The Org/OpenDocument Text Exporter Manual

Release 7.7

by Jambunathan K

This manual is for Org version 7.7.

Table of Contents

1 Exporting 1

1.1 OpenDocument Text export 1
1.1.1 Installing ODT exporter 1
1.1.2 ODT export commands 1
1.1.3 Exporting and Converting to Other formats 2
1.1.3.1 Configuring a converter 2
1.1.3.2 Using the converter 3
1.1.4 Applying Custom Styles 3
1.1.5 Links in ODT export 4
1.1.6 Tables in ODT export 5
1.1.7 Images in ODT export 6
1.1.8 Math formatting in ODT export 7
1.1.9 Literal Examples in ODT export 8
1.1.10 Working with raw OpenDocument XML 8
1.1.11 Additional documentation 9

Concept index 10

Key index 11

Command and function index 12

Variable index 13

1 Exporting

Org-mode documents can be exported into a variety of other formats. For printing and sharing of notes, ASCII export produces a readable and simple version of an Org file. HTML export allows you to publish a notes file on the web, while the XOXO format provides a solid base for exchange with a broad range of other applications. LaTeX export lets you use Org-mode and its structured editing functions to easily create LaTeX files. DocBook export makes it possible to convert Org files to many other formats using DocBook tools. OpenDocument Text(ODT) export allows seamless colloboration across organizational boundaries. For project management you can create gantt and resource charts by using TaskJuggler export. To incorporate entries with associated times like deadlines or appointments into a desktop calendar program like iCal, Org-mode can also produce extracts in the iCalendar format. Currently Org-mode only supports export, not import of these different formats.

Org supports export of selected regions when `transient-mark-mode` is enabled (default in Emacs 23).

1.1 OpenDocument Text export

Orgmode[1] supports export to OpenDocument Text(ODT) format using '`org-odt.el`' module. Documents created by this exporter use *OpenDocument-v1.2 specification*[2] and are compatible with LibreOffice 3.4.

1.1.1 Installing ODT exporter

Obtaining and Installing ODT exporter

The ODT exporter can be enabled in one of the following ways based on your mode of installation.

1. If you have downloaded Org, either as a distribution '`.zip`' or '`.tar`' file, or as a '`git`' archive, add the '`contrib`' subdir to `load-path` and customize variable `org-modules` to include the '`odt`' option.
2. If you are using Org that comes bundled with Emacs, then you can install the '`org-odt`' package using the package manager (See Info file '`emacs`', node '`Packages`').

Pre-requisites for ODT exporter

ODT exporter relies on '`zip`' program to create the final output. Check the availability of this program before proceeding further.

1.1.2 ODT export commands

Exporting to ODT

C-c C-e o `org-export-as-odt`

Export as OpenDocument Text file. If `org-export-odt-preferred-output-format` is specfied, automatically convert the exported file to that format. See [Automatically Exporting to Other formats], page 2.

[1] Versions 7.6 or later

[2] Open Document Format for Office Applications (OpenDocument) Version 1.2

For an Org file, 'myfile.org', the ODT file will be 'myfile.odt'. The file will be overwritten without warning. If there is an active region[3], only the region will be exported. If the selected region is a single tree[4], the tree head will become the document title. If the tree head entry has, or inherits, an EXPORT_FILE_NAME property, that name will be used for the export.

C-c C-e O org-export-as-odt-and-open

Export as OpenDocument Text file and open the resulting file. If org-export-odt-preferred-output-format is specified, open the converted file instead. See [Automatically Exporting to Other formats], page 2.

Automatically Exporting to Other formats

Very often, you will find yourself exporting to ODT format, only to immediately save the exported document to a different format like 'doc', 'pdf' etc. In such cases, you will find it convenient to configure a converter (see Section 1.1.3.1 [Configuring a converter], page 2) and specify your preferred output format by customizing the variable org-export-odt-preferred-output-format. This way the export commands (see [Exporting to ODT], page 1) can be extended to also export to the preferred format.

1.1.3 Exporting and Converting to Other formats

ODT exporter adds support for exporting Org outlines to formats that are not supported natively by Org. It also adds support to convert document from one format to another. To use these features, you need to configure a command-line converter.

1.1.3.1 Configuring a converter

Pre-configured converters

The ODT exporter supports two converters out of the box:

1. 'unoconv'

   This converter is available as an installable package in your favorite distribution.

