# Day One Export Documentation

***Release 0.8.3***

**Nathan Grigg**

**Apr 11, 2017**

# Contents

**Note:** The Day One Export project is hosted at http://github.com/nathangrigg/dayone_export.

Go there for installation instructions, to report issues, or if you are interested in contributing.

CHAPTER 1

# Use the command line tool

## Basic Usage

```
usage: dayone_export [--output FILE] [opts] journal

Export Day One entries using a Jinja template

positional arguments:
journal             path to Day One journal package

optional arguments:
-h, --help          show this help message and exit
--output FILE       file to write (default print to stdout). Using strftime
                    syntax will produce multiple output files with entries
                    grouped by date.
--format FMT        output format (default guess from output file extension)
--template NAME     name or file of template to use
--template-dir DIR  location of templates (default ~/.dayone_export)
--tags TAGS         export entries with these comma-separated tags. Tag
                    'any' has a special meaning.
--exclude TAGS      exclude entries with these comma-separated tags
--after DATE        export entries published on or after this date
--before DATE       export entries published before this date
--reverse           display in reverse chronological order
--autobold          autobold first lines (titles) of posts
--nl2br             convert each new line to a <br>
--version           show program's version number and exit

If the Day One package has photos, you may need to copy the "photos" folder
from the package into the same directory as the output file.
```

## Use a custom template

Use the `--template` option to specify a custom template.

For information on how to create templates, see *Create your own template*.

## Change the default template

You can override the default template by creating a `default.html` file and placing it in the folder `~/.dayone_export`.

You can also create default templates of other types in a similar manner. For example, `default.tex` would be a default LaTeX template. The default markdown template should be called *default.md*.

The program uses the extension of the output file to determine which default template to use. If there is no output file, use the `--format` option to specify the format.

If you wish to use a directory other than `~/.dayone_export`, as the location for default templates, you can use the `--template-dir` option.

## Filter by tag

Use the `--tags` option with a comma-separated list of tags to include.

If you use the option `--tags any`, then any entry with at least one tag will be included.

Also, you can exclude entries with specified tags, by using the `--exclude` option. Note that `--exclude` has a priority over `--tags`.

## Limit export to recent entries

Use the `--after` option to only export entries after a certain date.

For best results, use some kind of standard form for the date (e.g. `2012-03-04`).

## Markdown options

The `--autobold` option will convert the first line of each post into a heading, as long as it is relatively short (similar to the way Day One optionally can)

The `--nl2br` option will insert a `<br>` tag after each new line.

## Link to or embed photos

The default html template refers to photos by their relative names. To show the photos in the output file, you will need to copy the `photos` directory from inside the *Journal.dayone* package into the same directory as the output html file.

There is an alternate template which embeds photos directly into the html file as base64-encoded images. To use this template, use the option `--template imgbase64.html`.

## Template filenames and grouping

The `--output` option specifies the output filename if you want something other than stdout.

It also has another feature: you can include strftime-style formatting codes, in which case multiple files will be produced, each containing the journal entries with timestamps that result in the same filename.

Examples:

`--output journal_%Y_%m.md` will produces monthly files named journal_2013_02.md etc.

`--output diary_%a.html` will produce a separate file for each weekday.

Note that if you want a literal `%` in your output filename, you will need to escape it as `%%`.

CHAPTER 2

# Create your own template

Templates are written using Jinja2 syntax. You can learn a lot from their excellent Template Designer Documentation

When making your own template, a great place to start from is one of Day One Export's built-in templates.

## The journal variable

The most important variable that the program passes to the template is named `journal`. This is a list of *Entry* objects, each of which represents a single journal entry.

Generally, a template will loop over elements of the journal variable, like this:

```
... Document header, title, etc ...

{% for entry in journal %}

    ... Code for a single entry ...

{% endfor %}

... End of document stuff, etc ...
```

## Other variables

- `today`: The current date.

## The Entry object

An Entry object behaves a lot like a Python dictionary, which means you can access the data fields by name. For example, you use `entry['Text']` to get the text of an entry.

