command '("java -jar ~/Downloads/JabRef-2.9.2.jar" "-n")))
```

```
(require 'ox-jabref)
```

14.2 Example

```
@book{goossens94,
  author = "Michel Goossens and Frank Mittelbach and Alexander Samarin",
  title = "The {\LaTeX}\space companion",
  publisher = "Addison-Wesley",
  year = 1994
}
```

```
@book{kopka95,
  author = "Helmut Kopka and Patrick W. Daly",
  title = "A guide to {\LaTeXe}: document preparation
          for beginners and advanced users",
  publisher = "Addison-Wesley",
  year = 1995
}
```

```
@book{novices,
  author = "Nicola L. C. Talbot",
  title = "{\LaTeX}\space for Complete Novices",
  volume = 1,
  publisher = "Dickmaw Books",
  series = "Dickimaw {\LaTeX}\space Series",
  note = "\url{\baseurl/latex/novices/}",
  isbn="978-1-909440-00-5",
  year = 2012
}
```

```
@book{thesis,
  author = "Nicola L. C. Talbot",
  title = "Using {\LaTeX}\space to Write a Ph.D. Thesis",
  volume = 2,
  publisher = "Dickmaw Books",
  series = "The Dickimaw {\LaTeX}\space Series",
  note = "\url{\baseurl/latex/thesis/}",
  year = 2012
}
```

```
#+bib_file: "./biblatex-examples/novices.bib"
```

```
#+odt_jabref_citation_style: "Chicago (author-date)"
```

```
: Nicola L. C. Talbot in his book [cite:@novices; p. 97] says,
```

```
#+begin_quote
```

```
"If you have a large number of citations in your document, it's best
to use an external bibliographic application, such as =bibtex= or
=biber=. However, that is beyond the scope of this book. See,
instead, /A Guide to LaTeX/ \cite{kopka95}, /The LaTeX Companion/
\cite{goossens94} or [[http://www.dickimaw-books.com/latex/thesis/][Using LaTeX to Wri
[cite:@thesis]."
```