2. 'BasicODConverter'

   This converter is distributed as a LibreOffice extension and can be found in the your Org distribution. See the subdirectory pointed to by the variable org-odt-data-dir.

Installing a new converter

If you prefer to use a converter other than the two mentioned above, then you may have to do additional configuration. You can proceed as follows:

1. Register the converter

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

2. Configure it's capabilities

[3] This requires transient-mark-mode to be turned on

[4] To select the current subtree, use *C-c @*.

Specify the set of formats the converter can handle by customizing the variable `org-export-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 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-export-odt-convert-process`.

1.1.3.2 Using the converter

Once a command-line converter is configured you can use it to extend the list of formats to which Org can export to. See [Automatically Exporting to Other formats], page 2. You can also use it to perform one-off document conversion as detailed below.

M-x org-export-odt-convert

Convert an existing document from one format to another format as determined by variable `org-export-odt-convert-capabilities` (see [Configure converter capabilities], page 2).

Note that you can use this command to even convert documents that is produced outside of Org and in formats that is different from ODT format.

1.1.4 Applying Custom Styles

A note on the internals

ODT exporter relies on two files for generating it's output. These files are bundled under the '`styles`' subdirectory of the data directory (See variable `org-odt-data-dir`). The two files are:

- '`OrgOdtStyles.xml`'

  This file contributes to '`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 the '`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 is 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.

Overriding the default styles

The default styles that ship with the ODT exporter would suffice for generating well-formatted document. However it may not cater to your specific tastes. If this is the case, you can replace the factory defaults with your own by customizing the following variables:

- `org-export-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`' references additional files like header and footer images.

  4. `nil`

     Use the default '`styles.xml`'

- `org-export-odt-content-template-file`

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

Caution: For best results with custom styles, you need to ensure that all style names emitted by the ODT exporter be apriori defined in '`styles.xml`' and the template '`content.xml`' files. Unless sufficient care is exercised in choosing the custom style files, the result could be less than satisfactory. So it is highly recommended that you build your custom '`styles.xml`' from the default '`styles.xml`' bundled with the exporter.

Specifying Custom Styles on per-file basis

You can use `#+ODT_STYLES_FILE` option to specify custom styles on per-file basis. This option effectively overrides the value of `org-export-odt-styles-file` with the specified value just for this buffer. A typical setting will look like

```
#+ODT_STYLES_FILE: "/path/to/styles.xml"
```

or

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

1.1.5 Links in ODT export

ODT exporter creates cross-references (aka bookmarks) for links that are destined locally. It creates internet style links for all other links.

1.1.6 Tables in ODT export

Export of native Org-mode tables (See Info file ‘org’, node ‘Tables’) and simple ‘table.el’ tables is supported. However export of complex ‘table.el’ tables - tables that have column or row spans - are not supported. Such tables are stripped from the exported document.

By default, a table is exported with with top and bottom frames and with rules separating row and column groups (See Info file ‘org’, node ‘Column groups’). If the table specifies alignment and relative width for it’s columns (See Info file ‘org’, node ‘Column width and alignment’) then these are honored on export[5].

You can override the default formatting of the table by specifying a custom table style with the #+ATTR_ODT line.

This feature closely mimics the way table templates are defined in the OpenDocument-v1.2 specification[6].

To use this feature proceed as follows:

1. Create a table template[7]

   A table template is nothing but a set of ‘table-cell’ and ‘paragraph’ style 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 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’

[5] The column widths are interpreted as weighted ratios with the default weight being 1

[6] OpenDocument-v1.2 Specification

[7] See <table:table-template> element of OpenDocument-v1.2 specification

Odd column 'CustomOddColumnTableCell' 'CustomOddColumnTableParagraph'

To create a table template with name 'Custom', define the above styles in the <office:automatic-styles>...</office:automatic-styles> element of the content template file (see docstring of variable org-export-odt-content-template-file).