Jinja uses double braces to insert a variable into the document, so to insert the entry's text at a certain point in the document, you would include the following line in your template:

```
{{ entry['Text'] }}
```

Here are some keys that an entry may have:

- **Basic information:**
  - `Date`
  - `Text`
  - `Starred` (boolean)
  - `UUID`
  - `Activity`: Description of motion activity, e.g. "Stationary"
  - `Step Count`: Number of steps from the motion sensor
  - `Photo` (the relative path of the corresponding photo, if it exists)
- **Information about the location:**
  - `Place Name` (e.g. Boom Noodle)
  - `Locality` (e.g. Seattle)
  - `Administrative Area` (e.g. Washington)
  - `Country` (e.g. United States)
  - `Longitude`
  - `Latitude`
- **Information about the currently-playing music:**
  - `Album`
  - `Artist`
  - `Track`
- **Information about weather:**
  - `Fahrenheit`
  - `Celsius`
  - `Description`
  - `IconName`
  - `Sunrise Date`: The date and time of sunrise
  - `Sunset Date`: The date and time of sunset
  - `Visibility KM`
  - `Relative Humidity`
  - `Pressure MB`
  - `Wind Bearing`
  - `Wind Chill Celsius`
  - `Wind Speed KPH`

- **Information about the creation device:**
  - `Device Agent`
  - `Host Name`
  - `OS Agent`
  - `Software Agent`

Jinja will just leave a blank space if you try to access a nonexistent key. So if an entry has no location information, `{{ entry['Latitude'] }}` will have no effect.

For more information, see the documentation for *The Entry object*.

## Places

You may want to combine the place information into a single string. You can do this with the `place` method.

With no arguments, `entry.place()` inserts the place names in order from smallest to largest, separated by commas.

With a single integer argument, `entry.place(n)` inserts the place names from smallest to largest, but only uses the *n* smallest places. For example, `entry.place(3)` will always leave off the country.

If you want to get more specific, you can use a list as an argument. So `entry.place([1, 3])` will put the *Locality* and *Country*, but leave off the *Place Name* and *Administrative Area*.

Finally, you can use an `ignore` keyword argument to ignore a specific string. For example, `entry.place(ignore="United States")` will print the full location information, but leave off the country if it is "United States".

Don't forget that to insert any of this into the document, you need to put it inside double braces.

More information is available in the documentation for *The Entry object*.

## Weather

You may want to combine the weather into a single string. You can do this with the `weather` method.

The `weather` method takes one parameter to display the temperature as celcius or fahrenheit. For example, `entry.weather('F')` will display the temperature in fahrenheit. The same can be done for celsius but with `entry.weather('C')`.

Don't forget that to insert any of this into the document, you need to put it inside double braces.

More information is available in the documentation for *The Entry object*.

## Jinja Filters

Jinja allows you to transform a variable before inserting it into the document, using a filter which is denoted by a `|`.

For example, `{{ entry['Country'] | default("Unknown") }}` pass the Country through the `default` filter, which in turn changes it to the string `Unknown` if the country does not exist.

Since the `default` filter can be particularly useful, I will point out that it may happen that Day One has defined the country to be the empty string, in which case, the `default` filter will let it remain empty. If you want the filter to be

more aggressive (you probably do), you can use `{{ entry['Country'] | default("Unknown", true) }}`

There are several built-in Jinja filters available.

## Format dates

This program defines a custom filter called `format` which formats dates.

For example:

```
{{ entry['Date'] | format('%Y-%m-%d %H:%M:%S %z') }}
```

The `format` filter also accepts an optional timezone argument, which overrides the native timezone of every entry. For example:

```
{{ entry['Date'] | format('%-I:%M %p %Z', tz='America/Los_Angeles') }}
```

displays the date in US Pacific time, regardless of the timezone where the entry was recorded.

## Convert to Markdown

This program defines a custom filter called `markdown` which converts markdown text to html:

```
{{ entry['Text'] | markdown }}
```

## Latex Templates

The standard Jinja template syntax clashes with many Latex control characters. If you create a Latex template, you will need to use different syntax.

In a Latex template, you use `\CMD{...}` instead of `{% ... %}` for block statements and `\VAR{...}` instead of `{{ ... }}` to insert variables. For example:

```
\CMD{for entry in journal}
\section{\VAR{entry['Date'] | format}}
\CMD{endfor}
```

You will also find the `escape_tex` filter useful, which escapes Latex control characters:

```
\VAR{entry['Text'] | escape_tex}
```

Note that the `markdown` filter outputs HTML so should not be used. There is currently no support for converting markdown input to formatted Latex output.