```
#+end_quote
```

```
#+bibliography:
```

14.3 Sample Outputs

Sample output with `#+ODT_JABREF_CITATION_STYLE: "Numeric"`

Nicola L. C. Talbot in his book [1, p. 97] says,

"If you have a large number of citations in your document, it's best to use an external bibliographic application, such as `bibtex` or `biber`. However, that is beyond the scope of this book. See, instead, *A Guide to LaTeX* [2], *The LaTeX Companion* [3] or [Using LaTeX to Write a PhD Thesis](#) [4]."

References

- [1] Talbot, Nicola L. C.. 2012. *LaTeX\space for Complete Novices*. Vol. 1. : Dickmaw Books.
- [2] Kopka, Helmut and Daly, Patrick W.. 1995. *A guide to LaTeX\space: document preparation for beginners and advanced users*. : Addison-Wesley.
- [3] Goossens, Michel, Mittelbach, Frank and Samarin, Alexander. 1994. *The LaTeX\space companion*. : Addison-Wesley.
- [4] Talbot, Nicola L. C.. 2012. *Using LaTeX\space to Write a Ph.D. Thesis*. Vol. 2. : Dickmaw Books.

Sample output with `#+ODT_JABREF_CITATION_STYLE: "Chicago (full-note)"`

Nicola L. C. Talbot in his book ¹ says,

"If you have a large number of citations in your document, it's best to use an external bibliographic application, such as `bibtex` or `biber`. However, that is beyond the scope of this book. See, instead, *A Guide to LaTeX* ², *The LaTeX Companion* ³ or [Using LaTeX to Write a PhD Thesis](#) ⁴."

References

Talbot, Nicola L. C.. *LaTeXspace for Complete Novices*. Vol. 1. : Dickmaw Books, 2012.

Kopka, Helmut and Daly, Patrick W.. *A guide to LaTeX: document preparation for beginners and advanced users*. : Addison-Wesley, 1995.

Goossens, Michel, Mittelbach, Frank and Samarin, Alexander. *The LaTeXspace companion*. : Addison-Wesley, 1994.

Talbot, Nicola L. C.. *Using LaTeXspace to Write a Ph.D. Thesis*. Vol. 2. : Dickmaw Books, 2012.

¹ Nicola L. C. Talbot, *LaTeXspace for Complete Novices* { }, vol. 1 : Dickmaw Books, 2012), p. 97.

² Helmut Kopka and Patrick W. Daly, *A guide to LaTeX: document preparation for beginners and advanced users* { } : Addison-Wesley, 1995).

³ Michel Goossens, Frank Mittelbach and Alexander Samarin, *The LaTeXspace companion* { } : Addison-Wesley, 1994).

⁴ Nicola L. C. Talbot, *Using LaTeXspace to Write a Ph.D. Thesis* { }, vol. 2 : Dickmaw Books, 2012).

Sample output with `#+ODT_JABREF_CITATION_STYLE: "Chicago (author-date)"`

Nicola L. C. Talbot in his book (Talbot 2012: 1, p. 97) says,

"If you have a large number of citations in your document, it's best to use an external bibliographic application, such as `bibtex` or `biber`. However, that is beyond the scope of this book. See, instead, *A Guide to LaTeX* (Kopka and Daly 1995), *The LaTeX Companion* (Goossens et al. 1994) or [Using LaTeX to Write a PhD Thesis](#) (Talbot 2012: 2)."

References

Talbot, Nicola L. C.. 2012. *LaTeXspace for Complete Novices*. Vol. 1. : Dickmaw Books.

Kopka, Helmut and Daly, Patrick W.. 1995. *A guide to LaTeX: document preparation for beginners and advanced users*. : Addison-Wesley.

Goossens, Michel, Mittelbach, Frank and Samarin, Alexander. 1994. *The LaTeXspace companion*. : Addison-Wesley.

Talbot, Nicola L. C.. 2012. *Using LaTeXspace to Write a Ph.D. Thesis*. Vol. 2. : Dickmaw Books.

15 Advanced topics in ODT export

If you rely heavily on ODT export, you may want to exploit the full set of features that the exporter offers. This section describes features that would be of interest to power users.

15.1 Configuring a document converter

The ODT exporter can work with popular converters with little or no extra configuration from your side. See [Chapter 5 \[Extending ODT export\], page 8](#). If you are using a converter that is not supported by default or if you would like to tweak the default converter settings, proceed as below.

1. Register the converter

Name your converter and add it to the list of known converters by customizing the variable `'org-odt-convert-processes'`. Also specify how the converter can be invoked via command-line to effect the conversion.

2. Configure its capabilities

Specify the set of formats the converter can handle by customizing the variable `'org-odt-convert-capabilities'`. Use the default value for this variable as a guide for configuring your converter. As suggested by the default setting, you can specify the full set of formats supported by the converter and not limit yourself to specifying formats that are related to just the OpenDocument Text format.

3. Choose the converter

Select the newly added converter as the preferred one by customizing the variable `'org-odt-convert-process'`.

15.2 Working with OpenDocument style files

This section explores the internals of the ODT exporter and the means by which it produces styled documents. Read this section if you are interested in exploring the automatic and custom OpenDocument styles used by the exporter.

15.2.1 Factory styles

The ODT exporter relies on two files for generating its output. These files are bundled with the distribution under the directory pointed to by the variable `'org-odt-styles-dir'`. The two files are:

`'OrgOdtStyles.xml'`

This file contributes to the `'styles.xml'` file of the final 'ODT' document. This file gets modified for the following purposes:

1. To control outline numbering based on user settings.
2. To add styles generated by `'htmlfontify.el'` for fontification of code blocks.

`'OrgOdtContentTemplate.xml'`

This file contributes to the `'content.xml'` file of the final 'ODT' document. The contents of the Org outline are inserted between the `'<office:text>' ... '</office:text>'` elements of this file.

Apart from serving as a template file for the final `content.xml`, the file serves the following purposes:

1. It contains automatic styles for formatting of tables which are referenced by the exporter.
2. It contains `<text:sequence-decl>` ... `</text:sequence-decl>` elements that control how various entities - tables, images, equations etc - are numbered.

15.2.2 Overriding factory styles

The following two variables control the location from which the ODT exporter picks up the custom styles and content template files. You can customize these variables to override the factory styles used by the exporter.

`org-odt-styles-file`

Use this variable to specify the `styles.xml` that will be used in the final output. You can specify one of the following values:

1. A `styles.xml` file
Use this file instead of the default `styles.xml`
2. A `.odt` or `.ott` file
Use the `styles.xml` contained in the specified OpenDocument Text or Template file
3. A `.odt` or `.ott` file and a subset of files contained within them
Use the `styles.xml` contained in the specified OpenDocument Text or Template file. Additionally extract the specified member files and embed those within the final 'ODT' document.
Use this option if the `styles.xml` file references additional files like header and footer images.
4. `nil`
Use the default `styles.xml`

`org-odt-content-template-file`

Use this variable to specify the blank `content.xml` that will be used in the final output.

15.3 Creating one-off styles

There are times when you would want one-off formatting in the exported document. You can achieve this by embedding raw OpenDocument XML in the Org file. The use of this feature is better illustrated with couple of examples.

1. Embedding ODT tags as part of regular text

You can include simple OpenDocument tags by prefixing them with `@`. For example, to highlight a region of text do the following:

```
@<text:span text:style-name="Highlight">This is a
highlighted text@</text:span>. But this is a
regular text.
```


Hint: To see the above example in action, edit your ‘styles.xml’ (see [Section 15.2.1 \[Factory styles\]](#), page 30) and add a custom ‘Highlight’ style as shown below.

```
<style:style style:name="Highlight" style:family="text">
  <style:text-properties fo:background-color="#ff0000"/>
</style:style>
```

2. Embedding a one-line OpenDocument XML

You can add a simple OpenDocument one-liner using the ‘#+ODT:’ directive. For example, to force a page break do the following:

```
#+ODT: <text:p text:style-name="PageBreak"/>
```

Hint: To see the above example in action, edit your ‘styles.xml’ (see [Section 15.2.1 \[Factory styles\]](#), page 30) and add a custom ‘PageBreak’ style as shown below.

```
<style:style style:name="PageBreak" style:family="paragraph"
  style:parent-style-name="Text_20_body">
  <style:paragraph-properties fo:break-before="page"/>
</style:style>
```

3. Embedding a block of OpenDocument XML

You can add a large block of OpenDocument XML using the ‘#+BEGIN_ODT’ ... ‘#+END_ODT’ construct.

For example, to create a one-off paragraph that uses bold text, do the following:

```
+++BEGIN_EXPORT ODT
<text:p text:style-name="Text_20_body_20_bold">
This paragraph is specially formatted and uses bold text.
</text:p>
+++END_EXPORT ODT
```

15.4 Customizing tables in ODT export

You can override the default formatting of the table by specifying a custom table style with the ‘#+ATTR_ODT’ line. For a discussion on default formatting of tables see [Chapter 9 \[Tables in ODT export\]](#), page 16.

This feature closely mimics the way table templates are defined in the OpenDocument-v1.2 specification.¹

15.4.1 Custom table styles - an illustration

To have a quick preview of this feature, install the below setting and export the table that follows.

```
(setq org-odt-table-styles
  (append org-odt-table-styles
    '(("TableWithHeaderRowAndColumn" "Custom"
      ((use-first-row-styles . t)
       (use-first-column-styles . t)))
      ("TableWithFirstRowandLastRow" "Custom"
      ((use-first-row-styles . t)
```

¹ [OpenDocument-v1.2 Specification](#)

```

                (use-last-row-styles . t))))))
#+ATTR_ODT: :style "TableWithHeaderRowAndColumn"
| Name | Phone | Age |
| Peter | 1234 | 17 |
| Anna | 4321 | 25 |

```

In the above example, you used a template named ‘Custom’ and installed two table styles with the names ‘TableWithHeaderRowAndColumn’ and ‘TableWithFirstRowandLastRow’. (**Important:** The OpenDocument styles needed for producing the above template have been pre-defined for you. These styles are available under the section marked ‘Custom Table Template’ in `OrgOdtContentTemplate.xml` (see [Section 15.2.1 \[Factory styles\]](#), page 30). If you need additional templates you have to define these styles yourselves.

15.4.2 Custom table styles - the nitty-gritty

To use this feature proceed as follows:

1. Create a table template²

A table template is nothing but a set of ‘table-cell’ and ‘paragraph’ styles for each of the following table cell categories:

- Body
- First column
- Last column
- First row
- Last row
- Even row
- Odd row
- Even column
- Odd Column

The names for the above styles must be chosen based on the name of the table template using a well-defined convention.

The naming convention is better illustrated with an example. For a table template with the name ‘Custom’, the needed style names are listed in the following table.

Table cell type	‘table-cell’ style	‘paragraph’ style
Body	‘CustomTableCell’	‘CustomTableParagraph’
First column	‘CustomFirstColumnTableCell’	‘CustomFirstColumnTableParagraph’
Last column	‘CustomLastColumnTableCell’	‘CustomLastColumnTableParagraph’
First row	‘CustomFirstRowTableCell’	‘CustomFirstRowTableParagraph’
Last row	‘CustomLastRowTableCell’	‘CustomLastRowTableParagraph’
Even row	‘CustomEvenRowTableCell’	‘CustomEvenRowTableParagraph’
Odd row	‘CustomOddRowTableCell’	‘CustomOddRowTableParagraph’
Even column	‘CustomEvenColumnTableCell’	‘CustomEvenColumnTableParagraph’

² See the ‘<table:table-template>’ element of the OpenDocument-v1.2 specification

Odd column ‘CustomOddColumnTableCell’ ‘CustomOddColumnTableParagraph’

To create a table template with the name ‘Custom’, define the above styles in the ‘<office:automatic-styles>’ ... ‘</office:automatic-styles>’ element of the content template file (see [Section 15.2.1 \[Factory styles\]](#), page 30).

2. Define a table style³

To define a table style, create an entry for the style in the variable ‘org-odt-table-styles’ and specify the following:

- the name of the table template created in step (1)
- the set of cell styles in that template that are to be activated

For example, the entry below defines two different table styles ‘TableWithHeaderRowAndColumn’ and ‘TableWithFirstRowandLastRow’ based on the same template ‘Custom’. The styles achieve their intended effect by selectively activating the individual cell styles in that template.