2. Define a table style[8]

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

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

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

```
(setq org-export-odt-table-styles
      '(("TableWithHeaderRowsAndColumns"
 "Custom"
 ((use-first-row-styles . t)
  (use-first-column-styles . t)))
("TableWithHeaderColumns"
 "Custom" ((use-first-column-styles . t)))))
```

3. Associate a table with the table style

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

```
#+ATTR_ODT: TableWithHeaderColumns
| Name  | Phone | Age |
| Peter |  1234 |  17 |
| Anna  |  4321 |  25 |
```

1.1.7 Images in ODT export

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]]
```

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

[8] See 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, table:use-banding-column-styles - of <table:table> element in OpenDocument-v1.2 specification

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

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

How image size is computed

In order to scale the embedded images, the exporter needs to compute the size of the image. This is done by retrieving the image size in pixels and converting the pixel units to centimetres using `org-export-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.

Sizing and scaling of embedded images

Note that the exporter specifies the desired size of the image in the final document in units of centimetres. To compute the size of the original image in centimetres, the To convert the image size in pixels to equivalent units in cms `org-export-odt-pixels-per-inch` is used.

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 it's size, do the following:

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

Scale the image to a specific width
: To embed '`img.png`' to occupy a width of 10 cm while retaining the original height:width ratio, do the following:

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

Scale the image to a specific height
: To embed '`img.png`' to occupy a height of 10 cm while retaining the original height:width ratio, do the following

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

1.1.8 Math formatting in ODT export

LaTeX math snippets (See Info file '`org`', node '`LaTeX fragments`') can be embedded in the ODT document using 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 in to MathML fragments using an external LaTeX-to-MathML converter program. The resulting MathML fragments are then embedded as a 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 '`MathToWeb`'[9] 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")
```

2. png

   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.

1.1.9 Literal Examples in ODT export

Export of Literal examples (See Info file '`org`', node '`Literal examples`') with full fontification is supported. This feature is enabled by default and is activated automatically if an enhanced version of '`htmlfontify.el`' is available in the `load-path`[10].

The character styles used for fontification of the Literal blocks are auto-generated by the exporter in conjunction with '`htmlfontify.el`' library and need not be included in the default '`styles.xml`' file. These auto-generated styles have '`OrgSrc`' prefix and inherit their color based on the face used by Emacs `font-lock` library.

If you prefer to use your own custom styles for fontification and disable their auto-generation altogether, you can do so by customizing the variable `org-export-odt-create-custom-styles-for-srcblocks`.

You can turn off fontification support for Literal examples by customizing the variable `org-export-odt-fontify-srcblocks`.

1.1.10 Working with raw OpenDocument XML

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 prepending them with 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`' and add a custom '`Highlight`' style as shown below.

[9] See MathToWeb

[10] '`htmlfontify.el`' that ships with standard Emacs <= 24.1 has no support for ODT fontification

```
<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`' 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_ODT
   <text:p text:style-name="Text_20_body_20_bold">
   This paragraph is specially formatted and uses bold text.
   </text:p>
   #+END_ODT
   ```

1.1.11 Additional documentation

The ODT exporter is still in development. For up to date information, please follow Org mailing list emacs-orgmode@gnu.org closely.

Concept index

#

#+ATTR_ODT 5
#+ODT_STYLES_FILE 4

A

active region 1

B

'BasicODConverter' 2

C

convert 2
converter 2

D

doc, docx 2
dvipng 8

E

embedding images in ODT 6
export, OpenDocument 1
exporting 1

I

images, embedding in ODT 6

K

K, Jambunathan 1

L

LibreOffice 1

M

MathML 7

O

ODT 1
OpenDocument 1
org-modules 1
org-odt.el 1

P

property EXPORT_FILE_NAME 1

R

region, active 1

S

styles, custom 3

T

tables, in DocBook export 4, 5
template, custom 3
transient-mark-mode 1

U

'unoconv' 2

Z

zip 1

Key index

C-c C-e o 1
C-c C-e O 2

Command and function index

org-export-as-odt 1
org-export-as-odt-and-open 2

Variable index

This is not a complete index of variables and faces, only the ones that are mentioned in the manual. For a more complete list, use *M-x org-customize RET* and then click yourself through the tree.

org-export-odt-convert 3
org-export-odt-convert-capabilities 2
org-export-odt-convert-process 3
org-export-odt-convert-processes 2
org-export-odt-create-custom-styles-for-srcblocks 8
org-export-odt-fontify-srcblocks 8
org-export-odt-preferred-output-format .. 1, 2
org-export-odt-styles-file 3
org-export-odt-table-styles 6
org-latex-to-mathml-convert-command 7
org-latex-to-mathml-jar-file 7
org-odt-data-dir 2