Latex templates must end with the `.tex` extension.

## Inline images with base64 encoding

You can include the images inline with base64 encoding using a custom filter:

```
{{ entry['Photo'] | imgbase64 }}
```

The resulting entry looks like:

```
<img class="entry-photo" src="... ">
```

The base64 data can become quite large in size. If you have the Python imaging library installed, you can resize the images so that the resulting output remains sufficiently small (default maximum size is 400 pixels):

```
{{ entry['Photo'] | imgbase64(800) }}
```

This includes the image inline with a maxium size of 800 pixels.

## More templating information

For more details on Jinja templates, see the Jinja template designer documentation.

CHAPTER 3

# Module documentation

Here is information about the module itself.

You can use this in a Python script by using `import dayone_export`.

## The Entry class

**class** dayone_export.**Entry**(*filename*)

Parse a single journal entry.

**Raises** IOError, KeyError

Acts like a read-only dictionary. The keys are as defined in the plist file by the Day One App, with minor exceptions:

- What Day One calls "Entry Text", we call "Text".
- The "Location", "Weather", "Creator", and "Music" dictionaries are flattened, so that their subkeys are accessible as keys of the main dictionary.
- The "Photo" key is added and contains the path to attached photo.
- The "Date" key is added and contains the localized date.

Note that the "Creation Date" contains a naive date (that is, with no attached time zone) corresponding to the UTC time.

**place**(*levels*, *ignore=None*)

Format entry's location as string, with places separated by commas.

**Parameters**

- **levels** (*list of int*) – levels of specificity to include
- **ignore** (*string or list of strings*) – locations to ignore

The *levels* parameter should be a list of integers corresponding to the following levels of specificity defined by Day One.

- 0: Place Name
- 1: Locality (e.g. city)
- 2: Administrative Area (e.g. state)
- 3: Country

Alternately, *levels* can be an integer *n* to specify the *n* smallest levels.

The keyword argument *ignore* directs the method to ignore one or more place names. For example, you may want to ignore your home country so that only foreign countries are shown.

**keys**()
List all keys.

## Journal parsing and exporting

dayone_export.**parse_journal**(*foldername*[, *reverse=False*])
Return a list of Entry objects, sorted by date

dayone_export.**dayone_export**(*dayone_folder*[, ***kwargs*])
Render a template using entries from a Day One journal.

**Parameters**

- **dayone_folder** (*string*) – Name of Day One folder; generally ends in `.dayone`.
- **reverse** (*bool*) – If true, the entries are formatted in reverse chronological order.
- **tags** (*list of strings*) – Only include entries with the given tags. This paramater can also be the literal string `any`, in which case only entries with tags are included. Tags are interpreted as words at the end of an entry beginning with `#`.
- **exclude** (*list of strings*) – Exclude all entries with given tags.
- **before** (*naive datetime*) – Only include entries on before the given date.
- **after** (*naive datetime*) – Only include entries on or after the given date.
- **format** (*string*) – The file extension of the default template to use.
- **template** (*string*) – Template file name. The program looks for the template first in the current directory, then the template directory.
- **template_dir** (*string*) – Directory containing templates. If not given, the program looks in `~/.dayone_export` followed by the dayone_export package.
- **autobold** (*bool*) – Specifies that the first line of each post should be a heading
- **nl2br** (*bool*) – Specifies that new lines should be translated in to <br>s
- **filename_template** (*string*) – An eventual filename, which can include strftime formatting codes. Each time the result of formatting an entry's timestamp with this changes, a new result will be returned.

**Returns** Iterator yielding (filename, filled_in_template) as strings on each iteration.

CHAPTER 4

# Time Zone Information

The Day One apps store all dates in UTC. Newer versions of Day One also include the time zone where each journal entry was entered.

## Entries without time zone information

Older versions of Day One did not record the current time zone in the journal entry. For these entries, `dayone_export` makes a guess based on the time zone in other entries.

If you would like to manually set the time zone in a journal entry which was recorded with an older version of Day One, insert the following section directly into the entry's plist file:

```
<key>Time Zone</key>
<string>America/Los_Angeles</string>
```

This should be a key in the top-level dictionary. For more guidance on where to place it, look at an entry created by a current version of Day One.

## Time Zone Names

A list of time zone names can be found at http://en.wikipedia.org/wiki/List_of_tz_database_time_zones.

# Index