```
(setq org-odt-table-styles
      (append org-odt-table-styles
              '(("TableWithHeaderRowAndColumn" "Custom"
                ((use-first-row-styles . t)
                 (use-first-column-styles . t)))
              ("TableWithFirstRowandLastRow" "Custom"
                ((use-first-row-styles . t)
                 (use-last-row-styles . t))))))
```

3. Associate a table with the table style

To do this, specify the table style created in step (2) as part of the ‘ATTR_ODT’ line as shown below.

```
#+ATTR_ODT: :style "TableWithHeaderRowAndColumn"
| Name | Phone | Age |
| Peter | 1234 | 17 |
| Anna | 4321 | 25 |
```

15.5 Validating OpenDocument XML

Occasionally, you will discover that the document created by the ODT exporter cannot be opened by your favorite application. One of the common reasons for this is that the ‘.odt’ file is corrupt. In such cases, you may want to validate the document against the OpenDocument RELAX NG Compact Syntax (RNC) schema.

For de-compressing the ‘.odt’ file⁴: See [Section “File Archives” in emacs](#). For general help with validation (and schema-sensitive editing) of XML files: See [Section “Introduction” in nxml-mode](#).

If you have ready access to OpenDocument ‘.rnc’ files and the needed schema-locating rules in a single folder, you can customize the variable ‘org-odt-schema-dir’

³ See the attributes ‘table:template-name’, ‘table:use-first-row-styles’, ‘table:use-last-row-styles’, ‘table:use-first-column-styles’, ‘table:use-last-column-styles’, ‘table:use-banding-rows-styles’, and ‘table:use-banding-column-styles’ of the ‘<table:table>’ element in the OpenDocument-v1.2 specification

⁴ ‘.odt’ files are nothing but ‘zip’ archives

to point to that directory. The ODT exporter will take care of updating the `'rng-schema-locating-files'` for you.

16 Main Index

A

‘ATTR_ODT’, keyword 16, 17, 18, 32, 34

B

‘BIB_FILE’, keyword 23

‘BIBLIOGRAPHY’, keyword 23

O

‘ODT_JABREF_CITATION_STYLE’, keyword 23

‘ODT_STYLES_FILE’, keyword 9

U

unconv 8

17 Key Index

C-c C-e o..... 7 C-c C-e 0..... 7

18 Command and Function Index

<code>org-export-as-odf</code>	19	<code>org-odt-convert</code>	8
<code>org-export-as-odf-and-open</code>	19	<code>org-odt-export-to-odt</code>	7

19 Variable Index

<code>org-latex-to-mathml-convert-command</code>	19	<code>org-odt-create-custom-styles-for-srcblocks</code>	
<code>org-latex-to-mathml-jar-file</code>	19	22
<code>org-odt-category-strings</code>	21	<code>org-odt-fontify-srcblocks</code>	22
<code>org-odt-convert-capabilities</code>	30	<code>org-odt-pixels-per-inch</code>	17
<code>org-odt-convert-process</code>	8, 30	<code>org-odt-preferred-output-format</code>	7, 8
<code>org-odt-convert-processes</code>	30	<code>org-odt-styles-dir</code>	30
		<code>org-odt-styles-file</code>	9

20 What is New

Chapter 14 [Bibliography and Citations in ODT export], page 23

Use JabRef to produce ODT documents with Citation and Bibliography

Chapter 8 [List Tables in ODT export], page 11

Create a table with multi-paragraph content using a ‘**List table**’